

CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA  
CELSO SUCKOW DA FONSECA – CEFET/RJ

DIRETORIA DE PESQUISA E PÓS-GRADUAÇÃO  
COORDENADORIA DO PROGRAMA DE PÓS-GRADUAÇÃO EM TECNOLOGIA

DISSERTAÇÃO

APLICAÇÃO DE REDES NEURAIS ARTIFICIAIS EM  
IDENTIFICAÇÃO DE NÃO-LINEARIDADES RÍGIDAS

Wendel Furtado da Silva

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO PROGRAMA DE PÓS-GRADUAÇÃO EM TECNOLOGIA COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM TECNOLOGIA.

Paulo Lúcio Silva de Aquino, D.Sc.  
Orientador

RIO DE JANEIRO, RJ – BRASIL  
JUNHO / 2008

# **Livros Grátis**

<http://www.livrosgratis.com.br>

Milhares de livros grátis para download.

## SUMÁRIO

	Pág.
<b>INTRODUÇÃO</b>	1
<b>I– REVISÃO BIBLIOGRÁFICA</b>	5
<b>I.1– APLICAÇÃO DE REDES NEURAIS ARTIFICIAIS EM SISTEMAS NÃO-LINEARES</b>	5
<b>I.2– REDES NEURAIS ARTIFICIAIS</b>	6
I.2.1– Breve histórico	6
I.2.2– Neurônio biológico	8
I.2.3– Neurônio artificial	11
I.2.4– Funções de ativação	13
I.2.5– Principais arquiteturas de redes neurais artificiais	15
I.2.5.1– Redes neurais estáticas	15
I.2.5.2– Redes neurais recorrentes	17
I.2.6– Métodos de treinamento	18
I.2.6.1– Treinamento supervisionado	18
I.2.6.2– Treinamento não-supervisionado	20
I.2.6.3– Treinamento por reforço	20
I.2.7– Algoritmo de retropropagação de erro	21
I.2.8– Algumas considerações para treinamento	28
I.2.8.1– Número de neurônios da camada escondida	28
I.2.8.2– Inicialização dos pesos	30
I.2.8.3– Modos de treinamento	30
I.2.8.4– Taxa de aprendizagem	31
I.2.8.5– Constante de momento	31
I.2.8.6– Critérios de parada	32
I.2.8.7– Normalização dos dados	33

<b>I.3– SISTEMAS NÃO-LINEARES</b>	34
I.3.1– Algumas considerações em sistemas não-lineares	34
I.3.2– Não-linearidades rígidas	37
I.3.2.1– Saturação	38
I.3.2.2– Zona morta	39
I.3.2.3– Saturação com zona morta	40
I.3.2.4– Liga-desliga	41
I.3.2.5– Liga-desliga com histerese	42
I.3.2.6– Liga-desliga com zona morta	42
<b>II– MATERIAIS E MÉTODOS</b>	44
<b>II.1– RECURSOS UTILIZADOS</b>	44
II.1.1– Hardware	44
II.1.2– Software	44
<b>II.2– DEFINIÇÃO DOS PARÂMETROS DA RNA</b>	45
II.2.1– Número de neurônios da camada escondida	45
II.2.2– Funções de ativação	45
II.2.3– Modo de treinamento	45
II.2.4– Inicialização dos pesos	46
II.2.5– Taxa de aprendizagem e constante de momento	46
II.2.6– Critérios de parada	46
II.2.7– Conjunto de treinamento	47
II.2.7.1– Tamanho do conjunto de treinamento	47
II.2.7.2– Pares de entrada-saída	47
II.2.7.3– Normalização do conjunto de treinamento	55
<b>II.3– METODOLOGIA DE SIMULAÇÃO</b>	55
II.3.1– Geração de padrões entrada-saída	55
II.3.2– Simulação da RNA	56

<b>III– RESULTADOS OBTIDOS</b>	61
<b>III.1– SIMULAÇÕES PADRONIZADAS</b>	61
III.1.1– Simulação de saturação	61
III.1.2– Simulação de zona morta tipo I	64
III.1.3– Simulação de zona morta tipo II	67
III.1.4– Simulação de saturação com zona morta	70
III.1.5– Simulação de liga-desliga	73
III.1.6– Simulação de liga-desliga com histerese	76
III.1.7– Simulação de liga-desliga com zona morta	79
<b>III.2– SIMULAÇÕES COMPLEMENTARES</b>	81
III.2.1– Simulação complementar de zona morta tipo I	82
III.2.2– Simulação complementar de zona morta tipo II	85
III.2.3– Simulação complementar de liga-desliga	88
III.2.4– Simulação complementar de liga-desliga com histerese	91
III.2.5– Simulação complementar de liga-desliga com zona morta	94
<b>IV– DISCUSSÃO DOS RESULTADOS</b>	97
<b>CONCLUSÃO</b>	101
<b>SUGESTÕES DE CONTINUIDADE</b>	103
<b>REFERÊNCIAS BIBLIOGRÁFICAS</b>	104
<b>APÊNDICES</b>	
Apêndice 1: Código-fonte em MATLAB da função geradora de padrões referentes a não-linearidade de saturação	A1
Apêndice 2: Código-fonte em MATLAB da função geradora de padrões referentes a não-linearidade de zona morta tipo I	A2
Apêndice 3: Código-fonte em MATLAB da função geradora de padrões referentes a não-linearidade de zona morta tipo II	A3

Apêndice 4:	Código-fonte em MATLAB da função geradora de padrões referentes a não-linearidade de saturação com zona morta	A4
Apêndice 5:	Código-fonte em MATLAB da função geradora de padrões referentes a não-linearidade de liga-desliga	A5
Apêndice 6:	Código-fonte em MATLAB da função geradora de padrões referentes a não-linearidade de liga-desliga com histerese	A6
Apêndice 7:	Código-fonte em MATLAB da função geradora de padrões referentes a não-linearidade de liga-desliga com zona morta	A7
Apêndice 8:	Código-fonte em MATLAB para simulação da RNA	A8
Apêndice 9:	Tabela padronizada para disposição dos resultados obtidos pelas simulações	A16
Apêndice 10:	Exemplo de relatório com parâmetros utilizados e resultados obtidos	A17
Apêndice 11:	Código-fonte em MATLAB para plotagem dos EQMs da fase de treinamento/teste e indicação do EQM de generalização	A18
Apêndice 12:	Código-fonte em MATLAB para comparação do sinal de saída desejado com o emitido pela RNA	A19
Apêndice 13:	Código-fonte em MATLAB para plotagem dos pesos e bias, referentes à última época de treinamento, para redes com 1 neurônio na camada de entrada, 3 neurônios na camada escondida e 1 neurônio na camada de saída.	A21
Apêndice 14:	Código-fonte em MATLAB para plotagem dos pesos e bias, referentes à última época de treinamento, para redes com 2 neurônios na camada de entrada, 5 neurônios na camada escondida e 1 neurônio na camada de saída.	A23

Ficha catalográfica elaborada pela Biblioteca Central do CEFET-RJ

S586 Silva, Wendel Furtado da  
Aplicação de redes neurais artificiais em identificação de não-linearidades rígidas / Wendel Furtado da Silva.– 2008.  
xvi,106f.+ Apêndices: il., grafs, tabs.; enc.

Dissertação ( Mestrado ) Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, 2008.  
Bibliografia: f. 104--106

1.Redes neurais (Computação) 2.Sistemas não lineares  
3.MATLAB (Programa de computador) I.Título

CDD 006.3

Dedico este trabalho aos meus pais Valdecir e Ana, que nunca mediram esforços para me proporcionar acesso ao saber e à educação, a minha esposa Fátima e ao meu filho Mateus, pela compreensão nos meus vários momentos de ausência e, principalmente, por todo amor e carinho doados sem restrição.



## **Agradecimentos**

- A Deus, por mais este consentimento.

- Ao meu orientador Professor Paulo Lúcio Silva de Aquino, D.Sc., CEFET/RJ, pela genialidade, simplicidade, companheirismo, confiança e, sobretudo, paciência, depositada durante todo o desenvolvimento desta dissertação.

- Às Professoras Maria Luiza Fernandes Velloso, D.Sc., UERJ, e Luciana Faletti Almeida, D.Sc., CEFET/RJ, pelo auxílio, disposição e ensinamentos dispensados durante o desenvolvimento desta dissertação.

- Aos membros da banca examinadora, Professor Alessandro Rosa Lopes Zachy, D.Sc., CEFET/RJ e, mais uma vez, às Professoras Maria Luiza Fernandes Velloso, D. Sc., UERJ, e Luciana Faletti Almeida, D. Sc., CEFET/RJ, por terem lido, analisado e criticado este trabalho no intuito de contribuir para o progresso e o desenvolvimento científico.

- A minha esposa Fátima Cristina, pela compreensão nos momentos em que estive ausente e por todo amor e incentivo que foram transmitidos ao longo desta jornada, ao mesmo tempo em que cuidava de um presente especial dado por Deus.

- Aos meus pais, Valdecir e Ana, que nunca mediram esforços para me proporcionar acesso ao saber e à educação.

- As minhas irmãs Patrícia e Priscila, que cederam com contentamento o próprio computador para a realização de incontáveis simulações.

- Aos meus sogros, Antonio (em memória) e Gladice, pelo apoio incondicional nas tarefas diárias e primeiros cuidados com nosso Mateus.

- A todos os professores do Programa de Pós-graduação em Tecnologia do Centro Federal de Educação Tecnológica Celso Suckow da Fonseca – PPTEC/CEFET/RJ, pelos valiosos conhecimentos transmitidos nas disciplinas que foram ministradas ao longo deste curso.

- A todos os funcionários do Programa, em especial, a Abraão Ferreira e Bráulio Tito dos Santos, pela dedicação e presteza durante os trâmites administrativos.

- A todos os amigos e familiares que me fortaleceram em cada dificuldade encontrada e celebraram cada conquista.
  
- Ao meu colega de curso, Engenheiro Edmo Carlos Correia de Paiva Filho, pelo apoio e tranqüilidade transmitida nos momentos mais difíceis.
  
- Aos amigos do Centro de Instrução Almirante Wandenkolk, Capitão-Tenente Fernando Luiz de França Araújo, Gabriel de Oliveira Valente, Anibal Leonardo Pereira, José Jorge da Silva Araújo, Guilherme Schumann e Maria Souza de Carvalho, pelo apoio e colaboração.
  
- Aos amigos da Companhia Estadual de Águas e Esgotos do Rio de Janeiro, José Jorge Siqueira Barbosa, Elias Tavares Carlos, Rogério Batista Pinudo e Carlos Alberto da Silva Batista, por me permitirem participar dos encontros presenciais.
  
- Ao Centro Federal de Educação Tecnológica Celso Suckow da FONSECA e a Agência Financiadora de Pesquisa CAPES, que forneceram suporte financeiro para a pesquisa.
  
- Ao longo deste trabalho recebi o auxílio de grande número de pessoas, seja através de informações, discussões técnicas, apoio efetivo, enfim foram tantas e em momentos tão distintos que certamente algumas não foram lembradas.
  
- A todos, o mais sincero obrigado.

Resumo da dissertação submetida ao PPTEC/CEFET-RJ como parte dos requisitos necessários para obtenção do grau de mestre em tecnologia (M.T.).

## APLICAÇÃO DE REDES NEURAIS ARTIFICIAIS EM IDENTIFICAÇÃO DE NÃO-LINEARIDADES RÍGIDAS

Wendel Furtado da Silva

Junho de 2008

Orientador: Paulo Lucio Silva de Aquino, D. Sc.

Programa: PPTEC

Este trabalho consiste na abordagem do problema de identificação e representação das não-linearidades rígidas de saturação, zona morta, saturação com zona morta, liga-desliga, liga-desliga com histerese e liga-desliga com zona morta, através da utilização das redes neurais artificiais. Para tanto, explora-se a capacidade da arquitetura MLP em diferentes simulações, a partir de uma configuração padronizada quanto ao número de neurônios da camada escondida e o tamanho do conjunto de treinamento, conforme o teorema de Kolmogorov e a regra de Baum-Haussler, respectivamente. Os demais parâmetros, como número de épocas, taxa de aprendizagem e constante de momento, são ajustados empiricamente, de modo a proporcionar convergência rápida e curva de aprendizagem com oscilação mínima. Os experimentos são simulados a partir do ambiente de programação do MATLAB 7.5, cujos resultados, evidenciados a partir da análise do erro quadrático médio obtido na fase de generalização da rede, são apresentados para comprovar a eficiência do algoritmo proposto em identificar as não-linearidades rígidas supracitadas.

Palavras chave: Rede neural artificial, sistema não-linear, não-linearidade rígida.

Abstract of dissertation submitted to PPTEC/CEFET-RJ as partial fulfillment of the requirements for the degree of Master in Technology (M.T.).

## APPLICATION OF ARTIFICIAL NEURAL NETWORKS IN THE IDENTIFICATION OF HARD NONLINEARITIES

Wendel Furtado da Silva

June / 2008

Advisor: Paulo Lúcio Silva de Aquino, D.Sc.

Program: PPTEC

This work consists of the boarding of the problem of identification and representation of hard nonlinearities of saturation, dead zone, saturation with dead zone, on-off, on-off with hysteresis and on-off with dead zone, through the use of artificial neural networks. In order that, it explores the capacity of architecture MLP in different simulations, from a standardized configuration that much to the number of neurons in the hidden layer and the size of the set of training, both in the Kolmogorov theorem and the Baum-Haussler rule. The others parameters, as epochs, learning rate and constant moment, are adjusted empirically, in order to provide fast convergence and wave of learning with minimum oscillation. The experiments are simulated from the environment of programming of the MATLAB 7.5, which resulted, evidenced from the analysis of the average quadratic error in the phase of generalization of the artificial neural network, they are presented to prove the efficiency of the algorithm considered in identifying the mentioned hard nonlinearities.

Keywords: Artificial neural network, nonlinear system, hard nonlinearity.

## Lista de Figuras

	Pág.
Figura I.1 – Modelo de um neurônio biológico [26].	9
Figura I.2 – Interligação de neurônios [13].	10
Figura I.3 – Modelo do <i>perceptron</i> simples.	11
Figura I.4 – Funções de ativação.	14
Figura I.5 – Modelo de rede estática com camada única de neurônios.	15
Figura I.6 – Modelo de rede estática com múltiplas camadas de neurônios.	16
Figura I.7 – Modelo de Hopfield.	18
Figura I.8 – Treinamento supervisionado.	19
Figura I.9 – Treinamento não-supervisionado.	20
Figura I.10 – Treinamento por reforço.	21
Figura I.11 – Grafo do fluxo de sinal da saída do neurônio j [26].	22
Figura I.12 – Grafo do fluxo de sinal do neurônio k conectado ao neurônio j [26].	23
Figura I.13 – Atuação do termo de momento [27].	32
Figura I.14 – Característica entrada-saída para saturação [6].	38
Figura I.15 – Característica entrada-saída para zona morta [6].	39
Figura I.16 – Característica entrada-saída para saturação com zona morta.	40
Figura I.17 – Característica entrada-saída para liga-desliga [6].	41
Figura I.18 – Característica entrada-saída para liga-desliga com histerese [5].	42
Figura I-19 – Característica entrada-saída para liga-desliga com zona morta.	43
Figura II.1 – Gráfico da não-linearidade de saturação desejada.	48
Figura II.2 – Gráfico da não-linearidade de zona morta tipo I desejada.	49
Figura II.3 – Gráfico da não-linearidade de zona morta tipo II desejada.	50
Figura II.4 – Gráfico da não-linearidade de saturação com zona morta desejada.	51
Figura II.5 – Gráfico da não-linearidade de liga-desliga desejado.	52
Figura II.6 – Gráfico da não-linearidade de liga-desliga com histerese desejado.	53
Figura II.7 – Gráfico da não-linearidade de liga-desliga com zona morta desejado.	54

Figura II.8 – Tela do MATLAB (janela de comandos).	56
Figura III.1– EQMs obtidos para a simulação N <sup>o</sup> 6 de saturação.	62
Figura III.2 – Amostra comparativa dos sinais desejados e emitidos pela RNA, referentes à fase de generalização da simulação N <sup>o</sup> 6 de saturação.	63
Figura III.3– EQMs obtidos para a simulação N <sup>o</sup> 2 de zona morta tipo I.	65
Figura III.4 – Amostra comparativa dos sinais desejados e emitidos pela RNA, referentes à fase de generalização da simulação N <sup>o</sup> 2 de zona morta tipo I.	66
Figura III.5– EQMs obtidos para a simulação N <sup>o</sup> 3 de zona morta tipo II.	68
Figura III.6 – Amostra comparativa dos sinais desejados e emitidos pela RNA, referentes à fase de generalização da simulação N <sup>o</sup> 3 de zona morta tipo II.	69
Figura III.7– EQMs obtidos para a simulação N <sup>o</sup> 2 de saturação com zona morta.	71
Figura III.8 – Amostra comparativa dos sinais desejados e emitidos pela RNA, referentes à fase de generalização da simulação N <sup>o</sup> 2 de saturação com zona morta.	72
Figura III.9 – EQMs obtidos para a simulação N <sup>o</sup> 2 de liga-desliga.	74
Figura III.10 – Amostra comparativa dos sinais desejados e emitidos pela RNA, referentes à fase de generalização da simulação N <sup>o</sup> 2 de saturação com zona morta.	75
Figura III.11 – EQMs obtidos para a simulação N <sup>o</sup> 3 de liga-desliga com histerese.	77
Figura III.12 – Amostra comparativa dos sinais desejados e emitidos pela RNA, referentes à fase de generalização da simulação N <sup>o</sup> 3 de liga-desliga com histerese.	78
Figura III.13 – EQMs obtidos para a simulação N <sup>o</sup> 2 de liga-desliga com zona morta.	80
Figura III.14 – Amostra comparativa dos sinais desejados e emitidos pela RNA, referentes à fase de generalização da simulação N <sup>o</sup> 2 de liga-desliga com zona morta.	81
Figura III.15 – EQMs obtidos para a simulação N <sup>o</sup> 7 de zona morta tipo I.	83
Figura III.16 – Amostra comparativa dos sinais desejados e emitidos pela RNA, referentes à fase de generalização da simulação N <sup>o</sup> 7 de zona morta tipo II.	84
Figura III.17 – EQMs obtidos para a simulação N <sup>o</sup> 7 de zona morta tipo II.	86
Figura III.18 – Amostra comparativa dos sinais desejados e emitidos pela RNA, referentes à fase de generalização da simulação N <sup>o</sup> 7 de zona morta tipo II.	87
Figura III.19 – EQMs obtidos para a simulação N <sup>o</sup> 7 de liga-desliga.	89
Figura III.20 – EQMs obtidos para a simulação N <sup>o</sup> 8 de liga-desliga.	89

Figura III.21 – Amostra comparativa dos sinais desejados e emitidos pela RNA, referentes à fase de generalização da simulação N <sup>o</sup> 8 de liga-desliga.	90
Figura III.22 – EQMs obtidos para a simulação N <sup>o</sup> 7 de liga-desliga com histerese.	92
Figura III.23 – EQMs obtidos para a simulação N <sup>o</sup> 8 de liga-desliga com histerese.	92
Figura III.24 – Amostra comparativa dos sinais desejados e emitidos pela RNA, referentes à fase de generalização da simulação N <sup>o</sup> 8 de liga-desliga com histerese.	93
Figura III.25 – EQMs obtidos para a simulação N <sup>o</sup> 7 de liga-desliga com zona morta.	95
Figura III.26 – Amostra comparativa dos sinais desejados e emitidos pela RNA, referentes à fase de generalização da simulação N <sup>o</sup> 7 de liga-desliga com zona morta.	96

## Lista de Tabelas

	Pág.
Tabela II.1 – Configuração padrão da RNA.	57
Tabela III.1 – Resultados obtidos para a simulação de saturação.	61
Tabela III.2 – Resultados obtidos para a simulação de zona morta tipo I.	64
Tabela III.3 – Resultados obtidos para a simulação de zona morta tipo II.	67
Tabela III.4 – Resultados obtidos para a simulação de saturação com zona morta.	70
Tabela III.5 – Resultados obtidos para a simulação de liga-desliga.	73
Tabela III.6 – Resultados obtidos para a simulação de liga-desliga com histerese.	76
Tabela III.7 – Resultados obtidos para a simulação de liga-desliga com zona morta.	79
Tabela III.8 – Resultados obtidos para a simulações complementares de zona morta tipo I.	82
Tabela III.9 – Resultados obtidos para a simulação complementar de zona morta tipo II.	85
Tabela III.10 – Resultados obtidos para as simulações complementares de liga-desliga.	88
Tabela III.11 – Resultados obtidos para a simulações complementares de liga-desliga com histerese.	91
Tabela III.12 – Resultados obtidos para a simulação complementar de liga-desliga com zona morta.	94



## Abreviaturas e Símbolos

Abreviatura / Símbolo	Significado	Unidade
ADALINE	ADaptative LINear Combiner Element	-
CEFET	Centro Federal de Educação Tecnológica Celso Suckow da Fonseca	-
DIPPG	Diretoria de Pesquisa e Pós-Graduação	-
EQM	Erro Quadrático Médio	-
GB	Medida de informação binária	gigabyte
HD	Hard Disk	-
IEEE	Institute of Electrical and Electronics Engineers	-
INNS	International Neural Networks Society	-
MATLAB	MATrix LABoratory	-
MCP	McCulloch-Pitts	-
MLP	Multi Layer Perceptron	-
PC-AT	Personal Computer – Advanced Technology	-
PID	Proporcional-Integral-Derivativo	-
PPTEC	Programa de Pós-Graduação em Tecnologia	-
RAM	Random Access Memory	-
RJ	Rio de Janeiro	-
RNA	Rede Neural Artificial	-
SP2	Service Pack 2	-

## INTRODUÇÃO

O presente trabalho segue a linha de pesquisa do Laboratório de Controle e Automação do Programa de Engenharia Elétrica do CEFET/RJ, onde se trabalha na modelagem e aplicação de controle e automação de processos e especificamente, faz parte do projeto no qual se estuda o comportamento de sistemas de controle que tenham relações não-lineares entre suas entradas e saídas.

Dentro deste contexto, já foram desenvolvidos diversos estudos [1-7], os quais demonstram que não-linearidades, inerentes ou intencionais, merecem ser devidamente tratadas, a fim de garantir estabilidade e desempenho satisfatórios em toda a faixa de operação do sistema de controle.

A necessidade de desenvolver sistemas capazes de operar mesmo diante de elevado número de variáveis interagindo entre si, processos inerentemente não-lineares, existência de incertezas nas medições e freqüentes perturbações, motiva a ampliação de pesquisas que venham trazer contribuições para a determinação de metodologias de controle e automação eficazes [7, 8]. Entre os diversos estudos feitos na área, nota-se que as técnicas de controle convencional são as mais empregadas, estabelecidas por meio de modelos linearizados de sistemas físicos. No entanto, são lineares apenas em pequenos intervalos de operação e quando aplicadas em sistemas que extrapolam seus limites, o controlador linear torna-se passível de desempenho insuficiente ou instável [1].

É relevante frisar que sistemas físicos do mundo real são inerentemente não-lineares e, de certa forma, requerem sistemas de controle também não-lineares [1]. Estas não-linearidades podem causar comportamentos indesejados, caso não sejam devidamente compensadas, ou até mesmo intencionalmente introduzidas, a fim de aperfeiçoar o desempenho do controlador e/ou simplificar sua construção. Porém, a natureza descontínua dos efeitos não-lineares implica que os mesmos não são representados adequadamente por aproximações lineares e, por outro lado, as características de resposta de sistemas não-lineares dependem, normalmente, dos valores de entrada, de modo que não existem métodos

gerais de análise de sistemas não-lineares [4]. Neste aspecto, quando há presença de sistemas não-lineares onde, por exemplo, a determinação de um modelo já possui um alto grau de complexidade (elevado número de parâmetros, valores incertos, desconhecimento acerca dos fenômenos físico-químicos envolvidos etc.) e o formalismo para análise de desempenho, otimização e robustez não são suficientes para determinar uma lei de controle aceitável, a busca de soluções caminha na direção de sistemas heurísticos, baseados em regras de inferência, decisões baseadas em lógica nebulosa, redes neurais artificiais etc. [9]

## **Motivação**

As redes neurais artificiais representam um novo paradigma metodológico no campo da inteligência artificial [10]. São modelos matemático-computacionais inspirados no mecanismo de funcionamento do neurônio biológico e de interligação entre eles, que têm capacidade de adquirir conhecimento através da experiência e torná-lo disponível para uso [11]. Por serem ferramentas computacionais, podem ser implementadas através de *hardware* ou *software* [12].

A técnica de redes neurais artificiais tem sido proposta como um método alternativo para a solução de problemas que envolvem relações não-lineares, sistemas complexos, classificação e reconhecimento de padrões, nos mais diversos campos de atuação [13]. Especificamente, a engenharia de controle e automação de processos tende a se beneficiar bastante com o emprego das redes neurais artificiais, principalmente em aplicações como supervisão de sistemas, controle de plantas não-lineares e controle adaptativo de plantas não-lineares, devido ao potencial que possui para aproximação de mapeamentos não-lineares estáticos e dinâmicos em que pesem não-linearidades pouco conhecidas ou mesmo desconhecidas [14].

Apesar do notável potencial para solucionar problemas de identificação e controle de sistemas que envolvam não-linearidades, as redes neurais apresentam algumas questões quanto à determinação da estrutura necessária para representar um determinado sistema, técnicas de treinamento e dinâmicas temporais envolvidas no processamento [15].

Por oportuno, desde o ano 2000, alunos de mestrado do PPTEC/CEFET/RJ vêm desenvolvendo pesquisas que incluem a utilização de redes neurais artificiais em sistemas de controle [16] e bioengenharia [17].

### **Objetivo geral**

Este trabalho tem por objetivo identificar e representar não-linearidades rígidas de saturação, zona morta, saturação com zona morta, liga-desliga, liga-desliga com histerese e liga-desliga com zona morta, através de redes neurais artificiais tipo MLP com treinamento supervisionado por retropropagação de erro.

### **Objetivos específicos**

Para atingir o objetivo geral, o presente trabalho foi dividido em etapas que foram seguidas, constituindo os objetivos específicos, assim descritos:

- Aplicar o algoritmo da rede neural artificial MLP com treinamento supervisionado por retropropagação de erro na identificação das não-linearidades rígidas mencionadas no objetivo geral;
- Analisar comparativamente o EQM obtido pelas simulações propostas;
- Analisar a curva de evolução do EQM de treinamento;
- Avaliar a capacidade de rede neural artificial em identificar não-linearidades rígidas, conforme o erro quadrático médio obtido na fase de generalização.

### **Demais contribuições**

A presente dissertação presta-se, ainda, a contribuir com as pesquisas do Laboratório de Controle e Automação do Programa de Engenharia Elétrica do CEFET/RJ, no que pesa o desenvolvimento do sistema de controle do carro elétrico [18], para o qual o uso das redes neurais artificiais tem sido cogitado.

## **Organização da dissertação**

A estrutura desta dissertação compreende cinco partes. A primeira consiste nesta introdução, onde são apresentados alguns conceitos gerais e o objetivo deste trabalho.

No capítulo I são apresentados os conceitos fundamentais mais relevantes que serviram de orientação para este trabalho.

Assuntos relativos a materiais e métodos utilizados no desenvolvimento encontram-se no capítulo II. Neste apresentam-se as etapas de funcionamento da rede neural e a metodologia utilizada para definir parâmetros e validar os resultados das simulações realizadas.

A seguir, o capítulo III apresenta os resultados obtidos pela rede neural nas diferentes configurações propostas.

Por fim, no capítulo IV são apresentadas as discussões dos resultados, conclusão e sugestões de continuidade.

Outrossim, os códigos MATLAB utilizados na geração do conjunto de treinamento, simulação da RNA e obtenção de gráficos, estão disponíveis na forma de apêndices, conforme a seguir:

- Apêndices 1 a 7: geração de pares entrada-saída das não-linearidades estudadas;
- Apêndice 8: implementação da rede neural artificial;
- Apêndice 11: plotagem dos EQMs de treinamento/teste com indicação do EQM de generalização;
- Apêndice 12: comparação entre sinais desejados e emitidos pela RNA;
- Apêndices 13 e 14: evolução dos pesos e bias, conforme determinada topologia.

## CAPÍTULO I

### REVISÃO BIBLIOGRÁFICA

#### I.1– APLICAÇÃO DE REDES NEURAS ARTIFICIAIS EM SISTEMAS NÃO-LINEARES

O controle de sistemas dinâmicos não-lineares com redes neurais foi proposto primeiramente no final da década de 80. Inicialmente não havia nenhum critério teórico consagrado, sendo um método puramente experimental [19]. Porém, nos últimos vinte anos houve intenso interesse de pesquisadores e projetistas em desenvolver e aplicar metodologias de controle não-linear baseados em redes neurais artificiais [14, 20], que contribuíram sobremaneira para formação de bases teóricas amplamente disponíveis e que vem sendo continuamente discutidas. Entre os principais trabalhos, podem ser citados os de Hunt *et al* [3], Narendra [20], Cybenko [21, 22], Teng [23], Miller *et al* [24], incluindo os de pesquisadores nacionais como Cardenuto [9], Real [14], Comerlato [15] e Zuben [25], por exemplo, onde descrevem o potencial das redes neurais no aprendizado de mapeamentos estáticos e dinâmicos, sugerindo sua utilização na identificação e controle de sistemas dinâmicos não-lineares, especialmente quando não se conhece ou se conhece pouco a não-linearidade envolvida.

Basicamente, um sistema de controle deve influenciar o comportamento dinâmico de um processo para que as variáveis de saída se mantenham nos valores desejados ou para que estas sigam determinadas trajetórias. Neste sentido, as redes neurais artificiais possuem características que lhes permitem ser inseridas em projetos de controle de sistemas aptos a garantir estabilidade e desempenho satisfatório em toda a faixa operacional, mesmo diante de incertezas ou possíveis perturbações da realidade prática [7, 25].

## I.2– REDES NEURAIIS ARTIFICIAIS

Nos últimos anos, a técnica de redes neurais artificiais tem sido proposta como um método alternativo para a solução de processos de controle que tenham elevado número de variáveis interagindo entre si, não-linearidades e incertezas nas medições e parâmetros internos da planta [7].

As redes neurais artificiais correspondem a sistemas adaptativos ou sistemas de processamento paralelos distribuídos e são fundamentadas pelas redes neurais biológicas, que consistem de um elevado número de processadores simples intensamente interligados. A compreensão do mecanismo de funcionamento e comunicação das células nervosas, também chamadas de neurônios, tem sido a chave para o desenvolvimento de modelos matemáticos para testes e realização de aplicações práticas [13]. Tais estruturas de processamento têm a capacidade de criar suas próprias representações para "aprender", "memorizar" e "lembrar" [26].

Existem diversos tipos de redes neurais artificiais, mas todos apresentam pelo menos três características em comum: existência de neurônios (unidades de processamento); determinada interligação entre eles (topologia da rede) e um algoritmo de treinamento [27]. As próximas seções apresentam uma breve introdução a cada um destes tópicos, sendo que mais detalhes podem ser obtidos em [26 - 28], entre outros.

### I.2.1– Breve histórico

Em 1943, o neurofisiologista Warren S. McCulloch e o matemático Walter H. Pitts publicam um artigo que interpreta o funcionamento do neurônio natural como sendo um circuito binário. O modelo, chamado MCP (McCulloch e Pitts), tem dois estados, ativo ou inativo, conforme a saída dada pela função da soma ponderada das respectivas entradas [29].

A aprendizagem torna-se objeto de estudo apenas em 1949, quando Donald Hebb apresenta a formulação de uma teoria neurofisiológica para modificação de sinapses em neurônios reais [30]. A “regra de Hebb” ainda é atualmente empregada em diversos algoritmos de aprendizagem [28].

Em 1958, Frank Rosenblatt, apoiado nos estudos de McCulloch e Pitts, apresenta o modelo *Perceptron*, no qual os neurônios são organizados em três camadas e um algoritmo de treinamento é aplicado para ajustar as conexões situadas entre as duas primeiras [31].

Bernard Widrow e Marcian Hoff desenvolvem em 1960, o modelo *Adaline (ADaptive LINear combiner Element)* [32]. O modelo também é inspirado nos trabalhos de McCulloch, mas possui saídas analógicas e utiliza um algoritmo de aprendizado baseado no cálculo do erro entre a soma dos pesos das entradas e a saída desejada. A regra de aprendizagem ora proposta é conhecida como regra de Widrow-Hoff ou regra delta [28].

Em 1969, Minsky e Papert demonstram a limitação do modelo proposto por Rosenblatt em resolver problemas de associação de padrões que não sejam linearmente separáveis [33]. Esta observação provoca a paralisação do assunto até o início da década de 1980, restando apenas alguns poucos pesquisadores em atividade [27].

A partir de 1982, recomeça o interesse por redes neurais artificiais com o artigo “*Neural Networks and Physical Systems and Emergent Collective Properties*”, de John Hopfield [34], que ao tratar do princípio físico sobre armazenamento de informação em configurações dinamicamente estáveis, atrai o interesse de matemáticos e tecnólogos. O trabalho inclui um conjunto de neurônios binários, dispostos de forma que as entradas sejam realimentadas pelas saídas, caracterizando um dos primeiros modelos de redes neurais dinâmicas [26, 27].



Em 1986, Rumelhart e McClelland descrevem uma regra de aprendizagem para redes estáticas multicamadas, denominada *backpropagation*, a qual permite utilizar o *Perceptron* multicamadas para resolver problemas com padrões não linearmente separáveis [35].

Somente a partir de meados dos anos 1980 renasce o interesse pelas redes neurais artificiais na comunidade internacional, fomentada pelo aumento da capacidade computacional mais o fato de a inteligência artificial simbólica não ter apresentado avanços significativos na resolução de problemas considerados simples para um ser humano [27].

Em 1987 ocorre em São Francisco a primeira conferência de redes neurais em tempos modernos, a “*IEEE International Conference on Neural Networks*”, sendo formada a “*INNS - International Neural Networks Society*”. A partir destes acontecimentos acontece a fundação do *INNS Journal* em 1989, seguido do “*Neural Computation*” e do “*IEEE Transactions on Neural Networks*”, em 1990 [27].

### I.2.2– Neurônio biológico

Estruturalmente, o cérebro é constituído por células nervosas a que se chamam neurônios. Nos trabalhos pioneiros de Ramón y Cajál, citado por [13], são identificadas, descritas e apresentadas as definições destes neurônios.

O cérebro humano é constituído por cerca de cem bilhões de neurônios dos mais diversos tipos. Dentre estes, o neurônio motor, representado na Figura I.1, é o mais estudado e melhor entendido dos neurônios conhecidos. Sua estrutura é formada por um componente básico conhecido como corpo celular ou soma, que está ligado a uma única fibra nervosa ou axônio eferente. Em determinadas condições, sob estímulo, o soma produz o pulso elétrico que é conduzido pelo axônio eletricamente ativo. Esse pulso, conhecido como impulso nervoso ou potencial de ação, tem amplitude fixa e viaja ao longo do axônio a uma velocidade da ordem de

vinte milissegundos, durando cada pulso aproximadamente um milissegundo. Após ter sido disparado, o neurônio passa por um período de repouso que dura, no mínimo, dez milissegundos, até que seja disparado novamente.

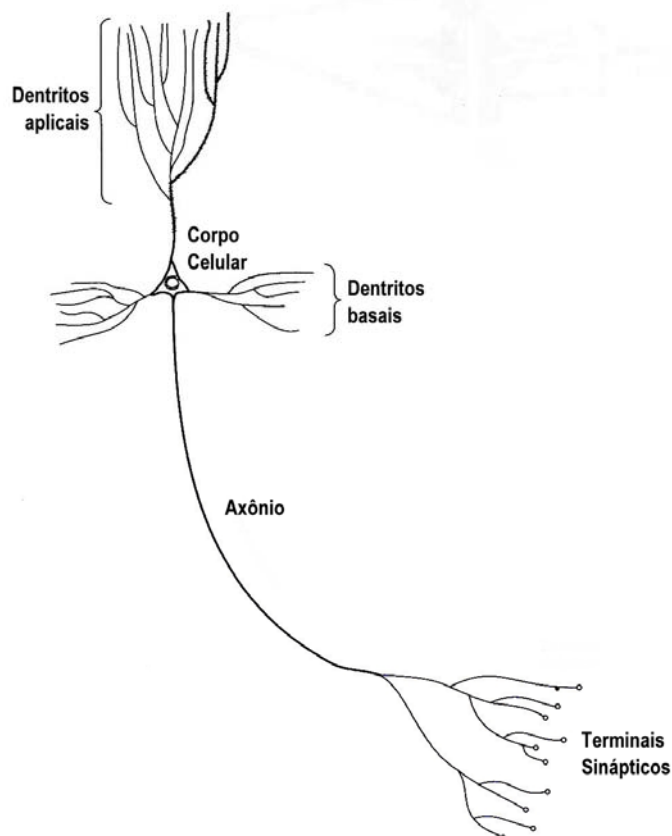


Figura I.1 – Modelo de um neurônio biológico [27].

As ramificações encontradas no extremo do axônio são chamadas subaxônios. Essas estruturas distais são responsáveis por conduzir o pulso elétrico aos dentritos dos neurônios vizinhos através de contatos especiais, ilustrados na Figura I.2. Esses contatos especiais são denominados sinapses e são mediados por agentes químicos conhecidos como neurotransmissores, presentes nas regiões intersinápticas, que asseguram a transmissão dos influxos nervosos entre os dentritos. São várias as substâncias que funcionam como agente

neurotransmissor (aproximadamente cem diferentes substâncias) e, dependendo do tipo de mediador, a sinapse pode ser excitante (positiva) ou inibidora (negativa).

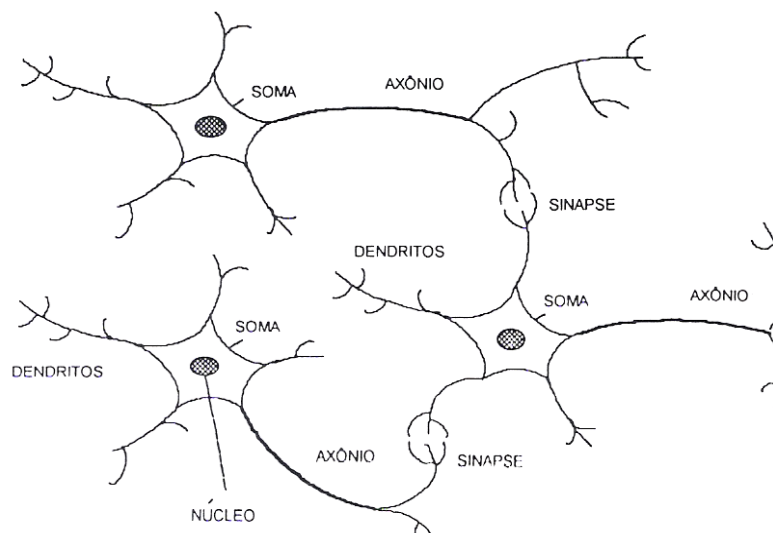


Figura 1.2 – Interligação de neurônios [13].

Basicamente, a transmissão de um pulso por um neurônio pós-sináptico depende do somatório das múltiplas ativações excitatórias ou inibitórias recebidas dos neurônios pré-sinápticos. Quando o somatório dos sinais ultrapassa um determinado valor, o neurônio dispara um sinal digital processado no soma, que é novamente transmitido a outro neurônio através do axônio eferente, prosseguindo o processo de forma repetida e sucessiva. Logo, para um dado estímulo (entrada), o cérebro reagirá com uma determinada resposta (saída), através do processamento de sinais em suas redes neurais.

A partir da elucidação dos princípios básicos de funcionamento dos neurônios motores e da rede neural biológica por eles composta, torna-se possível o desenvolvimento de modelos artificiais para fins específicos, assim as chamadas redes neurais artificiais.

### I.2.3– Neurônio artificial

As redes neurais artificiais são fundamentadas nos neurônios artificiais ou elementos de processamento [12]. Neles são desenvolvidas as seqüências de processos que definem a resposta da unidade, que são as ponderações sinápticas dos sinais de entrada, o somatório dos sinais ponderados e o cálculo da função de ativação.

O neurônio artificial tem origem no modelo MCP, desenvolvido em 1943, pelos pesquisadores McCulloch e Pitts [29], que se dedicaram à modelagem da capacidade computacional do neurônio biológico. O modelo MCP substitui os dendritos por entradas, e as conexões destas com o corpo celular artificial são executadas através de pesos. Assim, a sinapse que transmite um sinal forte é representada pela associação de um peso de maior valor a sua entrada e os estímulos captados pelas entradas são processados em uma função soma, sendo o resultado desta transmitido a uma função de ativação, responsável pela produção da saída. Entretanto, este modelo não contempla técnicas de aprendizagem.

Em 1958, surge o modelo proposto por Frank Rosenblatt [31], denominado *Perceptron*, o qual é composto por uma estrutura de rede com neurônios MCP e um algoritmo de treinamento.

A Figura I.3 demonstra o modelo *Perceptron* para uma única saída.

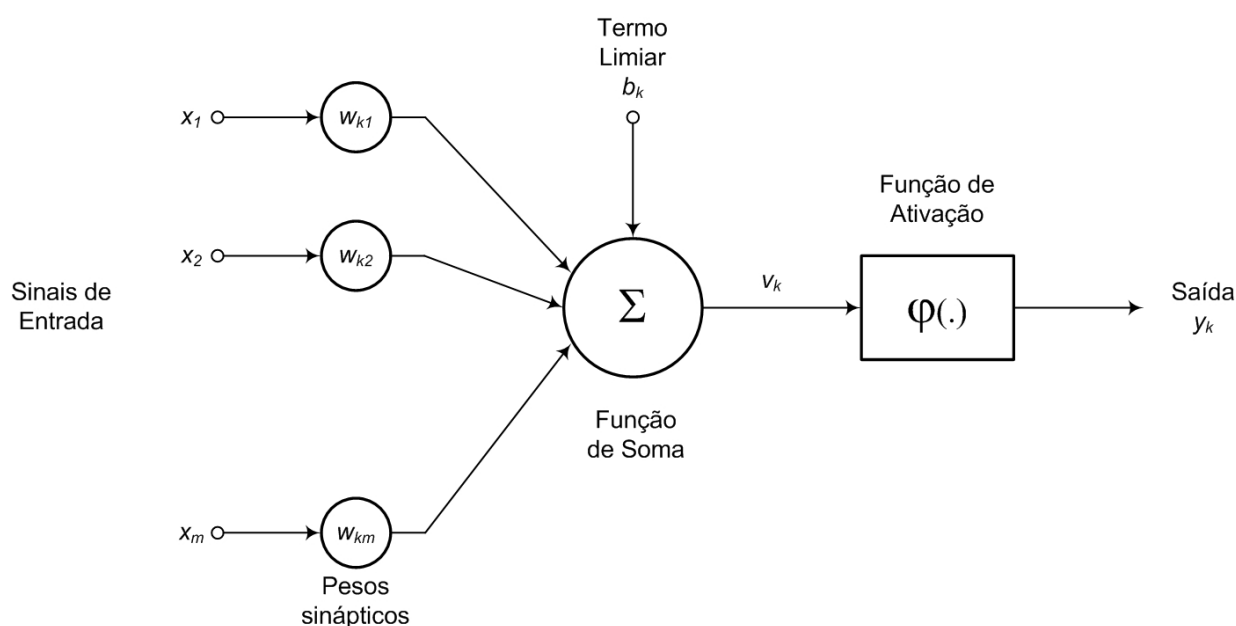


Figura I.3 – Modelo do *perceptron* simples.

O modelo neural apresentado também inclui o termo limiar, também conhecido por *bias* e representado por  $b_k$ . Este termo possui a função de causar um acréscimo ou decréscimo na entrada da função de transferência  $v_k$  [27].

Desta forma, para o neurônio  $k$ , tem-se:

$$v_k = \sum_{j=1}^N w_{kj} x_j + b_k, \quad \text{eq. 1.1}$$

$$y_k = \varphi(v_k), \quad \text{eq. 1.2}$$

onde,

- $x_1, x_2, \dots, x_n$  – Sinais de entrada;
- $w_{1k}, w_{2k}, \dots, w_{mk}$  – Pesos sinápticos do neurônio  $k$ ;
- $b_k$  – Termo limiar ou *bias*;
- $\varphi(v_k)$  – Função de ativação;
- $v_k$  – Entrada da função de ativação.

O *bias* pode ser incluído no somatório da eq. 1.1 como uma entrada adicional, caracterizada por  $x_0 = +1$  junto com o peso sináptico a ela associado  $w_{k0} = b_k$ . Desta forma, a eq. 1.1 pode ser reescrita da seguinte forma:

$$v_k = \sum_{j=0}^N w_{kj} x_j \quad \text{eq. 1.3}$$

A função de ativação (eq. 1.2), denotada por  $\varphi$ , é responsável por fornecer a saída do neurônio. Para o modelo *Perceptron* simples, a função de ativação utilizada é a função degrau unitário (eq. 1.4).

$$\varphi(v_k) = \begin{cases} y_k = 1 & \text{se } v_k \geq 0 \\ y_k = 0 & \text{se } v_k < 0 \end{cases} \quad \text{eq. 1.4}$$

### I.2.4– Funções de ativação

A função de ativação determina o valor de resposta de um neurônio e possui duas características distintas: uma faixa onde ocorre uma contínua ascensão com a elevação do argumento (faixa dinâmica) e a saturação fora desta faixa. Qualquer um dos tipos de função seguintes, entre outros, pode ser representativo da função de ativação: (a) função linear; (b) função rampa (idêntica a linear, exceto que considera a saturação); (c) função degrau; (d) função tangente hiperbólica; e (e) função sigmóide (função sigmoidal ou logística) [20 - 22], conforme ilustradas na Figura I.4. As seguintes relações matemáticas caracterizam estas funções de ativação:

Função linear:

$$\varphi(v) = v \quad \text{eq. I.5}$$

Função rampa:

$$\varphi(v) = \begin{cases} 1 & \text{se } v \geq +\frac{1}{2} \\ v & \text{se } -\frac{1}{2} < v < +\frac{1}{2} \\ 0 & \text{se } v \leq -\frac{1}{2} \end{cases} \quad \text{eq. I.6}$$

Função degrau:

$$\varphi(v) = \begin{cases} 1 & \text{se } v \geq 0 \\ 0 & \text{se } v < 0 \end{cases} \quad \text{eq. I.7}$$

Função sigmóide:

$$\varphi(v) = \frac{1}{1 + \exp(-av)} \quad \text{eq. I.8}$$

Função tangente hiperbólica:

$$\varphi(v) = \frac{1 - \exp(-av)}{1 + \exp(-av)} \quad \text{eq. I.9}$$

Nas eq. I.8 e I.9, o parâmetro  $a$  define a inclinação da função ou taxa de ascensão.

As funções de ativação sigmóide e tangente hiperbólica são as mais utilizadas na atualidade, por apresentarem características de saturação, ascensão contínua na faixa dinâmica, simetrias e derivadas contínuas [27].

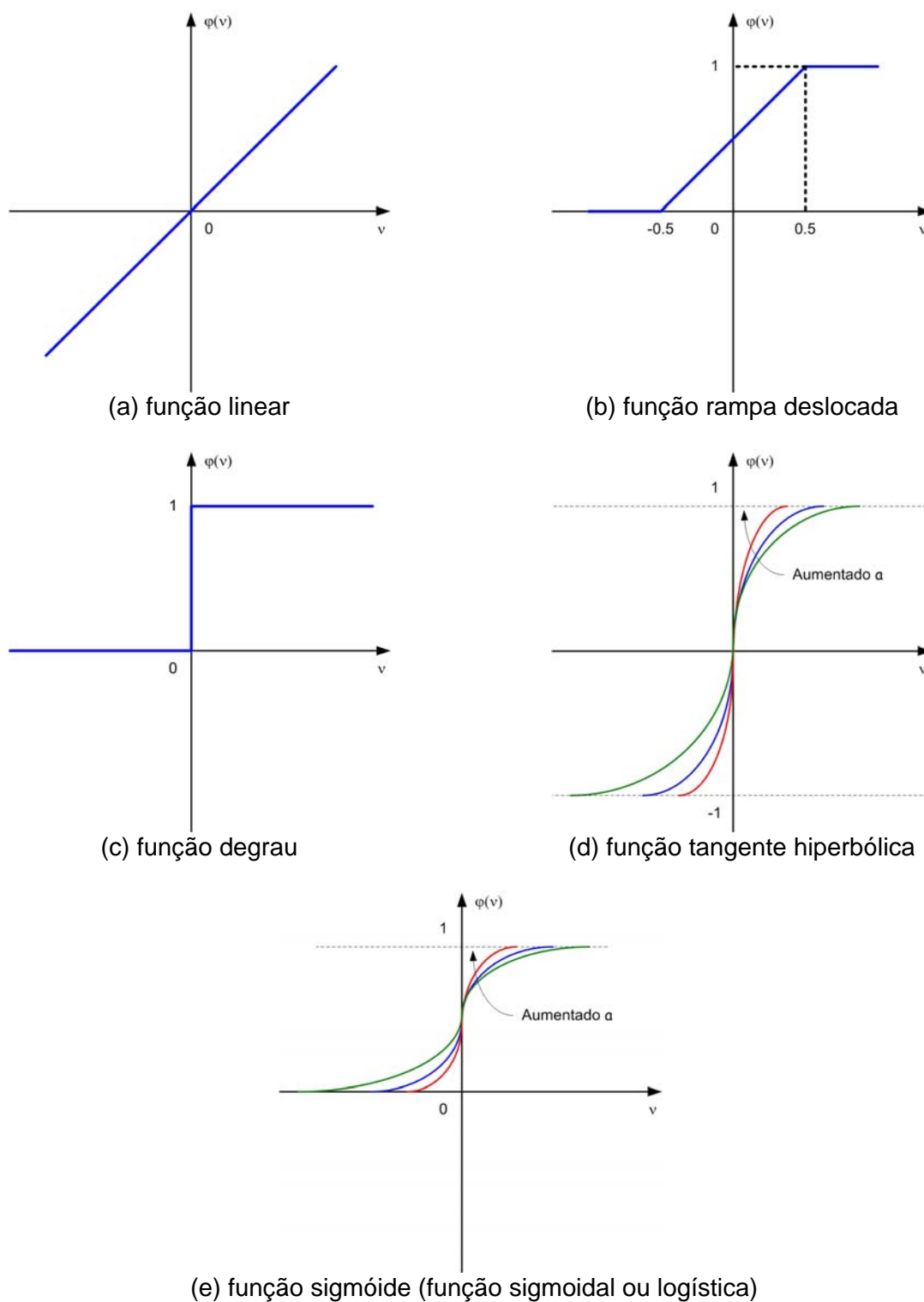


Figura I.4 – Funções de ativação.

## I.2.5– Principais arquiteturas de redes neurais artificiais

Uma rede neural pode ter os neurônios estruturados de diferentes formas e são classificadas, em geral, dentro de dois grupos principais: redes estáticas ou não recorrentes, e redes dinâmicas ou recorrentes [27, 28].

### I.2.5.1– Redes neurais estáticas

As redes neurais estáticas dependem apenas dos valores atuais das entradas, sendo divididas principalmente em:

#### – Redes de camada única

Os sinais são transmitidos apenas da entrada para a saída, constituindo-se na forma mais simples de estruturação dos neurônios em uma rede. A designação camada única refere-se à camada de saída, pois não se conta a camada de entrada, visto que ela não realiza qualquer computação [27]. Esta arquitetura é vista na Figura I.5.

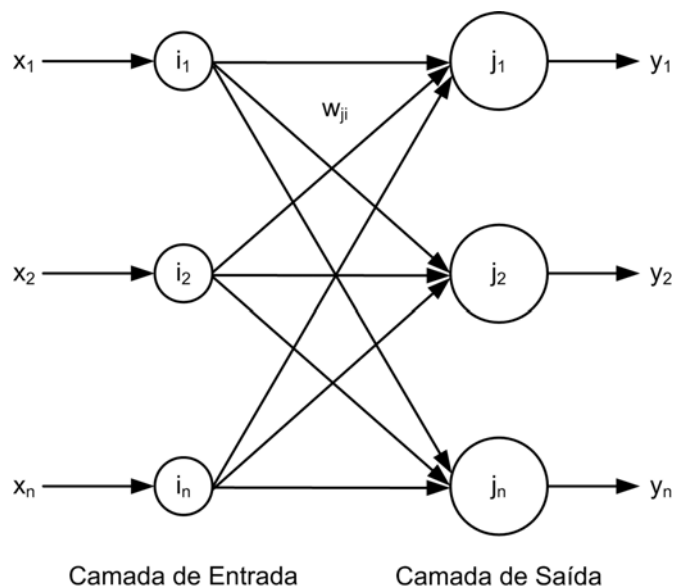


Figura I.5 – Modelo de rede estática com camada única de neurônios.



## – Redes multicamadas

As redes multicamadas ou MLP, abreviação para *MultiLayer Perceptron* (*Perceptron* de multicamadas) distinguem-se da anterior pela presença de uma ou várias camadas ocultas. A camada de entrada é responsável por receber os sinais de entrada, não exercendo nenhum processamento, por isso alguns autores nem a consideram uma camada de fato, mas apenas elementos receptores de dados, encarregada de transmiti-los à camada escondida [27]. A camada escondida realiza o primeiro processamento, que tem a capacidade de extrair informações estatísticas dos sinais de entrada  $x$  e, adiante, a camada de saída recebe os sinais da camada escondida para o último processamento, resultando na resposta global da rede, representada por  $y$ . A Figura I.6 mostra a arquitetura de uma rede neural MLP com uma camada escondida.

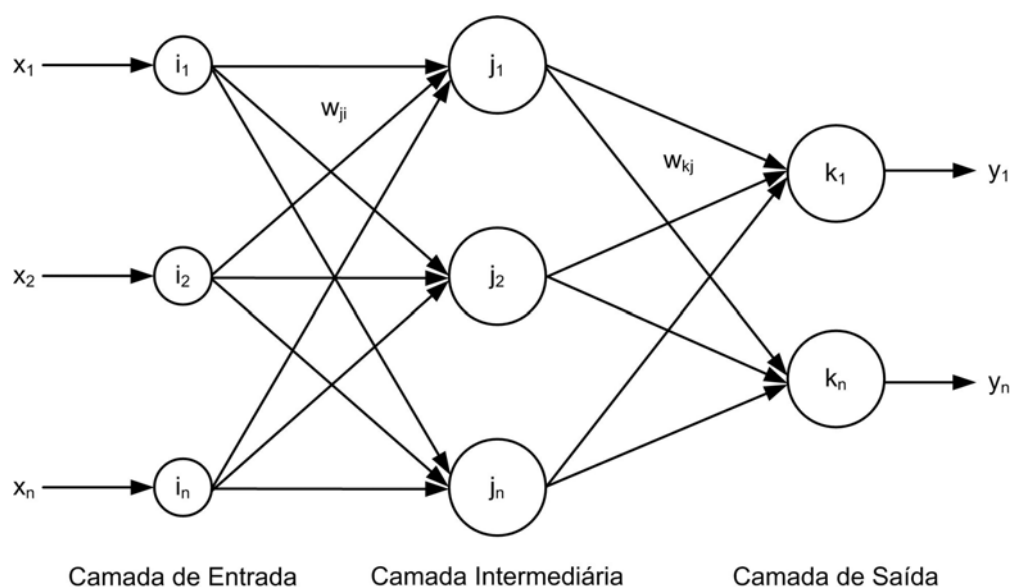


Figura I.6 – Modelo de rede estática com múltiplas camadas de neurônios.

Uma rede MLP é uma rede progressiva, o que significa dizer que as saídas dos neurônios de qualquer camada se conectam exclusivamente às entradas dos neurônios da

camada seguinte, sem haver laços de realimentação. Conseqüentemente, o sinal de entrada propaga-se em sentido progressivo através da rede.

A rede pode ser completamente conectada, caso em que cada nó em uma camada é conectado a todos os outros nós da camada adjacente. De forma alternativa, uma rede MLP pode ser parcialmente conectada, caso em que algumas sinapses podem estar faltando. Neste trabalho, é considerada apenas a rede MLP totalmente conectada.

Entre as camadas de neurônios estão as conexões que levam o valor de saída de um neurônio, multiplicando-o pelo seu peso, até a entrada de outro neurônio. Cada conexão possui um peso, representado por  $w$ , sendo a determinação do valor destes pesos o objetivo do aprendizado da rede.

Quanto ao número de camadas intermediárias, Cybenko [21] argumenta que basta apenas uma camada intermediária para aproximar qualquer mapeamento contínuo e duas camadas, no máximo, com um número suficiente de neurônios artificiais, para reproduzir quaisquer mapeamentos.

Especificamente, as redes neurais artificiais tipo MLP vem sendo amplamente utilizadas na representação de sistemas de controle não-linear, devido à característica de interligação entre seus neurônios, associada ao poder de discriminação não-linear [37].

#### I.2.5.2– Redes Recorrentes

A principal característica das redes recorrentes é a presença de pelo menos um laço de realimentação e podem ser classificadas como parcialmente recorrentes ou totalmente recorrentes. As totalmente recorrentes têm todas as conexões recorrentes e ajustáveis, ao contrário das parcialmente recorrentes, que têm somente uma parte das conexões recorrentes ou nem todas são ajustáveis. A presença de laços de realimentação tem profundo impacto na capacidade de aprendizagem da rede e no seu desempenho.

A rede de Hopfield [34] foi o primeiro modelo a ter recorrência, conforme apresentado na Figura I.7. Ele pode ser utilizado como memória associativa ou para resolver problemas de otimização. A arquitetura é formada por diversos neurônios distribuídos em camada única.

Nessa camada, os sinais de entrada  $x$  são captados pelos neurônios e o sinal de saída  $y$  correspondente é realimentado em todos os seus vizinhos.

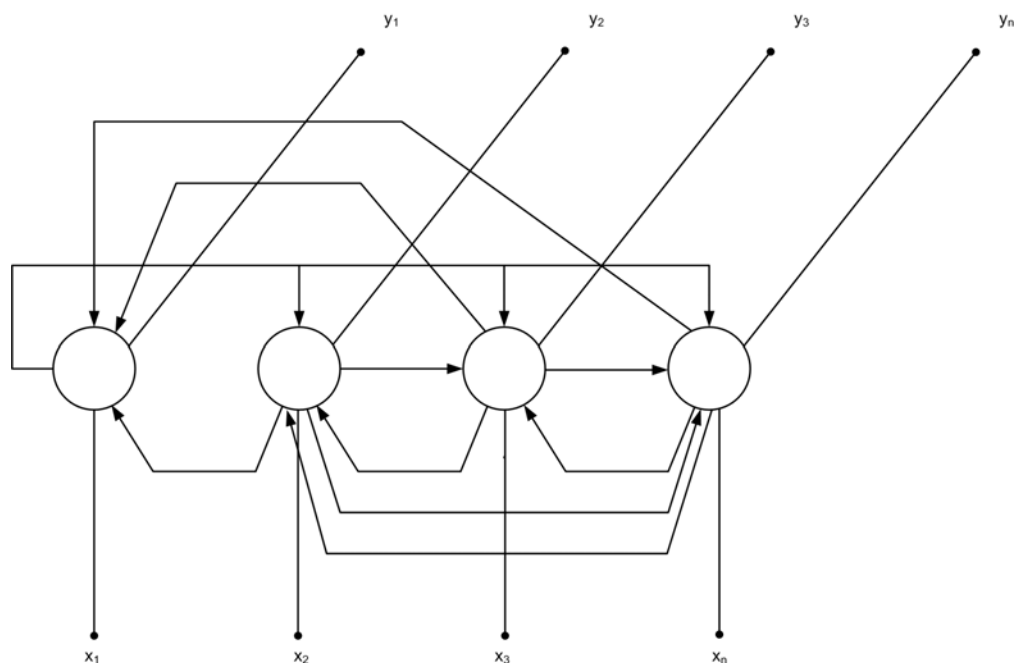


Figura I.7 – Modelo de Hopfield.

## I.2.6– Métodos de treinamento

O aprendizado ou treinamento de redes neurais tem como principal característica o comportamento adaptativo com melhora gradual de desempenho através de um processo contínuo de simulação do ambiente no qual a rede está inserida, onde se busca ajustar os parâmetros livres ou adaptáveis da rede [27]. Os métodos de treinamento dividem-se em: treinamento supervisionado, treinamento não-supervisionado e treinamento por reforço.

### I.2.6.1– Treinamento supervisionado

O treinamento supervisionado é caracterizado pela disponibilidade de um supervisor ou professor externo, cujo conhecimento sobre o sistema está representado por um conjunto de padrões entrada-saída, chamado conjunto de treinamento [27]. A partir deste conjunto, é

possível determinar um sinal de erro comparando-se a saída da rede neural com a saída correspondente do conjunto de treinamento, para uma dada entrada. O sinal de erro e os parâmetros de treinamento definem o ajuste dos pesos da rede, e o treinamento continua até que o erro estipulado pelo projetista seja alcançado, quando possível. O mecanismo de treinamento supervisionado é ilustrado na Figura I.8.

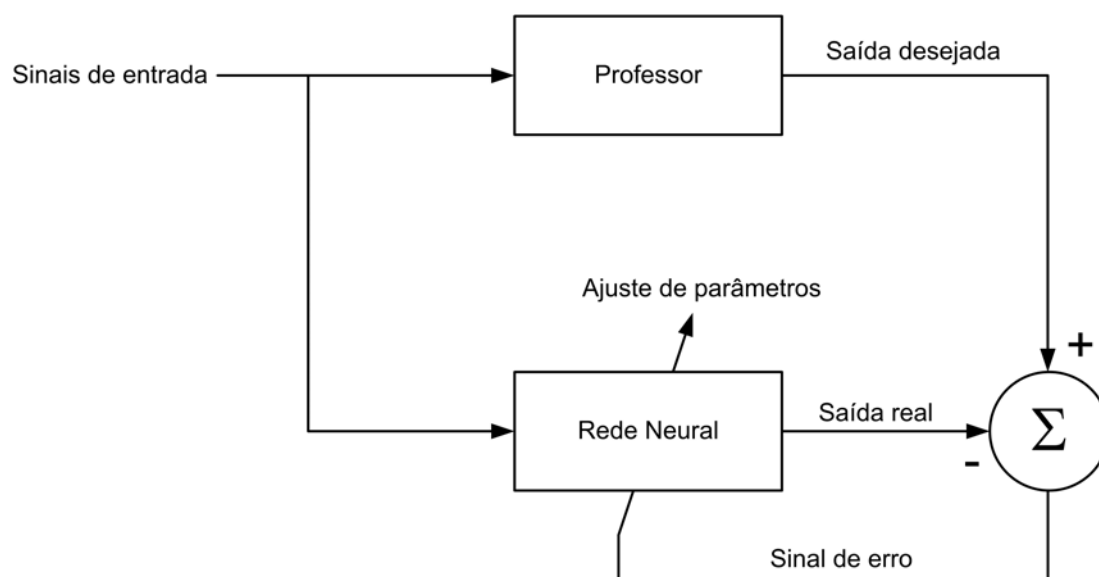


Figura I.8 – Treinamento supervisionado.

O treinamento supervisionado pode ser executado *off-line* ou *on-line*. No treinamento *off-line*, os padrões utilizados durante o treinamento são fixos e correspondem àqueles que são apresentados na fase de generalização da rede. A desvantagem deste modo reside no fato de a rede não ser capaz de aprender novas estratégias para situações particulares, caso estas não sejam consideradas na escolha do conjunto de treinamento. Por outro lado, se a rede deve adaptar-se continuamente por força da mudança constante do conjunto de dados, trata-se então do treinamento *on-line* [28].

De forma análoga àquela utilizada pelo ser humano e outros animais, é a partir de exemplos que uma rede neural adquire conhecimento ou informação relevante sobre um problema a ser solucionado quando se aplica um algoritmo de treinamento supervisionado [25].

### I.2.6.2– Treinamento não-supervisionado

Também designado de auto-organizado, no treinamento não-supervisionado não existe um elemento externo que inspecione o processo, isto é, não há um conjunto determinado de padrões a ser aprendido pela rede neural, mas uma medida de qualidade da representação requerida pela rede (Figura I.9), segundo a qual os parâmetros são ajustados. Extraídas algumas características dos dados de entrada, a rede será capaz de gerar representações internas que classificam estas características [26, 28].

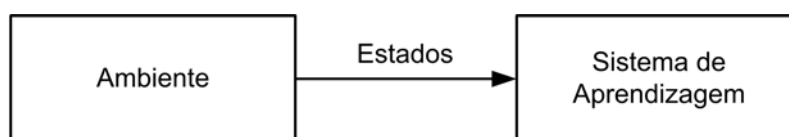


Figura I.9 – Treinamento não-supervisionado.

### I.2.6.3 – Treinamento por reforço

O treinamento por reforço compõe-se de um sistema no qual a rede aprende o comportamento desejado através de tentativas, admitindo-se acertos e erros durante a interação com determinado ambiente [38].

O treinamento por reforço pode ser considerado como um caso particular de treinamento supervisionado, tendo como principal diferença a utilização de um método do tipo crítico externo, o qual busca reforçar as ações boas cometidas pela rede [28]. O sinal de avaliação é um escalar e em caso extremo, existe uma única informação para indicar se a saída da rede está certa ou errada [39].

A Figura I.10 esquematiza um modelo básico de aprendizagem por reforço em que o sistema se comunica com o ambiente via ações (saídas) e percepções (entradas e sinal de reforço).

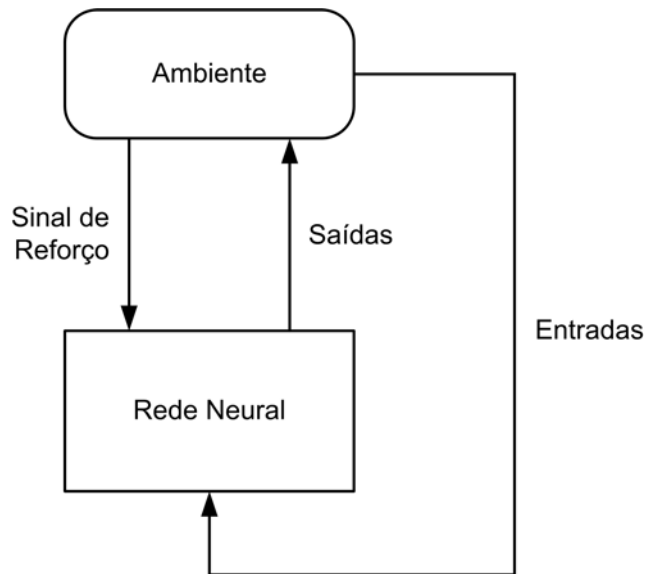


Figura I.10 – Treinamento por reforço.

Durante as iterações, o sistema de aprendizagem recebe como entrada um sinal indicando o estado atual do ambiente e o sistema faz a escolha da ação a ser tomada e gera a saída que será avaliada. Uma mudança no estado do ambiente é produzida por esta ação e comunicada ao aprendiz via sinal de reforço. É, portanto, através do processo iterativo de recompensa e punição que o sistema escolhe a ação que acrescenta o valor do sinal de reforço.

### I.2.7– Algoritmo de retropropagação de erro

O algoritmo de retropropagação de erro, também conhecido como *backpropagation*, é um dos mais usados e bem sucedidos algoritmos de treinamento supervisionado aplicado em redes com topologia MLP. Este algoritmo pode ser dividido basicamente em duas fases: a primeira se dá pela propagação dos sinais de entrada até a camada de saída, e a segunda, quando ocorre o sentido inverso da propagação [27].

O aprendizado consiste em comparar o sinal emitido pela rede com o sinal de referência. A diferença entre eles é o sinal de erro usado para ajustar os parâmetros livres da rede, de forma a minimizar a diferença quadrática entre as saídas do sistema desconhecido [28].

A partir do grafo de fluxo de sinal (Figura I.11), que representa a saída do neurônio  $j$ , na iteração  $n$ , o sinal de erro da saída correspondente à análise do  $n$ -ésimo par de exemplos apresentados à rede é definido pela eq. I.10.

$$e_j(n) = d_j(n) - y_j(n) \quad \text{eq. I.10}$$

onde,

- $e_j(n)$  - Erro instantâneo;
- $d_j(n)$  - Saída desejada;
- $y_j(n)$  - Saída calculada pela rede.

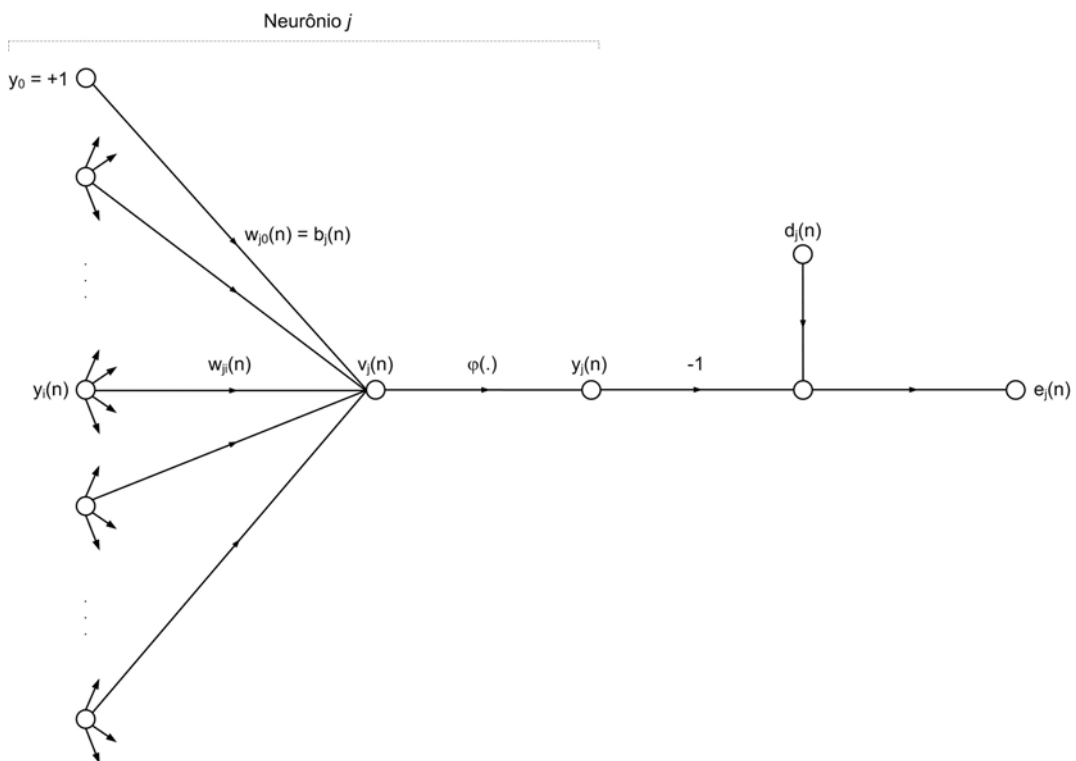


Figura I.11 – Grafo do fluxo de sinal da saída do neurônio  $j$  [21].

O valor do erro quadrático instantâneo para o neurônio  $j$  é dado pela eq. I.11.

$$E(n) = \frac{1}{2} e_j^2(n) \quad \text{eq. I.11}$$

E a soma dos erros quadráticos de todos os neurônios de saída, na iteração  $n$ , é obtida através da eq. I.12, onde  $C$  é o total de neurônios da camada de saída.

$$E(n) = \frac{1}{2} \sum_{j \in C} e_j^2(n) \quad \text{eq. I.12}$$

Considerando-se que  $N$  representa o número total de padrões entrada-saída do conjunto de treinamento, a eq. I.13 define a função de custo que se quer minimizar através do ajuste dos parâmetros livres da rede.

$$\bar{E} = \frac{1}{N} \sum_{n=1}^N E(n) \quad \text{eq. I.13}$$

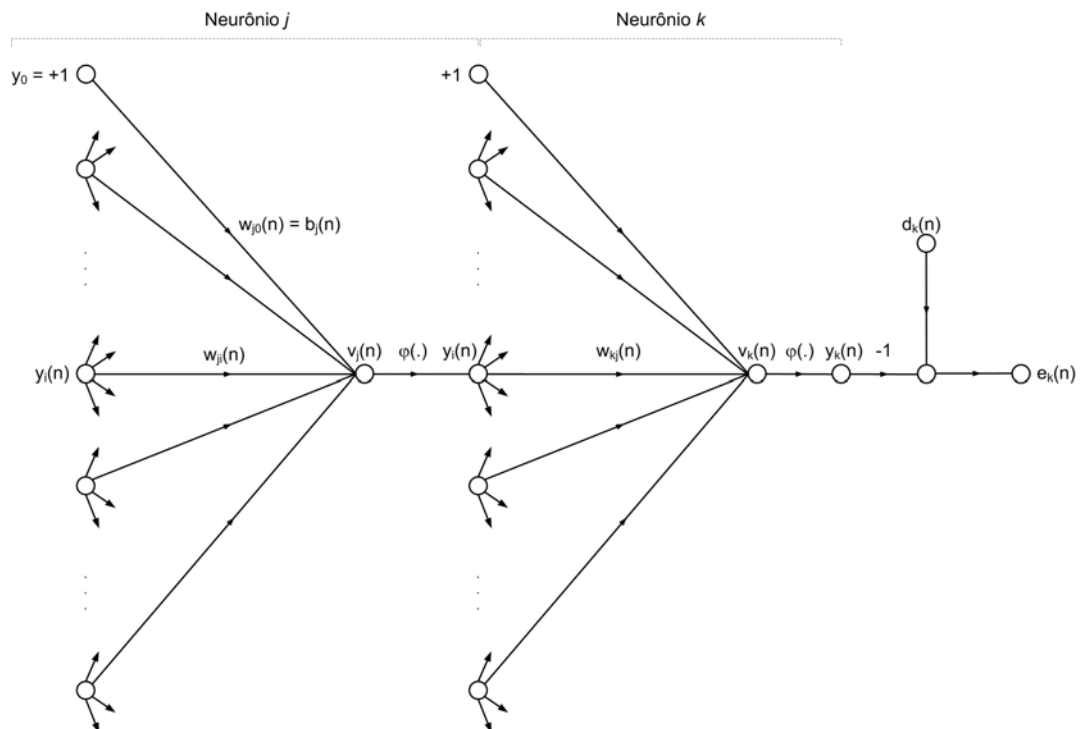


Figura I.12 – Grafo do fluxo de sinal do neurônio  $k$  conectado ao neurônio  $j$  [27].



Admitindo-se que o neurônio  $j$  recebe sinais da função  $y_i(n)$ , vindos dos neurônios  $i$  da camada anterior, sendo cada sinal associado a um peso  $w_{ji}(n)$ , a resposta do neurônio  $j$  é dada pela eq. I.14, onde  $I$  é o número total de neurônios que enviam sinais ao neurônio  $j$ , e  $w_{oj}(n)$  corresponde ao *bias* de  $j$ , conforme Figura I.12,

$$v_j(n) = \sum_{i=0}^I w_{ji}(n) y_i(n) \quad \text{eq. I.14}$$

Assim, o sinal produzido pela rede é apresentado na eq. I.15.

$$y_j(n) = \varphi(v_j(n)) \quad \text{eq. I.15}$$

O algoritmo de retropropagação aplica uma correção  $\Delta w_{ji}(n)$  no peso  $w_{ji}(n)$ , proporcional à derivada parcial, definida na eq. I.16.

$$\frac{\partial E(n)}{\partial w_{ji}(n)} \quad \text{eq. I.16}$$

O *backpropagation* baseia-se no ajuste dos pesos sinápticos proporcionalmente à diminuição do erro  $E(n)$ . Portanto, através da regra da cadeia tem-se a eq. I.17.

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = \frac{\partial E(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \frac{\partial v_j(n)}{\partial w_{ji}(n)} \quad \text{eq. I.17}$$

Diferenciando-se ambos os lados da eq. I.12, obtém-se:

$$\left( \frac{\partial E(n)}{\partial e_j(n)} \right) = e_j(n) \quad \text{eq. I.18}$$

Diferenciando-se ambos os lados da eq. I.10, tem-se:

$$\left( \frac{\partial e_j(n)}{\partial y_j(n)} \right) = -1 \quad \text{eq. I.19}$$

Diferenciando-se ambos os lados da eq. I.15, obtém-se:

$$\left( \frac{\partial y_j(n)}{\partial v_j(n)} \right) = \phi'(v_j(n)) \quad \text{eq. I.20}$$

Diferenciado-se a eq. I.14 em relação à  $w_{ji}(n)$ , tem-se:

$$\left( \frac{\partial v_j(n)}{\partial w_{ji}(n)} \right) = y_i(n) \quad \text{eq. I.21}$$

Substituindo as equações I.18 a I.21 na eq. I.10, obtém-se:

$$\left( \frac{\partial E(n)}{\partial w_{ji}(n)} \right) = -e_j(n) \phi'(v_j(n)) y_i(n) \quad \text{eq. I.22}$$

A correção  $\Delta w_{ji}(n)$  aplicada a  $w_{ji}(n)$  é determinada pela regra delta [27] como:

$$\Delta w_{ji}(n) = -\eta \left( \frac{\partial E_j(n)}{\partial w_{ji}(n)} \right) \quad \text{eq. I.23}$$

Onde  $\eta$  é o parâmetro da taxa de aprendizagem. O sinal negativo indica que o aprendizado é feito em direção à redução do gradiente de erro.

Substituindo-se eq. I.22 em I.23, tem-se:

$$\Delta w_{ji}(n) = \eta e_j(n) \phi'(v_j(n)) y_i(n) \quad \text{eq. I.24}$$

O gradiente local é definido por:

$$\delta_j(n) = \left( \frac{\partial E(n)}{\partial v_j(n)} \right) \quad \text{eq. I.25}$$

Aplicando-se a regra da cadeia, tem-se:

$$\delta_j(n) = - \left( \frac{\partial E(n)}{\partial e_j(n)} \right) \left( \frac{\partial e_j(n)}{\partial y_j(n)} \right) \left( \frac{\partial y_j(n)}{\partial v_j(n)} \right) \quad \text{eq. I.26}$$

$$\delta_j(n) = e_j(n) \phi'(v_j(n)) \quad \text{eq. I.27}$$

Substituindo-se I.27 em I.24, obtém-se:

$$\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n) \quad \text{eq. I.28}$$

Quando o neurônio  $j$  é um neurônio de saída, o cálculo do erro é bastante simples, pois a resposta desejada é conhecida. Determina-se então diretamente o sinal de erro e calcula-se o gradiente local usando a eq. I.24.

Quando o neurônio  $j$  se encontra em uma camada oculta não existe uma resposta desejada específica. Neste caso, o erro deve ser determinado a partir de sinais de erro oriundos dos neurônios da camada seguinte que estão conectados ao neurônio  $j$ .

De acordo com o grafo do fluxo de sinal da Figura I.12, pode-se redefinir o gradiente local  $\delta_j(n)$  para o neurônio oculto  $j$  através da eq. I.25, como:

$$\begin{aligned} \delta_j(n) &= - \left( \frac{\partial E(n)}{\partial v_j(n)} \right) \left( \frac{\partial y_i(n)}{\partial v_j(n)} \right) \\ &= - \frac{\partial E(n)}{\partial v_j(n)} \phi_j'(v_j(n)) \end{aligned} \quad \text{eq. I.29}$$

Considerando um neurônio  $k$ , participante da camada de saída, pode-se calcular o erro conforme eq. I.30.

$$E(n) = \frac{1}{2} \sum_{k \in C} e_k^2(n) \quad \text{eq. I.30}$$

Diferenciando a eq. I.30 em relação à  $y_j(n)$ , tem-se:

$$\frac{\partial E(n)}{\partial y_j(n)} = \sum_k e_k(n) \left( \frac{\partial e_k(n)}{\partial y_j(n)} \right) \quad \text{eq. I.31}$$

Aplicando-se a regra da cadeia na eq. I.31, obtém-se:

$$\frac{\partial E(n)}{\partial y_j(n)} = \sum_k e_k(n) \left( \frac{\partial e_k(n)}{\partial v_k(n)} \right) \left( \frac{\partial v_k(n)}{\partial y_j(n)} \right) \quad \text{eq. I.32}$$

Como o erro na última camada pode ser calculado com base na resposta desejada, para o neurônio  $k$ , tem-se:

$$\begin{aligned} e_k(n) &= d_k(n) - y_k(n) \\ &= d_k(n) - \varphi_k(v_k(n)) \end{aligned} \quad \text{eq. I.33}$$

Diferenciando-se ambos os lados da eq. I.33, tem-se:

$$\left( \frac{\partial e_k(n)}{\partial v_k(n)} \right) = -\varphi'_k(v_k(n)) \quad \text{eq. I.34}$$

A resposta para o neurônio  $k$  é dada pela eq. I.35, onde  $m$  é o total de entradas, excluindo-se o *bias*.

$$v_k(n) = \sum_{j=0}^m w_{kj}(n) y_j(n) \quad \text{eq. I.35}$$

Diferenciando-se a eq. I.32 em relação à  $y_j(n)$ , obtém-se:

$$\left( \frac{\partial v_k(n)}{\partial y_j(n)} \right) = w_{kj}(n) \quad \text{eq. I.36}$$

Substituindo-se as equações I.34 e I.36 em I.32, tem-se a relação desejada:

$$\left( \frac{\partial E(n)}{\partial y_j(n)} \right) = - \sum_k e_k(n) \varphi'_k(v_k(n)) w_{kj}(n) \quad \text{eq. I.37}$$

Substituindo-se a equação I.27 em I.37, tem-se:

$$\left( \frac{\partial E(n)}{\partial y_j(n)} \right) = - \sum_k \delta_k(n) w_{kj}(n) \quad \text{eq. I.38}$$

Substituindo-se I.37 em I.29, obtém-se a fórmula da retropropagação para o gradiente local  $\delta_j(n)$  quando  $j$  é um neurônio oculto.

$$\delta_j(n) = \varphi'_j(v_j(n)) \sum_k \delta_k(n) w_{kj}(n) \quad \text{eq. I.39}$$

Resumindo, tem-se:

1. Sendo  $j$  um neurônio de saída, o gradiente local é igual ao produto da derivada  $\varphi'_j(v_j(n))$  pelo sinal de erro  $e_j(n)$ , ambos associados ao neurônio  $j$ , conforme eq. I.27;
2. Sendo  $j$  um neurônio oculto, o gradiente local  $\delta_j(n)$  é igual ao produto da derivada  $\varphi'_j(v_j(n))$  pela soma ponderada dos gradientes locais determinados para a próxima camada oculta ou de saída, conectadas ao neurônio  $j$ , de acordo com a eq. I.39

## I.2.8– Algumas considerações para treinamento

### I.2.8.1– Número de neurônios da camada escondida

O desempenho da rede neural tem relação com o número de neurônios em cada camada, pois quando o número de parâmetros torna-se demasiado, a rede pode “decorar” os dados do treinamento (*overfitting*) ao invés de extrair as características gerais que permitem a generalização, ou seja, a rede superdimensionada inclui todas as soluções do modelo de

menor complexidade, mais aquelas proporcionadas pelos parâmetros excedentes. Por outro lado, caso o número de neurônios seja subestimado, a rede não converge (*underfitting*), que significa a impossibilidade de se determinar os pesos corretamente [26, 36].

O número de parâmetros de uma rede neural pode ser calculado a partir da eq. I.40.

$$n_p = (N_{ent} \cdot N_{esc} + N_{esc}) + (N_{esc} \cdot N_{sai} + N_{sai}) \quad \text{eq. I.40}$$

onde,

- $n_p$  – Número total de pesos ou parâmetros;
- $N_{ent}$  – Número de neurônios da camada de entrada;
- $N_{esc}$  – Número de neurônios da camada escondida;
- $N_{sai}$  – Número de neurônios da camada de saída.

Como os neurônios das camadas de entrada e saída são inerentes ao problema, a complexidade do modelo neural recai sobre o número de neurônios da camada escondida, arbitrado conforme a experiência do projetista. Entretanto, torna-se conveniente adotar alguma heurística para se aproximar do número adequado de neurônios da camada escondida, como o teorema de Kolmogorov [26, 40], definido pela eq. I.41.

$$N_{esc} = 2 \cdot N_{ent} + 1 \quad \text{eq. I.41}$$

A proposição de Kolmogorov, porém, desconsidera parâmetros úteis, como número de exemplos de treinamento e o EQM de generalização admissível. Uma regra que prevê a relação entre o número total de parâmetros da rede e a quantidade de pares de treinamento (T) foi proposta por Baum & Haussler (eq. I.42) [41], onde  $E > 0$  é o erro percentual máximo admissível.

$$T > \frac{n_p}{E} \quad \text{eq. I.42}$$

Substituindo-se a equação 1.40 em 1.42, tem-se:

$$T > \frac{(N_{ent} \cdot N_{esc} + N_{esc}) + (N_{esc} \cdot N_{sai} + N_{sai})}{E} \quad \text{eq. 1.43}$$

Isolando-se o termo  $N_{esc}$  de eq. 1.43, obtém-se a relação:

$$N_{esc} \approx \left[ \frac{E \cdot T - N_{sai}}{N_{ent} + N_{sai} + 1} \right] \quad \text{eq. 1.44}$$

### 1.2.8.2– Inicialização dos Pesos

Os valores iniciais dos pesos contribuem para a qualidade e eficiência da aprendizagem supervisionada, assim como a escolha adequada da estrutura da rede, definição de funções de ativação e regras de aprendizagem. O conjunto inicial de pesos a ser utilizado tem relação direta com a velocidade de aprendizagem e na qualidade da solução final. Uma escolha inicial pouco acertada pode trazer problemas de mínimos locais pobres ou de saturação antecipada. Usualmente, não é possível conhecer previamente os valores ótimos para estes parâmetros, devido às características do conjunto de treinamento bem como da natureza da solução. Então, recomenda-se que a inicialização dos parâmetros seja feita randomicamente, com distribuição uniforme e magnitude pequena [27, 42].

### 1.2.8.3– Modos de treinamento

Denomina-se de época de treinamento quando todo o conjunto de pares entrada-saída é apresentado à rede neural. O processo de treinamento persiste enquanto não se alcança um número máximo de épocas ou o erro quadrático médio não atinge um valor suficientemente pequeno. O algoritmo de retropropagação pode ser aplicado de dois modos diferentes para um determinado conjunto de treinamento [27, 28]:

1. Padrão a padrão – Neste modo, a rede atualiza os pesos a cada padrão entrada-saída apresentado, ou seja, os pesos são ajustados T vezes por época, sendo cada ajuste baseado no cálculo do gradiente de erro de um único padrão por vez, onde T é a quantidade de padrões do conjunto de treinamento;

2. Em lote – No aprendizado em lote ou *batch*, um único ajuste dos pesos é feito a cada época. Neste caso, todos os padrões de treinamento são propagados pela rede e os erros correspondentes são calculados, sendo os pesos atualizados pela soma dos gradientes de cada padrão apresentado à época.

Recomenda-se apresentar os padrões em ordem aleatória para cada época de treinamento. Isto faz que a busca no espaço dos pesos seja estocástica, garantindo que a ordem dos padrões de treinamento não represente informação significativa para condicionar o ajuste dos pesos [36].

#### 1.2.8.4– Taxa de aprendizagem

A taxa de aprendizagem atua como reguladora da velocidade com a qual os ajustes dos pesos se dirigem ao erro mínimo. Para valores pequenos de  $\eta$ , menor será a variação dos pesos de uma iteração para a próxima e, no caso do algoritmo de retropropagação, a trajetória no espaço dos pesos será mais suave. Por outro lado, se  $\eta$  toma valores muito altos, a velocidade de convergência aumenta, mas a variação dos pesos entre iterações será também maior, podendo afetar a convergência do algoritmo [27].

#### 1.2.8.5– Constante de momento

Uma solução para aumentar a taxa de aprendizagem sem comprometer a convergência é alterar a regra delta da equação 1.28 por meio da inclusão de um termo denominado constante de momento, representado por  $\alpha$ , conforme eq. 1.45 [27]. O intervalo da constante de momento deve ser restrito ao intervalo  $0 \leq |\alpha| \leq 1$  [28]. Para  $\alpha = 0$ , o treinamento ocorre sem a influência da constante de momento.

$$\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n) + \alpha \Delta w_{ji}(n-1) \quad \text{eq. 1.45}$$



A constante de momento confere inércia ao aprendizado, aumentando a velocidade de convergência em regiões de descida da superfície do erro, uma vez que o ajuste considera parte do ajuste anterior. Entretanto, este método deve ser usado com cautela, uma vez que também é sensível o suficiente para causar instabilidade, dependendo da superfície de erro [28]. A Figura I.13 demonstra a atuação do termo momento na superfície de erro.

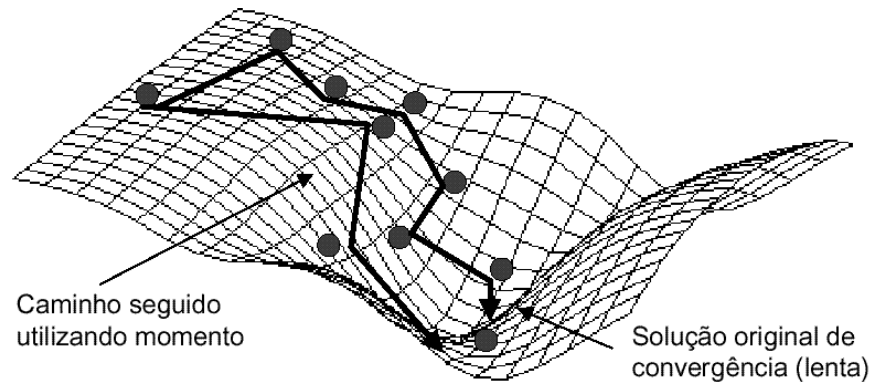


Figura I.13 – Atuação do termo de momento [28].

#### I.2.8.6– Critérios de parada

Não existe critério de parada para o algoritmo de retropropagação que garanta a obtenção da melhor solução. Contudo, dois critérios são comumente utilizados para dar término antecipado ao processo de treinamento [27]:

1. Considerando as propriedades dos mínimos locais ou globais da superfície de erro e sendo  $w = w^*$  um mínimo local ou global, o vetor gradiente calculado para este mínimo será aproximadamente nulo. Então, se a norma Euclidiana do vetor gradiente for menor que um valor suficientemente pequeno, entende-se que a convergência tenha sido atingida; e
2. Quando os valores do erro quadrático médio ou da função objetivo deixam de variar.

Além destes, há o critério de validação cruzada, que atua na prevenção do *overfitting* através da monitoração do erro de generalização durante o processo de treinamento [36]. A capacidade de generalização da rede neural está em fornecer respostas adequadas para padrões diferentes daqueles processados na fase de treinamento. Logo, após cada ajuste de pesos, a rede é avaliada com o conjunto de teste, sendo o treinamento finalizado caso o EQM de teste apresente tendência de crescimento em relação ao obtido pelo treinamento [28].

#### I.2.8.7– Normalização dos dados

Os dados de entrada devem ser tratados por um processo de normalização, para torná-los apropriados ao processo de treinamento da rede, a fim de evitar a saturação das funções de ativação e garantir que os atributos dos dados tenham o mesmo domínio [27]. A normalização pode ser feita eficientemente de duas formas [43], onde  $x$  representa a variável do conjunto de dados:

1. Média 0 (zero) e desvio-padrão 1, representada na eq. I.46.

$$y = \frac{x - \bar{x}}{\sigma(x)} \quad \text{eq. I.46}$$

2. Valor mínimo -1 e valor máximo 1, obtidos pela eq. I.47, onde  $\min(x)$  e  $\max(x)$  representam os valores mínimo e máximo, respectivamente.

$$y = \frac{x - [\max(x) + \min(x)]/2}{[\max(x) - \min(x)]/2} \quad \text{eq. I.47}$$

## I.3– SISTEMAS NÃO-LINEARES

### I.3.1– Algumas considerações em sistemas não-lineares

Sistemas lineares são aqueles representados por equações lineares e têm como característica o princípio da superposição, o qual estabelece que a resposta de uma aplicação simultânea de duas forças de excitação diferentes é igual à soma das duas respostas individuais [5]. Sistemas não-lineares, por outro lado, são aqueles que têm pelo menos um componente não-linear na equação do modelo [1].

Slotine & Li [1] frisam que sistemas físicos são inerentemente não-lineares e, conseqüentemente, demandam por sistemas de controle também não-lineares. Por exemplo, há fenômenos que acontecem somente em sistemas não-lineares, como dependência da amplitude de excitação, tempo de escape finito, pontos de equilíbrio múltiplos, não-unicidade da solução, dependência crítica dos parâmetros, bifurcações, caos ou dependência crítica das condições iniciais, ciclos limite ou oscilações e existência de harmônicas e de sub-harmônicas. Detalhes sobre estes comportamentos são encontrados em [1 – 5].

Diante deste cenário, pesquisadores e projetistas de diversas áreas como controles aeronáutico e espacial, robótica, de processos e engenharia biomédica, sentem-se motivados a desenvolver e aplicar metodologias de controle não-linear. Entre as principais motivações, destacam-se [1]:

- Melhoria do desempenho de controladores existentes: modelos linearizados assumem que o sistema opera próximo a um ponto de operação. Caso esta condição não seja válida, os controladores lineares apresentam desempenho pobre ou não garantem a estabilidade. Por outro lado, controladores não-lineares podem atuar diretamente sobre uma faixa superior de operação. Por exemplo, em controle de movimentos robóticos, o modelo linear negligencia as forças não-lineares associadas e tem seu desempenho rapidamente degradado, enquanto o controlador

não-linear é capaz de compensar completamente as forças não-lineares de movimento do robô;

- Análise de não-linearidades rígidas: É comum em sistemas de controle a presença de não-linearidades cuja natureza descontínua impede que seus efeitos sejam representados por aproximações lineares, como atrito de Coulomb, saturação, zona-morta, liga-desliga, folga e histerese. Portanto, métodos de análise de sistemas não-lineares tornam-se desejáveis para prever o desempenho de sistemas na presença destes tipos de não-linearidades que, freqüentemente, causam comportamentos indesejados aos sistemas de controle;
- Lidar com modelos de incertezas: Para projetar controladores lineares, normalmente é necessário assumir que os parâmetros do modelo sejam bem conhecidos. Contudo, problemas de controle envolvem incertezas que podem variar no tempo lenta ou abruptamente. Além disso, a insistência de um controlador linear baseado em parâmetros incorretos pode ter como consequência a degradação do desempenho ou mesmo instabilidade.

Além destas, o emprego de técnicas lineares em instalações industriais para controlar máquinas que tenham elevada faixa operacional costuma significar custos indevidos e longos períodos de desenvolvimento, pois dependem da qualidade dos atuadores e sensores, e as restrições do código de controle dificultam seu transporte mesmo para aplicações similares. Os controladores PID, por exemplo, embora presentes em 80% do parque industrial, são inadequados para aplicações em plantas essencialmente não-lineares [8].

Diferente dos sistemas lineares, as características de resposta de sistemas não-lineares dependem, de certa maneira, dos valores de entrada, de forma que não existem métodos

gerais de análise de sistemas não-lineares [4]. Não linearidades muito correntes em processos e sistemas reais, como atritos de Coulomb, folgas, zonas mortas, saturações etc. dificultam a linearização do sistema em torno de um ponto de funcionamento [1, 3].

Controle não-linear, por sua vez, é o conjunto de procedimentos destinados a fazer com que as variáveis de saída de um sistema não-linear se aproximem de uma determinada referência e estabilizem na vizinhança do seu valor. Como acontece no caso linear, o controle não-linear utiliza realimentação, quer da saída quer do estado, para gerar um sinal de controle que vai atuar sobre o processo. Por vezes, a malha de realimentação é projetada como não-linear, a fim de compensar as não-linearidades do sistema a ser controlado ou para melhorar certos aspectos da planta [1].

Controle não-linear envolve análise, para determinar as características de comportamento do sistema e projeto, com fim de elaborar o controlador para a planta não-linear, de modo que o sistema controlado atenda a requisitos previamente estabelecidos. Na prática as tarefas de projeto e análise estão interconectadas, pois o projeto de sistemas de controle não-lineares normalmente envolve processos iterativos de análise e projeto [1]

Em sistemas de controle não-lineares, os métodos clássicos utilizados no estudo dos sistemas lineares, em particular a análise em frequência, não são aplicáveis em sistemas não-lineares, pois nem sempre é possível encontrar soluções analíticas diretas para equações diferenciais não-lineares, e não se aplicam transformações para o domínio da frequência [1, 4]. Contudo, vários métodos têm sido propostos, como análise pelo plano de fase, teoria de Lyapunov e o método do primeiro harmônico ou função descritiva, cujos detalhes podem ser obtidos em [1 – 6].

Para suprir as dificuldades implícitas em determinar uma lei de controle para sistemas não-lineares submetidos à condição em que a determinação do modelo é complexa no sentido de se limitar a faixa operacional do controlador, ou não se conhecem os fenômenos físico-químicos presentes, ou ainda, não há certeza sobre os parâmetros disponíveis, e que o

formalismo para análise de desempenho, otimização e robustez já não é suficiente para determinar uma lei de controle viável, novas soluções a partir de sistemas heurísticos, baseadas em regras de inferência, decisões baseadas em lógica nebulosa, redes neurais artificiais etc. têm sido propostas [9].

No passado, a aplicação de métodos de controle não-linear havia sido limitada pelas dificuldades computacionais associadas com projetos e análises de controle não-linear. Porém, atualmente, com o auxílio de modernos microprocessadores e *softwares* especializados, a implementação de controladores não-lineares torna-se bastante atrativa.

### 1.3.2– Não-linearidades rígidas

Diferentes tipos de não-linearidades podem ser encontrados em sistemas de controle reais, sendo classificadas como inerentes ou intencionais. São inerentes quando existem naturalmente nos sistemas. Alguns exemplos de não-linearidades inerentes são: saturação, zona-morta, histerese, folga (*backlash*), atrito estático, atrito de Coulomb e outros tipos de atrito não-lineares, mola não-linear, compressibilidade de fluidos etc. Usualmente estas não-linearidades causam efeitos indesejáveis, como ciclo-limite e instabilidade, e os controladores devem compensá-las apropriadamente. Já as não-linearidades intencionais são aquelas introduzidas deliberadamente em um sistema para melhorar seu desempenho e/ou simplificar sua construção. [1, 5]. Geralmente, estas não-linearidades são chamadas de rígidas ou descontínuas [1].

Neste trabalho são consideradas, para efeito de representação por RNA, as não-linearidades de saturação, zona morta, saturação com zona morta, liga-desliga, liga-desliga com histerese e liga-desliga com zona morta, discriminadas nas seções seguintes.

### I.3.2.1 – Saturação

A maioria dos sistemas físicos exibe uma curva característica da entrada-saída para a não-linearidade saturação, conforme indicada na Figura I.14. A saída de um elemento de saturação é proporcional à entrada quando os sinais de entrada são pequenos, mas não aumenta proporcionalmente para sinais de entrada maiores [5].

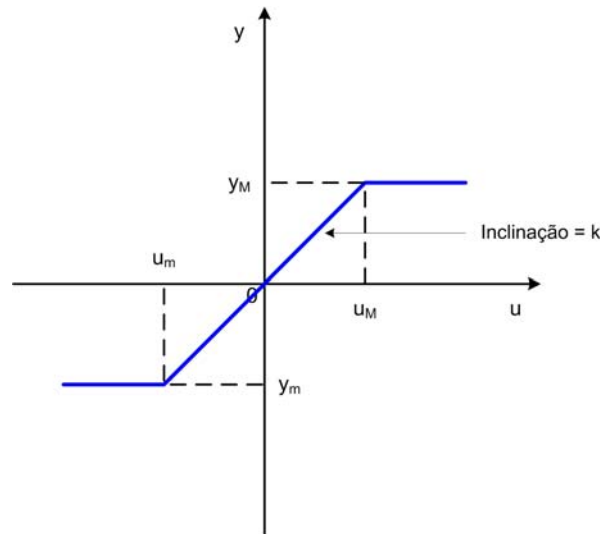


Figura I.14 – Característica entrada-saída para saturação [6].

A função saturação, representada na Fig. 1.14, é descrita analiticamente assim [6]:

$$y = \begin{cases} y_m & \text{se } u < u_m \\ ku + c & \text{se } u_m \leq u \leq u_M \\ y_M & \text{se } u > u_M \end{cases} \quad \text{eq. I.48}$$

em que  $y_m = y(u_m)$  e  $y_M = y(u_M)$ . Qualquer dos valores  $y_m$  ou  $y_M$  pode ser positivo ou negativo, com a condição  $y_m < y_M$ . Um dos valores,  $y_m$  ou  $y_M$  poderá ser infinito (em módulo), mas apenas um deles.

A saturação é muito comum em dispositivos de controle industrial, como em válvulas, onde o efeito pode ser aplicado para não forçar a abertura do mecanismo além da sua faixa operacional.

### I.3.2.2 – Zona morta

A não-linearidade de zona morta, também chamada de limiar, pode ser de dois tipos, designadas por tipos I e II [6], caracterizada por uma curva de entrada-saída semelhante às da Figura I.15, onde as saídas são nulas para as entradas compreendidas na faixa de zona morta [5].

A zona morta ocorre com frequência nos processos industriais e é causadora de instabilidades [8], mas também pode ser aplicada intencionalmente em sistemas de controle para enviar um sinal nulo do controlador para o atuador, quando a saída daquele for pequena, contribuindo para o tempo de vida útil de determinados tipos de equipamentos mecânicos [5].

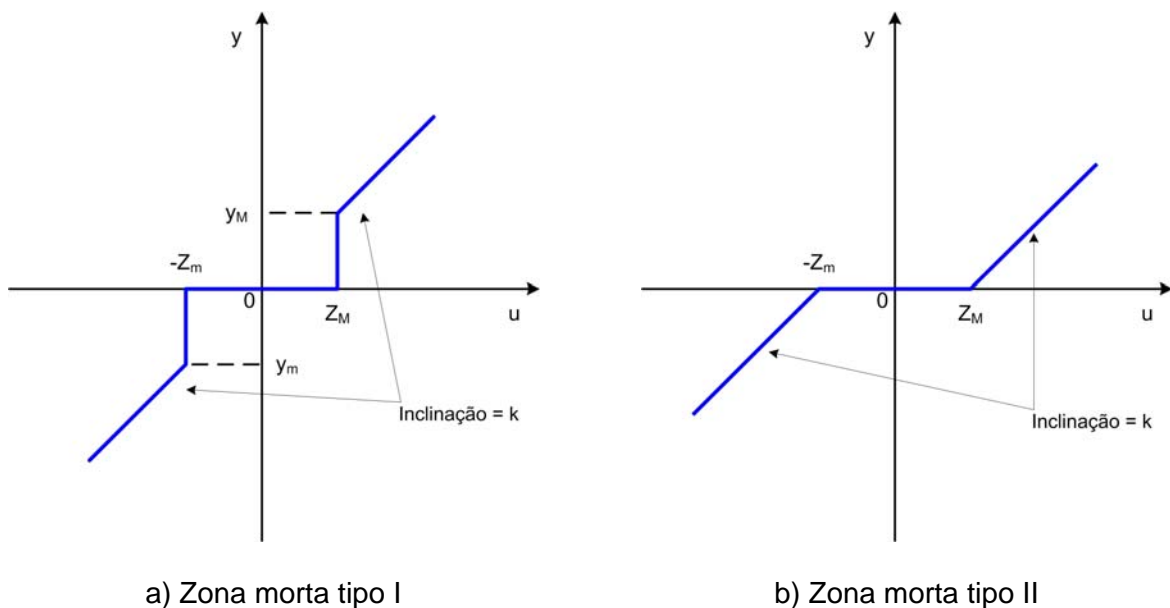


Figura I.15 – Característica entrada-saída para zona morta [6].

A relação entre  $u$  e  $y$  pode ser escrita analiticamente para os dois tipos de zona morta, como sendo, respectivamente [6]:

Zona morta tipo I,

$$y = \begin{cases} 0 & \text{se } |u| \leq Z_M \\ u & \text{se } |u| > Z_M \end{cases} \quad \text{eq. I.49}$$



Zona morta tipo II,

$$y = \begin{cases} u + Z_M & \text{se } u < -Z_M \\ 0 & \text{se } -Z_M \leq u \leq Z_M \\ u - Z_M & \text{se } u > Z_M \end{cases} \quad \text{eq. I.50}$$

### I.3.2.3– Saturação com zona morta

A não-linearidade de saturação com zona morta caracteriza-se pela combinação das duas condições tratadas anteriormente. A relação entre  $u$  e  $y$  pode ser vista na Figura I.16.

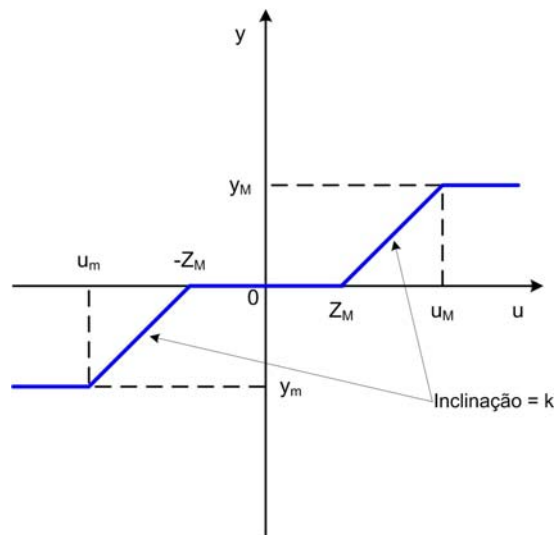


Figura I.16 – Característica entrada-saída para saturação com zona morta.

A descrição analítica da saturação com zona morta pode ser descrita conforme a seguir:

$$y = \begin{cases} y_M & \text{se } u \geq u_M \\ u + Z_M & \text{se } u_m < u < -Z_M \\ 0 & \text{se } -Z_M \leq u \leq Z_M \\ u - Z_M & \text{se } Z_M < u < u_M \\ y_m & \text{se } u \leq u_m \end{cases} \quad \text{eq. I.51}$$

### I.3.2.4– Liga-desliga

Esta não-linearidade tipo liga-desliga, também designada por *on-off*, *bang-bang* ou de duas posições, tem a curva característica expressa pela Figura I.17. A saída deste elemento é uma constante positiva ou negativa [5].

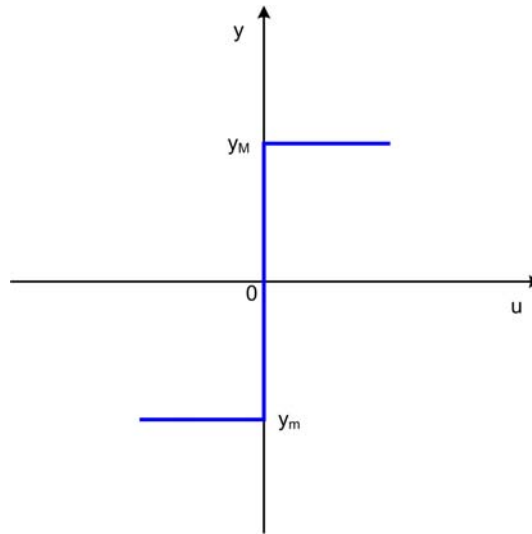


Figura I.17 – Característica entrada-saída para liga-desliga [6].

Este tipo de não-linearidade tem a seguinte descrição analítica [6]:

$$y = \begin{cases} y_m & \text{se } u < 0 \\ \text{não definido} & \text{se } u = 0 \\ y_M & \text{se } u > 0 \end{cases} \quad \text{eq. I.52}$$

A não linearidade liga-desliga é uma ação de controle simples e econômica, comumente empregada nos meios industriais. Contudo, possui limitações no que diz respeito ao comportamento dinâmico e em regime permanente do sistema em malha fechada. Por isso, suas aplicações restringem-se a sistemas onde não sejam exigidos precisão e desempenho dinâmico [5].

### I.3.2.5– Liga-desliga com histerese

No caso da função liga-desliga ser implementada como simples comparador ou relé físico, pode haver inconsistência em zero e, devido à presença de ruídos, também podem ocorrer chaveamentos espúrios quando o sinal de entrada estiver próximo de zero. Para evitar este efeito indesejável, introduz-se a não-linearidade liga-desliga com histerese (Figura I.18), na qual a história do sinal é particularmente útil. Para que haja chaveamento, dois sinais devem ser analisados, a entrada  $u(t)$  e a saída  $y(t-1)$ .

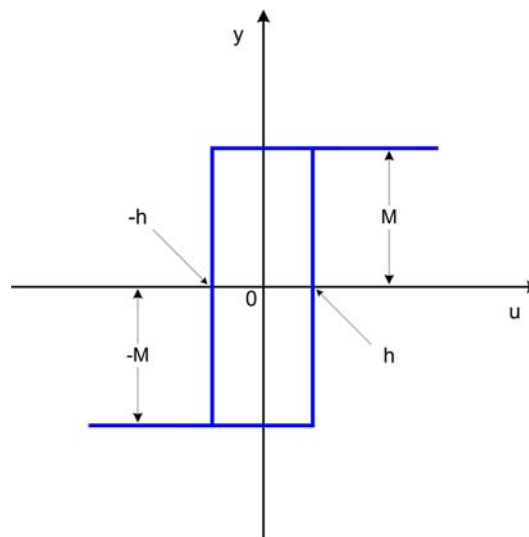


Figura I.18 – Característica entrada-saída para liga-desliga com histerese [5].

A descrição analítica de um controlador liga-desliga tem sua implementação discutida na seção II.2.7.2.5.

### I.3.2.6– Liga-desliga com zona morta

De acordo o sinal de entrada, a aplicação da não-linearidade liga-desliga pode deixar de responder na faixa de operação caracterizada como zona morta. A Figura I.19 representa a ação deste efeito não-linear.

A seguinte descrição analítica é atribuída para esta não-linearidade:

$$y = \begin{cases} y_M & \text{se } u > Z_M \\ 0 & \text{se } -Z_M \leq u \leq Z_M \\ y_m & \text{se } u < -Z_M \end{cases} \quad \text{eq. I.53}$$

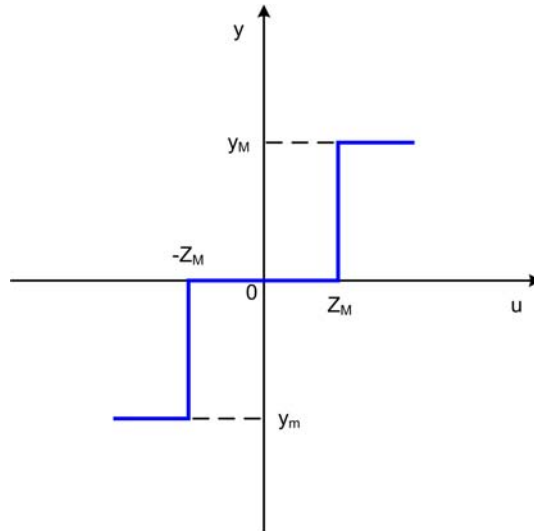


Figura I-19 – Característica entrada-saída para liga-desliga com zona morta.

## CAPÍTULO II

### MATERIAIS E MÉTODOS

#### II.1– RECURSOS UTILIZADOS

A presente dissertação utiliza-se de simulações computacionais, realizadas com os seguintes recursos:

##### II.1.1– Hardware

Microcomputador PC-AT Processador Intel™ *Celeron* 1.4 GHz, 512 MB de memória RAM e HD de 40 GB.

##### II.1.2- Software

As redes neurais artificiais, incluindo os padrões de treinamento, foram implementadas no ambiente de programação MATLAB® versão 7.5.0.342 (R2007b), sobre sistema operacional Microsoft Windows XP® 2002 SP2.

A adoção do MATLAB (que abrevia *MATrix LABORatory* – Laboratório de Matrizes) deve-se ao fato deste ser um *software* de programação de uso específico para executar cálculos matemáticos e de engenharia bastante conhecido nos meios acadêmicos e de pesquisa científica; e disponível para uso no Laboratório de Controle e Automação do CEFET/RJ.

O MATLAB tem ainda as vantagens de uma linguagem de alto nível, face às soluções dos problemas serem expressas de um modo bem próximo das expressões matemáticas; código portátil, ou seja, os programas são executados sobre diversos sistemas operacionais, como Windows 9x, NT, 2000 e em diversas versões de UNIX; possui bibliotecas de funções pré-definidas, que proporcionam soluções para necessidades rotineiras; caixas de ferramentas para resolver problemas em processamentos de sinais, sistemas de controle, processamento de imagens etc.; facilidade para exibir desenhos e imagens técnicas e científicas; interface

gráfica amigável com ferramentas interativas [44, 45]. É oportuno destacar que o *Matlab Neural Networks Toolbox* não é utilizado neste trabalho.

## II.2– DEFINIÇÃO DOS PARÂMETROS DA RNA

A seguir é apresentado o processo de configuração da rede neural com base nas informações teóricas levantadas. A fim de facilitar a organização, os códigos de programação estão disponíveis na forma de apêndices.

### II.2.1– Número de neurônios da camada escondida

A partir do teorema de Kolmogorov, apresentado na seção I.2.8.1, a camada escondida tem três neurônios em todas os casos, exceto para a simulação da não-linearidade de ligação com histerese, que recebe cinco neurônios na referida camada.

### II.2.2– Funções de ativação

A rede MLP exige que a função de ativação empregada em pelo menos uma das camadas intermediárias seja diferenciável [27], pois a utilização de funções lineares somente tornaria a rede como sendo de uma só camada e, portanto, limitada [28].

Funções não-lineares que atendem aos requisitos da rede MLP são, por exemplo, sigmóide e tangente hiperbólica. Neste caso, tanto a camada intermediária como a de saída utilizam a tangente hiperbólica, apresentada na seção I.2.4. Esta escolha visa aproveitar os levantamentos empíricos que lhe atribuem melhor rendimento no processo de treinamento, se comparada com a função logística [46].

### II.2.3– Modo de treinamento

Esta dissertação utilizou a rede neural tipo MLP com treinamento supervisionado por retropropagação de erro, sendo o modo treinamento padrão a padrão, conforme discutido na seção I.2.8.3.

#### II.2.4– Inicialização dos pesos

Os pesos iniciais recebem valores iniciais contidos no intervalo  $[-0.1; 0.1]$ .

#### II.2.5– Taxa de aprendizagem e constante de momento

A taxa de aprendizagem e a constante de momento são parâmetros empíricos, estipulados em 0.1 e 0.75, respectivamente. Para as simulações complementares, estes valores podem ser alterados conforme o comportamento da curva de erro quadrático médio, em termos de convergência da rede.

#### II.2.6– Critério de parada

São definidos três critérios de parada:

1. Número máximo de épocas. A configuração padronizada tem cem épocas para treinamento, enquanto as simulações complementares têm valor diverso, estimado conforme a possibilidade de convergência da rede quanto ao cumprimento do EQM de generalização desejável;

2. EQM de teste inferior a 0.001. O erro de teste é aferido a cada cinco épocas e interrompe o processo de treinamento caso o valor obtido seja inferior a 0.001 por três vezes consecutivas;

3. EQM de teste com tendência de crescimento em relação ao EQM de treinamento. O EQM de teste é verificado a cada cinco épocas e finaliza o processo de treinamento caso apresente valor superior ao obtido pelo EQM de treinamento, por três vezes consecutivas.

## II.2.7– Conjunto de treinamento

### II.2.7.1– Tamanho do conjunto de treinamento

O tamanho do conjunto de treinamento pode ser definido arbitrariamente antes de cada simulação. Entretanto, pretende-se determinar um tamanho ótimo em torno dos parâmetros de número de neurônios da camada escondida. Para isso, adota-se a regra de Baum-Haussler (eq. I.43), considerando-se o erro percentual máximo de generalização de 0.1%. Dessa forma, as simulações de saturação, zona morta tipos I e II, saturação com zona morta, liga-desliga e liga-desliga com zona morta recebem 10001 pares de entrada-saída no conjunto de treinamento, e para a não-linearidade de liga-desliga com histerese, o referido conjunto é constituído por 21001 padrões.

Com a finalidade de melhorar a capacidade de generalização e permitir a avaliação periódica da rede, o conjunto de treinamento é dividido aleatoriamente em duas partes, sendo uma formada por 80% dos dados, utilizada no ajuste dos pesos, e a outra, constituída pelos 20% restantes para a fase de teste, na qual os pesos se mantêm fixos. Cada subconjunto de pares entrada-saída tem seus respectivos índices permutados a cada época.

### II.2.7.2– Pares entrada-saída

Os sinais de entrada devem ser significativos e representar amplamente o comportamento do sistema em condições operativas, não devendo satisfazer apenas as operações normais e rotineiras, mas também as exceções contidas nos limites do domínio do problema, como ruídos e perturbações, por exemplo [13].

Para este intento, admitindo-se que não foi utilizado um sistema físico real, as variáveis de entrada são emuladas a partir de eq. II.1.

$$f(u) = 2.0 \left[ \cos(7\pi u) \sin(10\pi u) + \cos(58\pi u) + \sin(0.66\pi u) + \sin(6\pi u) + \sin(10\pi u) \right] \quad \text{eq. II.1}$$

O sinal desejado, por sua vez, é formado de acordo com as não-linearidades discriminadas nas seções II.2.7.2.1 a II.2.7.2.6.



## – Saturação

Nesta simulação, a saturação ocorre para sinais de entrada ( $u$ ) fora do intervalo  $[-6;6]$ , de acordo com a eq. II.2, onde  $d$  representa o sinal de saída. A Figura II.1 ilustra o efeito não-linear de saturação desejado.

$$d = \begin{cases} -1 & \text{se } u < -6 \\ \frac{u}{6} & \text{se } -6 \leq u \leq 6 \\ 1 & \text{se } u > 6 \end{cases} \quad \text{eq. II.2}$$

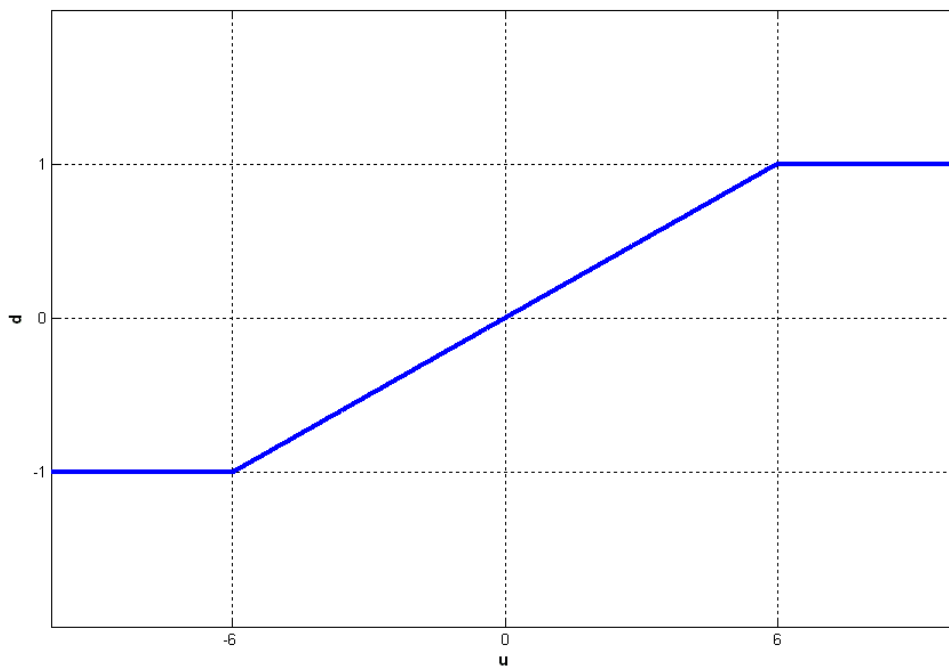


Figura II.1 – Gráfico da não-linearidade de saturação desejada.

O código MATLAB para composição dos pares entrada-saída da não-linearidade de saturação consta do Apêndice 1.

– Zona morta

– Zona morta tipo I

A não-linearidade de zona morta define-se para sinais de entrada contidos no intervalo  $[-3;3]$ , conforme eq. II.3, demonstrada pela Figura II.2.

$$d = \begin{cases} 0 & \text{se } |u| \leq 3 \\ u & \text{se } |u| > 3 \end{cases} \quad \text{eq. II.3}$$

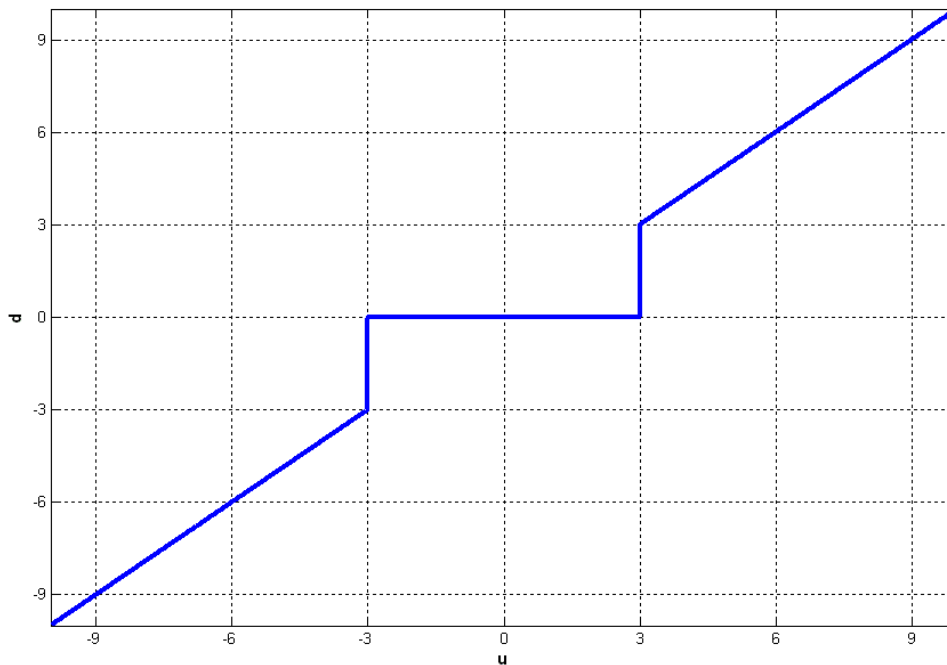


Figura II.2 – Gráfico da não-linearidade de zona morta tipo I desejada.

– Zona morta tipo II

A zona morta tipo II acontece para sinais de entrada contidos no intervalo  $[-4;4]$ , conforme eq. II.4. A Figura II.3 apresenta a não-linearidade de zona morta desejada.

$$d = \begin{cases} u+4 & \text{se } u \leq -4 \\ 0 & \text{se } -4 < u < 4 \\ u-4 & \text{se } u \geq 4 \end{cases} \quad \text{eq. II.4}$$

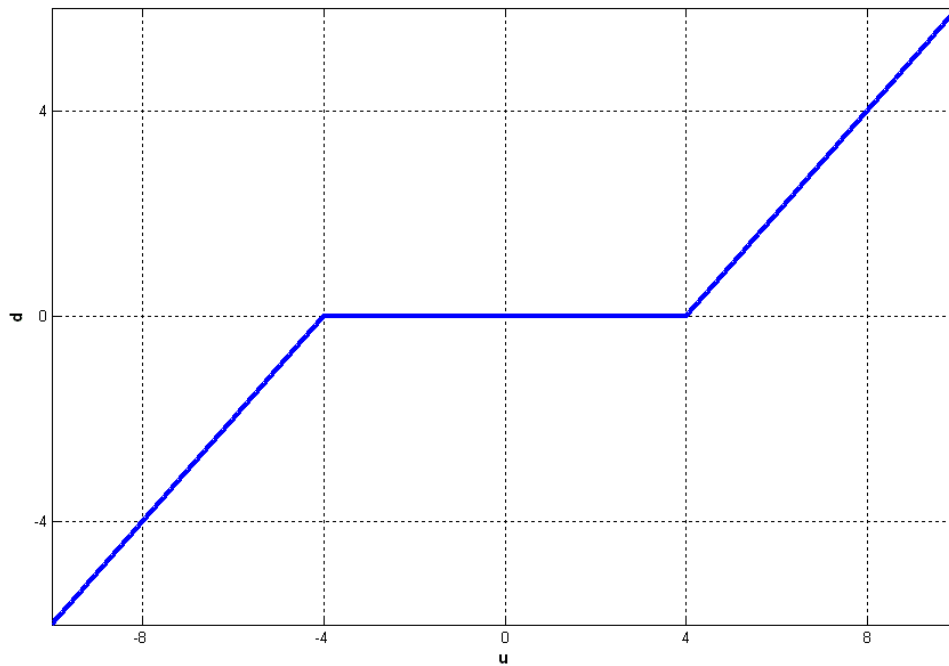


Figura II.3 – Gráfico da não-linearidade de zona morta tipo II desejada.

Os códigos MATLAB para geração do conjunto de treinamento para as simulações de zona morta tipos I e II estão nos Apêndices 2 e 3, respectivamente.

– Saturação com zona morta

Nesta simulação ocorrem os efeitos de saturação e zona morta, de acordo com o estipulado pela eq. II.5, representada na Figura II.4.

$$d = \begin{cases} -1 & \text{se } u < -6 \\ \frac{u}{3} + 1 & \text{se } -6 \leq u \leq -3 \\ 0 & \text{se } -3 < u < 3 \\ \frac{u}{3} - 1 & \text{se } 3 \leq u \leq 6 \\ 1 & \text{se } u > 6 \end{cases} \quad \text{eq. II.5}$$

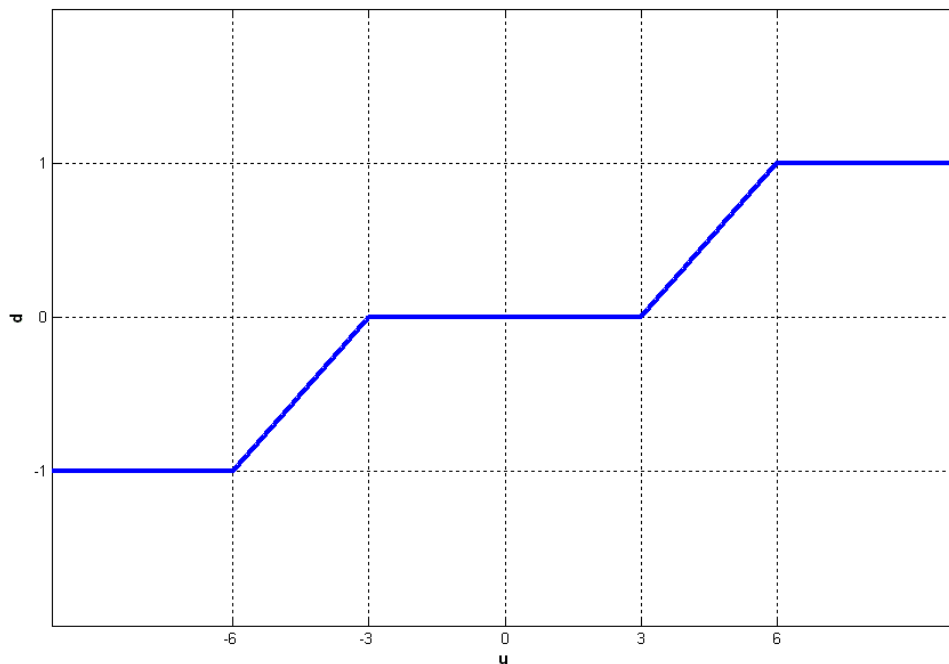


Figura II.4 – Gráfico da não-linearidade de saturação com zona morta desejada.

A programação MATLAB para a geração do conjunto de treinamento da não-linearidade de saturação com zona morta está disposta no Apêndice 4.

– Liga-desliga

A não-linearidade liga-desliga tem modelagem bastante simples, conforme eq. II.6, representada pela Figura II.5. O código ora implementado, constante do Apêndice 5, simplifica a saída em -1 quando as entradas são nulas.

$$d = \begin{cases} 1 & \text{se } u > 0 \\ -1 & \text{se } u \leq 0 \end{cases} \quad \text{eq. II.6}$$

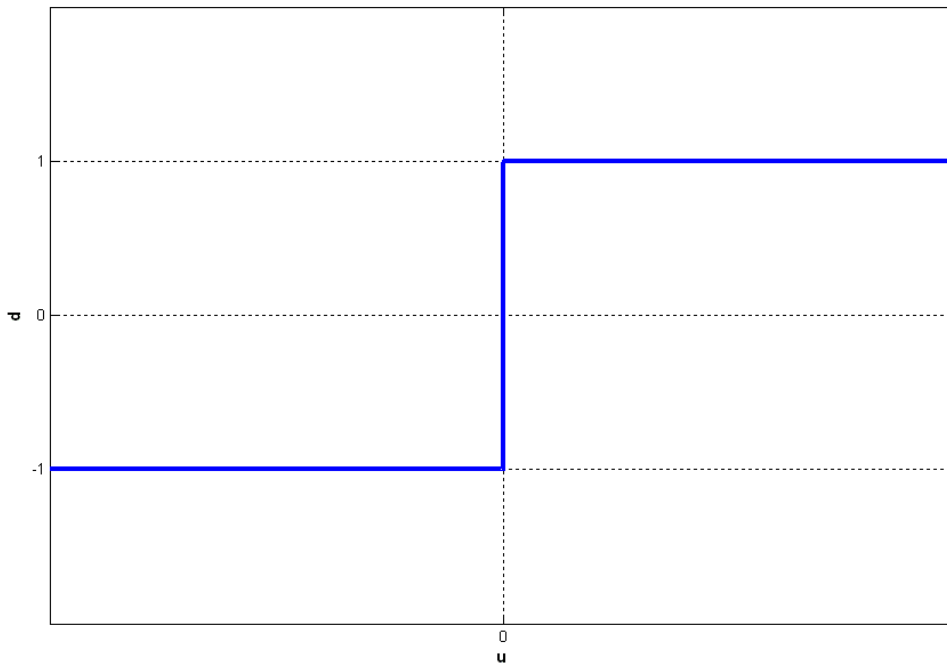


Figura II-5 – Gráfico da não-linearidade de liga-desliga desejada.

### – Liga-desliga com histerese

Para simular um controlador implementado com a não-linearidade de liga-desliga com histerese, esta é introduzida para evitar chaveamentos espúrios mediante as seguintes regras:

1. Os sinais de entrada variam no intervalo  $[-10;10]$ , aproximadamente;
2. Caso o sinal de entrada seja menor que -8 (limiar inferior) e a saída anterior  $d(t-1)$  esteja em 1, o controlador alterna a saída para -1;
3. O controlador oferece a saída em 1 somente quando o sinal de entrada ultrapassar o valor 8 (limiar superior) e a saída anterior  $d(t-1)$  estiver em -1.

Na ausência das duas últimas condições, o sinal de saída corrente permanece inalterado, isto é,  $d(t) = d(t-1)$ . A Figura II.6 ilustra a ação que se pretende com um controlador configurado com a não-linearidade de liga-desliga com histerese nestes parâmetros.

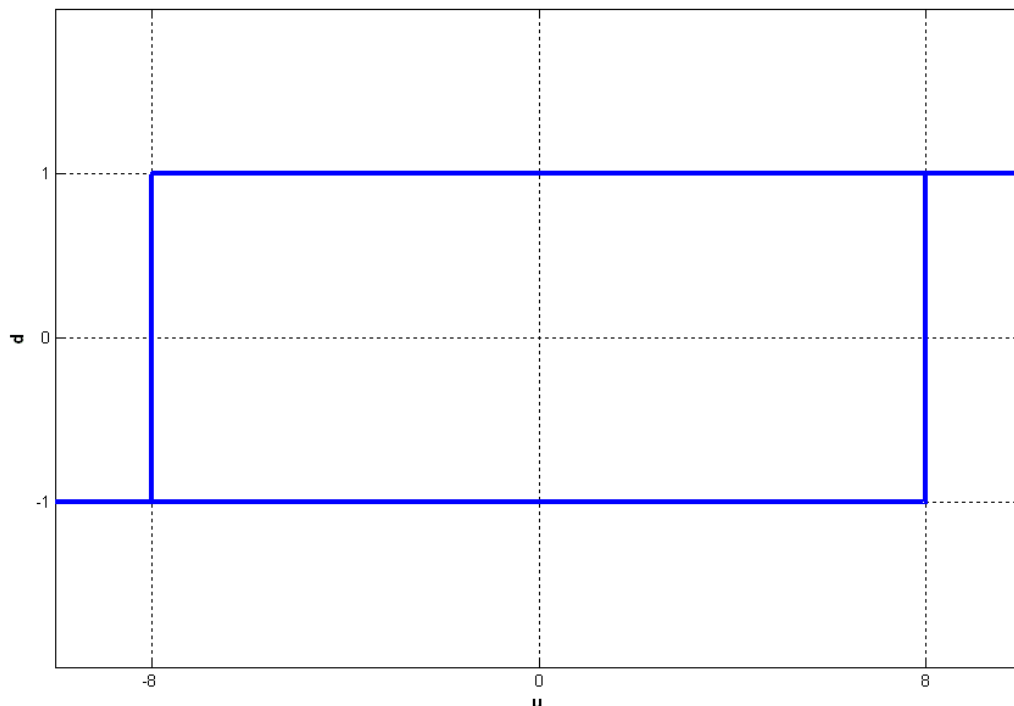


Figura II.6 – Gráfico da não-linearidade de liga-desliga com histerese desejado.

Para efeitos de programação, foi estipulado que o controlador inicia em 1. Observa-se que para calcular a saída são necessárias duas entradas, sendo o sinal de entrada corrente e o sinal de saída do controlador atrasado em uma iteração. O código MATLAB para gerar os padrões entrada-saída está disposto no Apêndice 6.

– Liga-desliga com zona morta

Esta modelagem, representada na Figura II.7, refere-se à não-linearidade de liga-desliga com inclusão de uma faixa de zona morta, compreendida no intervalo  $[-3;3]$ . A eq. II.7 define a ocorrência destes efeitos. O Apêndice 7 apresenta a codificação necessária para formar o conjunto de padrões entrada-saída deste efeito não-linear.

$$d = \begin{cases} -1 & \text{se } u < -3 \\ 0 & \text{se } -3 \leq u \leq 3 \\ 1 & \text{se } u > 3 \end{cases} \quad \text{eq. II.7}$$

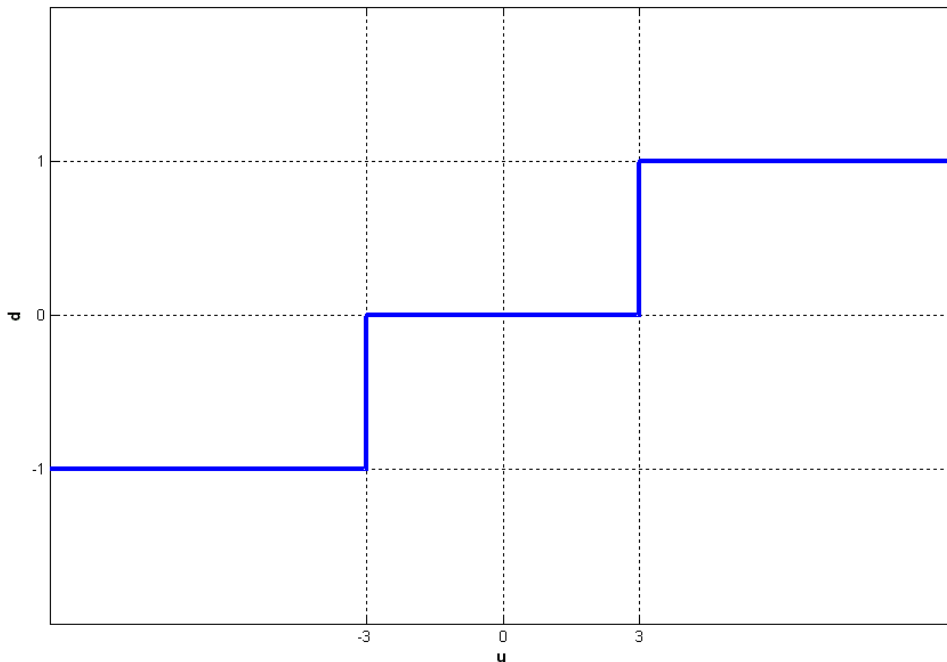


Figura II.7 – Gráfico da não-linearidade de liga-desliga com zona morta desejado.

### II.2.7.3– Normalização do conjunto de treinamento

De acordo com o discutido na seção I.2.8.7, todos os sinais de entrada e saída são submetidos à normalização, sendo utilizado o critério de valor mínimo -1 e valor máximo 1 (eq. I.47).

## II.3– METODOLOGIA DE SIMULAÇÃO

### II.3.1– Geração de padrões de entrada-saída

Para compor o conjunto de padrões entrada-saída foram criados sete códigos-fonte em MATLAB, um para cada não-linearidade estudada. Os referidos códigos constam dos Apêndices 1 a 7.

A execução do código para geração de padrões faz-se na linha de comando do MATLAB (Figura II.8) e depende da introdução do número de iterações e da taxa de amostragem, cujos parâmetros servem para aproximar o sinal de entrada ao sistema que se quer representar. Outrossim, de acordo com os valores atribuídos a estes parâmetros, pode haver repetência indesejada de padrões entrada-saída no conjunto de treinamento. Nesta condição, o código faz uma crítica na qual se sugere manter ou alterar tais valores.

Quando o processo termina, os sinais de entrada e saída são salvos em dois arquivos distintos, denominados “P.mat” e “T.mat”, respectivamente. Neste momento, recomenda-se armazená-los em pastas de arquivos distintas, a fim de evitar a sobreposição acidental destes, o que pode prejudicar a etapa de simulação da RNA.



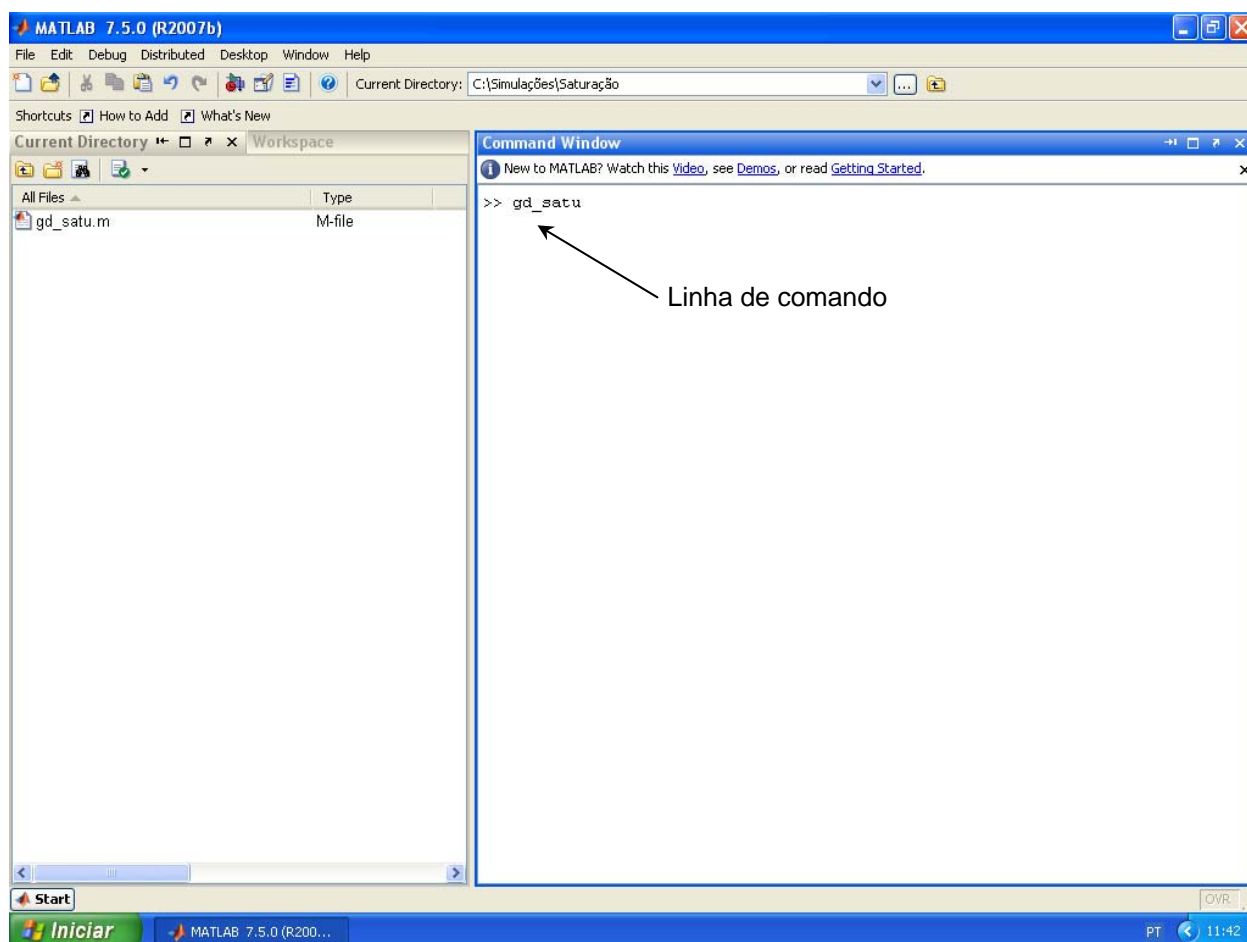


Figura II.8 – Tela do MATLAB (janela de comandos).

### II.3.2– Simulação da RNA

As simulações da rede neural estão baseadas no código MATLAB, disponibilizado em [47], tendo sido providenciadas as adaptações necessárias ao propósito deste trabalho. As alterações ora implementadas estão discriminadas ao final desta seção.

Uma vez concluída a geração de padrões entrada-saída para treinamento, a simulação da RNA inicia-se por meio da execução do arquivo MATLAB referente ao código-fonte disposto no Apêndice 8.

As simulações estão organizadas em duas etapas. Na primeira fase, a rede neural é avaliada entorno de uma configuração padrão, conforme consta na Tabela II.1.

Tabela II.1 – Configuração padrão da RNA.

Nº de neurônios da camada de entrada:	1 neurônio, para todas as não-linearidades, exceto liga-desliga com histerese, que possui 2 neurônios na referida camada.
Nº de neurônios da camada escondida:	3 neurônios para todas as não-linearidades, exceto liga-desliga com histerese, que possui 5 neurônios na referida camada.
Nº de neurônios da camada de saída:	1 neurônio, para todos os casos.
Nº de pares de entrada-saída do conjunto de treinamento:	10001 pares para todas as não-linearidades, exceto liga-desliga com histerese, que possui 21001 pares no referido conjunto.
Valor dos pesos e bias iniciais:	Aleatórios, entre -0.1 e 0.1.
Função de ativação da camada escondida:	Tangente hiperbólica.
Função de ativação da camada de saída:	Tangente hiperbólica.
Taxa de aprendizado:	0.1
Constante de momento:	0.75
Número de épocas:	100
CrITÉrios de parada:	(1) Nº máximo de épocas; (2) EQM de teste $\leq 0.001$ três vezes consecutivas; (3) EQM de teste com tendência de crescimento por três vezes consecutivas.

Para que se permita avaliar a conformidade do número de neurônios da camada escondida bem como o tamanho do conjunto de treinamento, na busca da configuração que melhor atenda o EQM de generalização estimado, são realizadas mais cinco simulações, com valores diversos dos pré-estabelecidos. Os resultados obtidos são organizados segundo a disposição da tabela constante do Apêndice 9, onde são dispostas as seis simulações e que tem a seguinte ordenação: número da simulação; número de padrões utilizados para treinamento e teste, em ordem crescente; e, para conjuntos de treinamento coincidentes, número de neurônios na camada escondida em ordem crescente.

As simulações N<sup>os</sup> 1, 3 e 6 têm mesmo número de neurônios na camada escondida, mas diferem sobre o tamanho do vetor de treinamento. Deste modo, a simulação N<sup>o</sup> 1 pode averiguar a capacidade de convergência da rede diante de um conjunto de treinamento reduzido a 50% da quantidade estimada. E, por sua vez, a simulação N<sup>o</sup> 6 possibilita inferir sobre a generalização da rede, quando esta é treinada com um conjunto de treinamento 100% maior que o determinado.

As simulações N<sup>os</sup> 2, 3, 4 e 5 são diferenciadas pelo número de neurônios da camada escondida, sendo que as de N<sup>os</sup> 2 e 4 mantêm valores imediatamente próximos à configuração padrão e servem para avaliar se o número de neurônios é adequado para cumprir o mapeamento não-linear da rede, por meio da análise comparativa dos EQMs de generalização obtidos. A simulação N<sup>o</sup> 5 permite medir os efeitos do incremento de parâmetros livres da rede, provocado pelo número superestimado de neurônios da camada escondida.

A segunda etapa refere-se às simulações complementares, através das quais se pretende atingir o EQM de generalização de 0.001 quando não for possível obtê-lo na primeira etapa. Estas se concentram no ajuste dos parâmetros de taxa de aprendizagem, constante de momento, número de pares entrada-saída para treinamento e número de épocas, que podem ser implementados isolados ou concorrentemente. O número de neurônios da camada escondida é definido consoante a configuração que apresentou melhor resultado de generalização na primeira fase. Os ajustamentos são empíricos, tendo como critérios de avaliação o valor do EQM de generalização obtido e o comportamento da curva de EQM de treinamento. No caso de haver a segunda etapa de simulações, os respectivos dados são dispostos em uma tabela complementar, cuja configuração responde à especificidade de cada caso.

Cada simulação produz ainda um relatório com os parâmetros utilizados e resultados obtidos. O Apêndice 10 contém um exemplo desta exposição, referente à simulação Nº 1 da não-linearidade de saturação. Também são disponibilizados quatro códigos para geração de gráficos, disponíveis nos Apêndices 11 a 14.

Quanto às alterações do código original, as principais alterações são:

- Inclusão de *menu* para definição de parâmetros;
- Normalização baseada na valor mínimo -1 e valor máximo 1;
- Número de neurônios da camada intermediária definido, por padrão, segundo o teorema de Kolmogorov;
- Utilização da função de ativação tangente hiperbólica nas camadas escondida e de saída;
- Utilização de bias positivo;
- Inclusão das variáveis WWL e MML para armazenar a evolução dos pesos e *bias* das camadas intermediária e de saída, respectivamente, referentes à última época de treinamento;
- Inclusão das variáveis WWLE e MMLE para armazenar os pesos e *bias* das camadas intermediária e de saída, respectivamente, obtidos ao término de cada época de treinamento;
- Inclusão das variáveis T1Des, T1Sai, T2Des, T2Sai, GDes e GSai para armazenar os seguintes sinais, na ordem:
  - Sinal desejado emitido durante a fase de treinamento;
  - Sinal da rede emitido durante a fase de treinamento;
  - Sinal desejado emitido durante a fase de teste;
  - Sinal da rede emitido durante a fase de teste;

- Sinal desejado emitido durante a fase de generalização;
- Sinal da rede emitido durante a fase de generalização;
- Estabelecimento de três critérios de parada, sendo:
  - Número máximo de épocas;
  - EQM de teste inferior a 0.001 três vezes consecutivas;
  - EQM de teste com tendência de crescimento em três vezes consecutivas;
- Inclusão de cronômetro para medir o tempo de cada simulação;
- Emissão de relatório com parâmetros de configuração e resultados obtidos;
- Opção de salvar todos os dados de uma simulação em arquivo único.

## CAPÍTULO III

### RESULTADOS OBTIDOS

A seguir são apresentados os resultados obtidos pelas redes neurais artificiais na identificação das não-linearidades examinadas nesta dissertação.

#### III.1– SIMULAÇÕES PADRONIZADAS

##### III.1.1– Simulação de saturação

Na Tabela III.1 são apresentados os resultados obtidos para a simulação de saturação.

Tabela III.1 – Resultados obtidos para a simulação de saturação.

Nº DA SIMULAÇÃO	Nº DE PADRÕES	Nº DE NEURÔNIOS POR CAMADA	Nº DE ÉPOCAS	CRITÉRIO DE PARADA	EQM DE TREINAMENTO	EQM DE TESTE	EQM DE GENERALIZAÇÃO	TEMPO DA SIMULAÇÃO
1	5001	1:3:1	20	2	0.000105666	0.000127878	0.000114124	00m 07s
2	10001	1:2:1	15	2	0.000138587	0.000139774	0.000139712	00m 11s
3	10001	1:3:1	15	2	0.000083085	0.000111747	0.000075876	00m 12s
4	10001	1:4:1	20	2	0.000076562	0.000089437	0.000070353	00m 16s
5	10001	1:10:1	100	1	0.001511943	0.001398790	0.001531720	01m 12s
6	20001	1:3:1	15	2	0.000039356	0.000060255	0.000035169	00m 30s

De acordo com os dados exibidos na Tabela III.1, tem-se as seguintes análises mais relevantes:

- Todas as simulações, exceto a de N<sup>o</sup> 5, obtiveram EQM de generalização inferior a 0.001;
- A simulação N<sup>o</sup> 6 obteve o menor EQM de generalização;
- As simulações N<sup>os</sup> 1, 2, 3, 4 e 6 alcançaram o EQM de generalização desejado antes do número máximo de épocas;
- A simulação N<sup>o</sup> 5 apresentou o maior EQM de generalização, tendo utilizado cem épocas de treinamento.

A Figura III.1 exibe a evolução da curva do EQM de treinamento e as indicações dos EQMs de teste, referentes à simulação N<sup>o</sup> 6 para a não-linearidade de saturação.

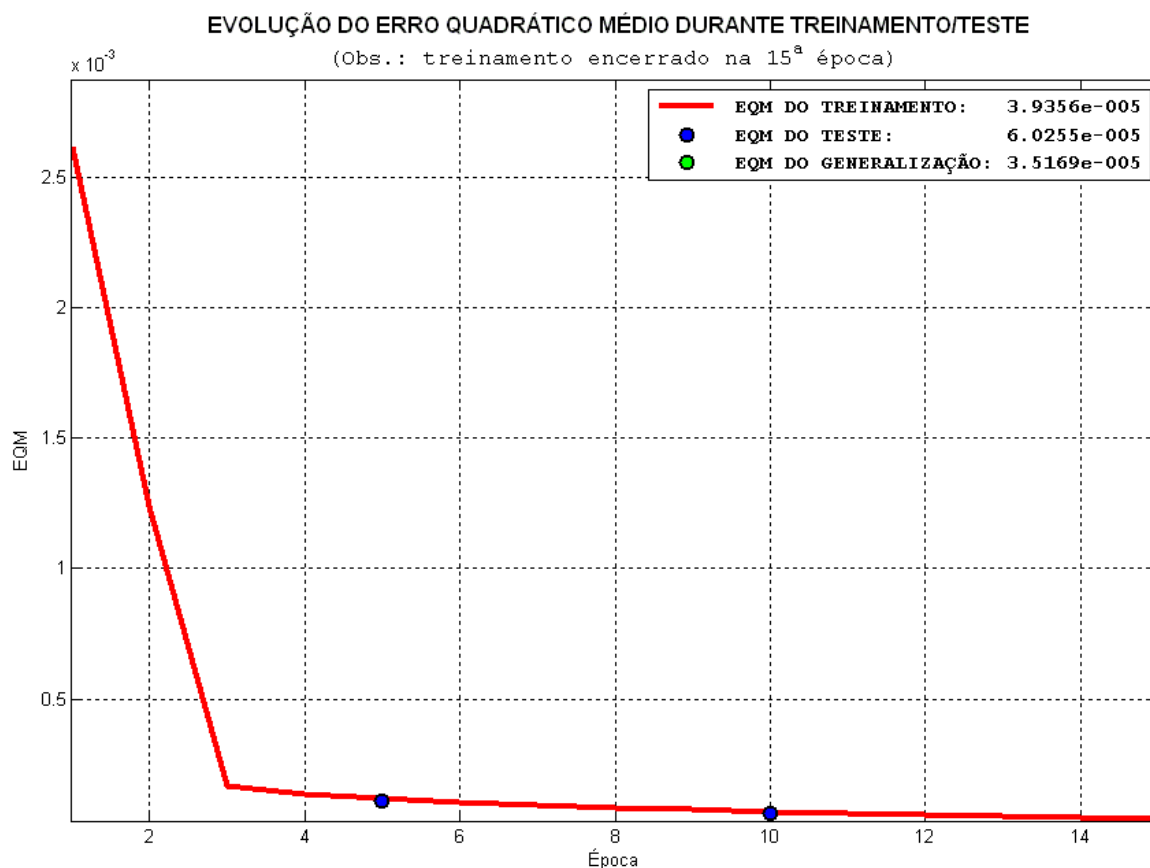


Figura III.1 – EQMs obtidos para a simulação N<sup>o</sup> 6 de saturação.

Conforme o gráfico da Figura III.1, o treinamento foi encerrado na 15ª época, tendo o EQM de teste obtido valor inferior a 0.001 nas épocas 5, 10 e 15.

A Figura III.2 traz uma comparação entre os cem últimos sinais desejados com os emitidos pela RNA, atinentes à fase de generalização da simulação Nº 6 para a não-linearidade de saturação.

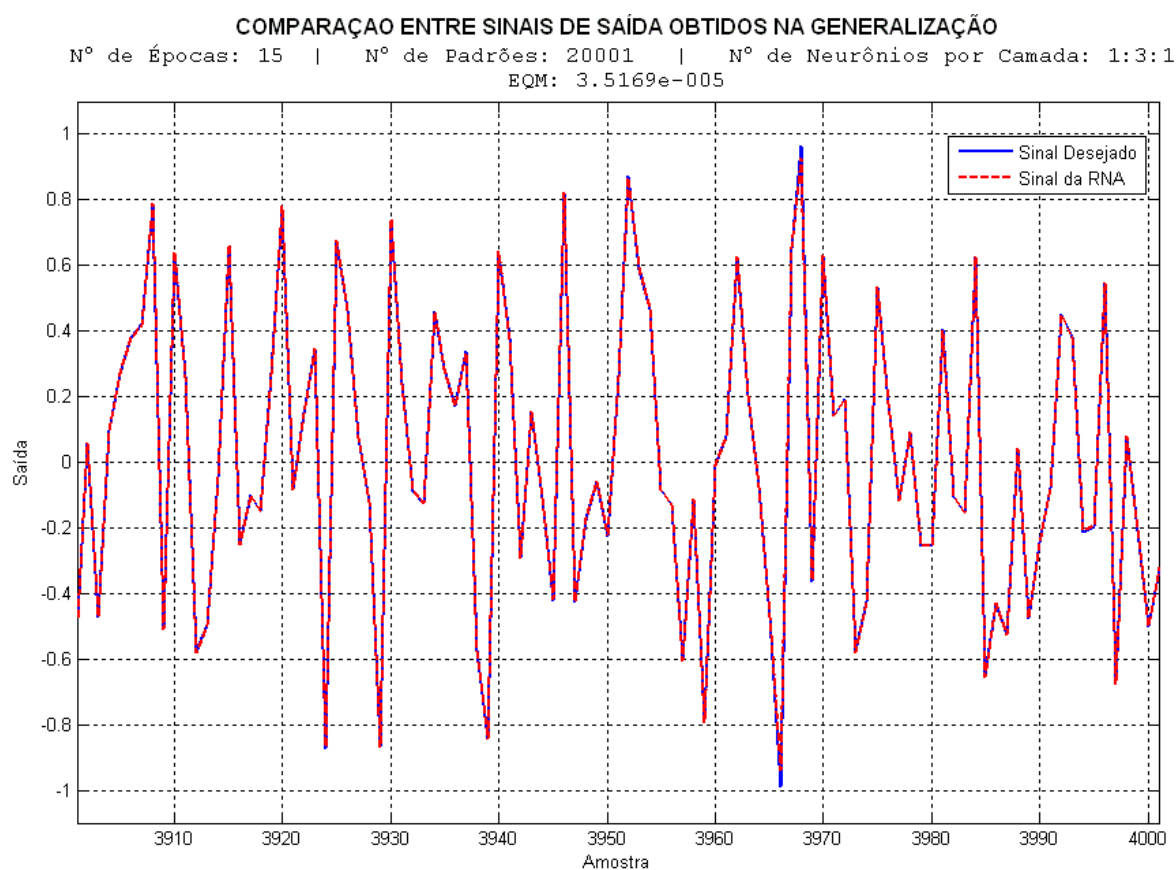


Figura III.2 – Amostra comparativa dos sinais desejados e emitidos pela RNA, referentes à fase de generalização da simulação Nº 6 de saturação.

A Figura III.2 traz uma amostragem que confere êxito à RNA em simular efeitos não-lineares de saturação, corroborada pelo EQM de generalização de 0.000035169.



### III.1.2– Simulação de zona morta tipo I

Na Tabela III.2 são apresentados os resultados obtidos para a simulação de zona morta tipo I.

Tabela III.2 – Resultados obtidos para a simulação de zona morta tipo I.

Nº DA SIMULAÇÃO	Nº DE PADRÕES	Nº DE NEURÓNIOS POR CAMADA	Nº DE ÉPOCAS	CRITÉRIO DE PARADA	EQM DE TREINAMENTO	EQM DE TESTE	EQM DE GENERALIZAÇÃO	TEMPO DA SIMULAÇÃO
1	5001	1:3:1	100	1	0.012080735	0.011901430	0.011888830	00m 34s
2	10001	1:2:1	100	1	0.002353159	0.002248263	0.002323055	01m 08s
3	10001	1:3:1	100	1	0.011861398	0.010677478	0.010668015	01m 11s
4	10001	1:4:1	100	1	0.011777902	0.009987692	0.010276191	01m 09s
5	10001	1:10:1	45	3	0.011393598	0.011604898	0.014309234	00m 32s
6	20001	1:3:1	100	1	0.011910927	0.010400083	0.010618362	02m 22s

De acordo com os dados exibidos na Tabela III.2, depreende-se o seguinte:

- O menor EQM de generalização foi obtido pela simulação Nº 2;
- A simulação Nº 5 apresentou o maior EQM de generalização, tendo o respectivo processo de treinamento encerrado na 45ª época, devido ao EQM de teste ter apresentado tendência de crescimento.
- O EQM de generalização da simulação Nº 6 foi similar àqueles conquistados pelas configurações Nºs 1, 3 e 4;
- Nenhuma configuração obteve EQM de generalização abaixo de 0.001;
- Todas as simulações, exceto Nº 5, esgotaram o número máximo de cem épocas de treinamento.

A Figura III.3 exibe a evolução da curva do EQM durante a fase de treinamento bem como os pontos dos EQMs de teste, referentes à simulação N<sup>o</sup> 2 de zona morta tipo I.

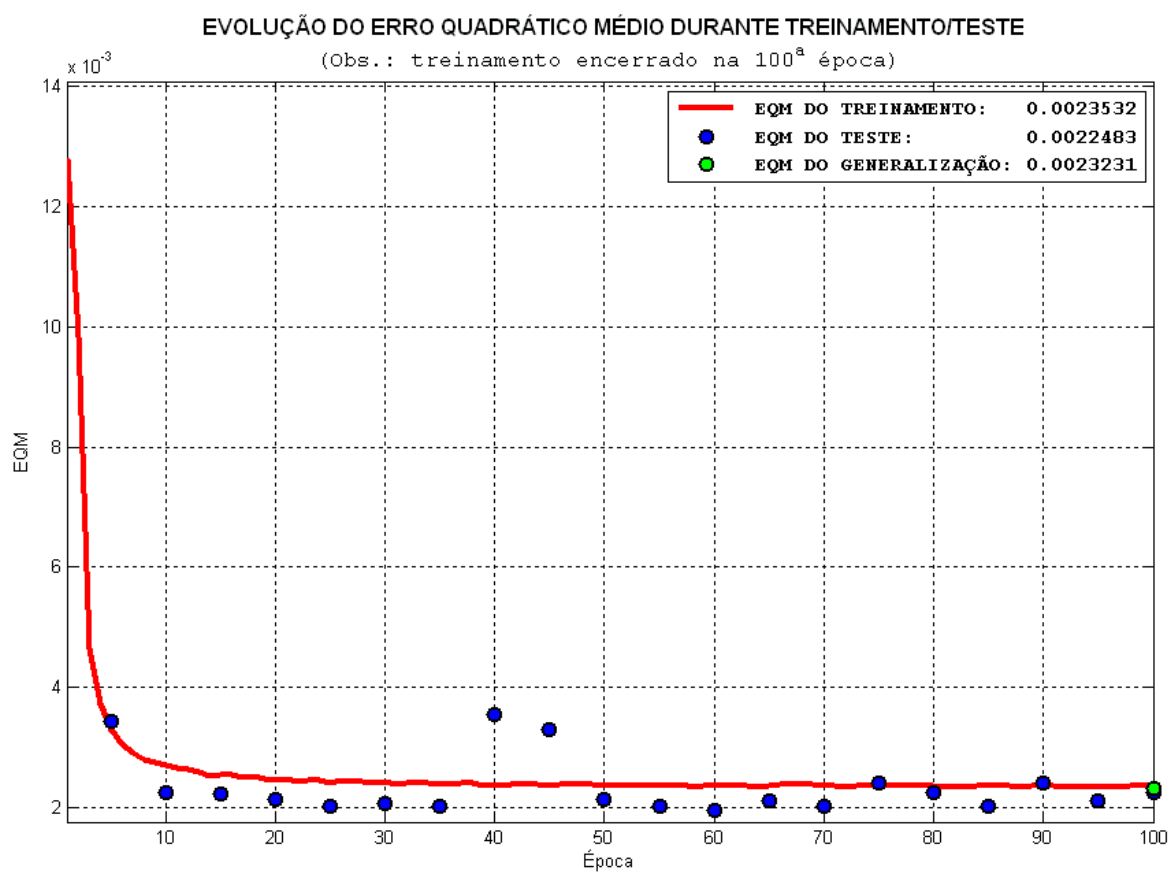


Figura III.3 – EQMs obtidos para a simulação N<sup>o</sup> 2 de zona morta tipo I.

De acordo com gráfico da Figura III.3, o EQM de treinamento estabilizou por volta da 50<sup>a</sup> época, com valores próximos aos obtidos pela fase de teste.

A Figura III.4 traz uma comparação entre os cem últimos sinais desejados com os emitidos pela RNA, gerados pela simulação N<sup>o</sup> 2 da não-linearidade zona morta tipo I.

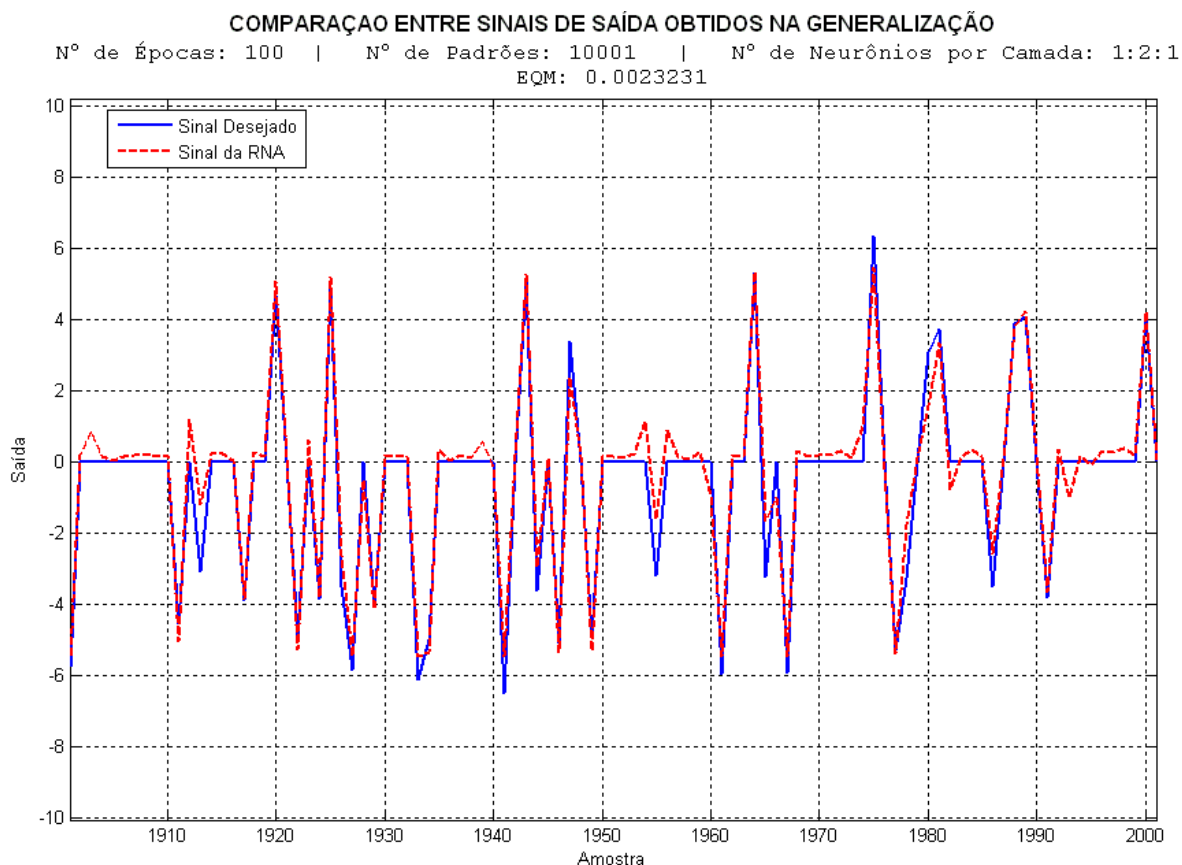


Figura III.4 – Amostra comparativa dos sinais desejados e emitidos pela RNA, referentes à fase de generalização da simulação N<sup>o</sup> 2 de zona morta tipo I.

A Figura III.4 representa uma amostragem que possibilita inferir sobre a generalização da rede em representar o efeito não-linear de zona morta tipo I para um EQM de 0.002323055.

### III.1.3– Simulação de zona morta tipo II

Na Tabela III.3 são apresentados os resultados obtidos para a simulação de zona morta tipo II.

Tabela III.3 – Resultados obtidos para a simulação de zona morta tipo II.

Nº DA SIMULAÇÃO	Nº DE PADRÕES	Nº DE NEURÔNIOS POR CAMADA	Nº DE ÉPOCAS	CRITÉRIO DE PARADA	EQM DE TREINAMENTO	EQM DE TESTE	EQM DE GENERALIZAÇÃO	TEMPO DA SIMULAÇÃO
1	5001	1:3:1	100	1	0.006348745	0.006244817	0.005793279	00m 34s
2	10001	1:2:1	100	1	0.006171926	0.005371927	0.005802724	01m 08s
3	10001	1:3:1	100	1	0.006343353	0.005549968	0.004899522	01m 08s
4	10001	1:4:1	100	1	0.006203985	0.005433464	0.005264311	01m 08s
5	10001	1:10:1	100	1	0.006281192	0.005997828	0.005276862	01m 11s
6	20001	1:3:1	100	1	0.006246597	0.005396097	0.005530600	02m 22s

Segundo as informações contidas na Tabela III.3, tem-se as seguintes conclusões:

- A simulação que obteve menor EQM de generalização é a de Nº 3;
- As simulações Nºs 1 e 6 têm, respectivamente, o menor e o maior conjunto de padrões entrada-saída para treinamento, mas a diferença entre seus EQMs de generalização é de apenas 0.02%;
- A simulação Nº 5 tem dez neurônios na camada escondida, enquanto a de Nº 4 tem quatro neurônios na mesma camada, porém, a diferença de EQM de generalização entre as mesmas é de 0.01%, aproximadamente;
- Nenhuma das configurações simuladas atingiu o EQM desejado de 0.001;
- Todas as simulações tiveram cem épocas de treinamento.

A Figura III.5 exibe a evolução da curva do EQM durante a fase de treinamento e as indicações dos EQMs de teste, referentes à simulação N<sup>o</sup> 2 da não-linearidade de zona morta tipo II.

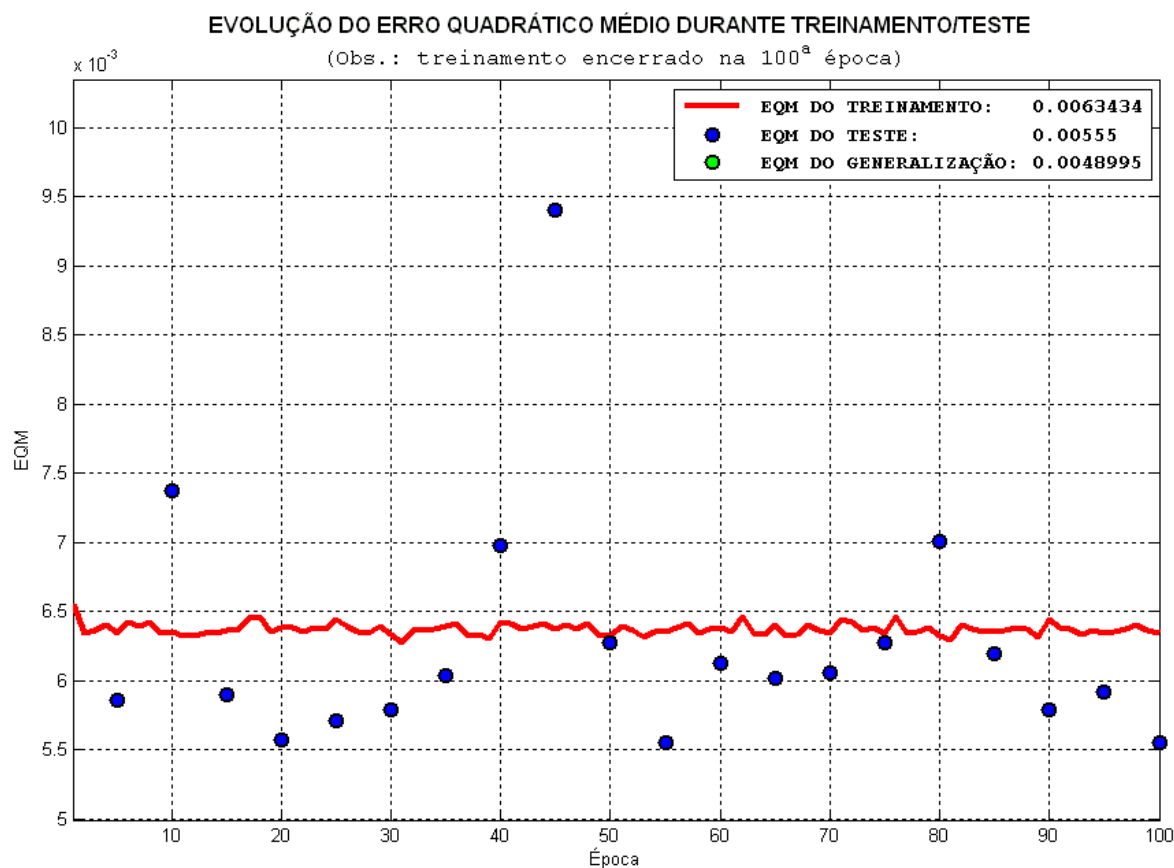


Figura III.5 – EQMs obtidos para a simulação N<sup>o</sup> 3 de zona morta tipo II.

A partir do gráfico exposto na Figura III.5, depreende-se que a curva do EQM de treinamento não estabilizou nem indicou tendência de aproximação para o valor estimado. Segundo o mesmo gráfico, o EQM de teste apresentou erro superior ao obtido pelo treinamento em quatro épocas distintas.

A Figura III.6 traz uma comparação entre os cem últimos sinais emitidos pela RNA com os sinais ideais da não-linearidade de zona morta tipo II, obtidos pela simulação Nº 3.

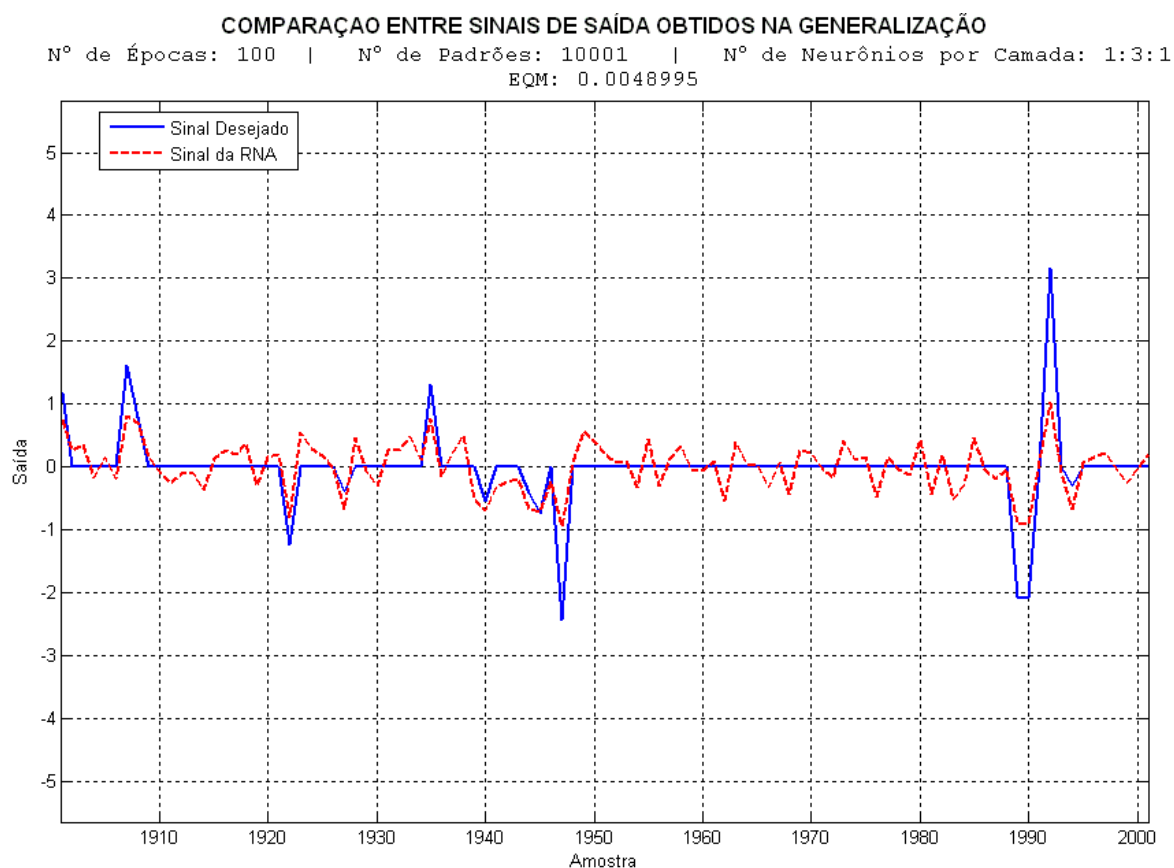


Figura III.6 – Amostra comparativa dos sinais desejados e emitidos pela RNA, referentes à fase de generalização da simulação Nº 3 de zona morta tipo II.

O gráfico da Figura III.6 representa uma amostra comparativa a partir da qual se permite deduzir que as respostas inadequadas da rede se concentram na representação da faixa de zona morta.

### III.1.4– Simulação de saturação com zona morta

Na Tabela III.4 são apresentados os resultados obtidos para a simulação de saturação com zona morta.

Tabela III.4 – Resultados obtidos para a simulação de saturação com zona morta.

Nº DA SIMULAÇÃO	Nº DE PADRÕES	Nº DE NEURÔNIOS POR CAMADA	Nº DE ÉPOCAS	CRITÉRIO DE PARADA	EQM DE TREINAMENTO	EQM DE TESTE	EQM DE GENERALIZAÇÃO	TEMPO DA SIMULAÇÃO
1	5001	1:3:1	100	1	0.021410535	0.020768642	0.020607376	00m 34s
2	10001	1:2:1	15	2	0.000496470	0.000459981	0.000654306	00m 11s
3	10001	1:3:1	100	1	0.022234762	0.020072046	0.018671704	01m 08s
4	10001	1:4:1	100	1	0.021869096	0.020446714	0.020217481	01m 09s
5	10001	1:10:1	100	1	0.021083807	0.022389113	0.022727655	01m 11s
6	20001	1:3:1	100	1	0.021787230	0.018776882	0.019602766	02m 20s

A partir da Tabela III.4, é possível extrair as seguintes observações:

- A simulação Nº 2 foi a única a atingir EQM abaixo de 0.001;
- A simulação Nº 5 deteve o maior EQM de generalização;
- Considerando-se o tamanho do conjunto de treinamento, as configurações Nºs 1, 3 e 6 obtiveram desempenho semelhante em termos de EQM de generalização obtidos;
- Todas as simulações, exceto Nº 2, consumiram cem épocas de treinamento.

A Figura III.7 demonstra a evolução da curva do EQM durante a fase de treinamento e pontua os EQMs de teste, referentes à simulação N<sup>o</sup> 2 da não-linearidade de saturação com zona morta.

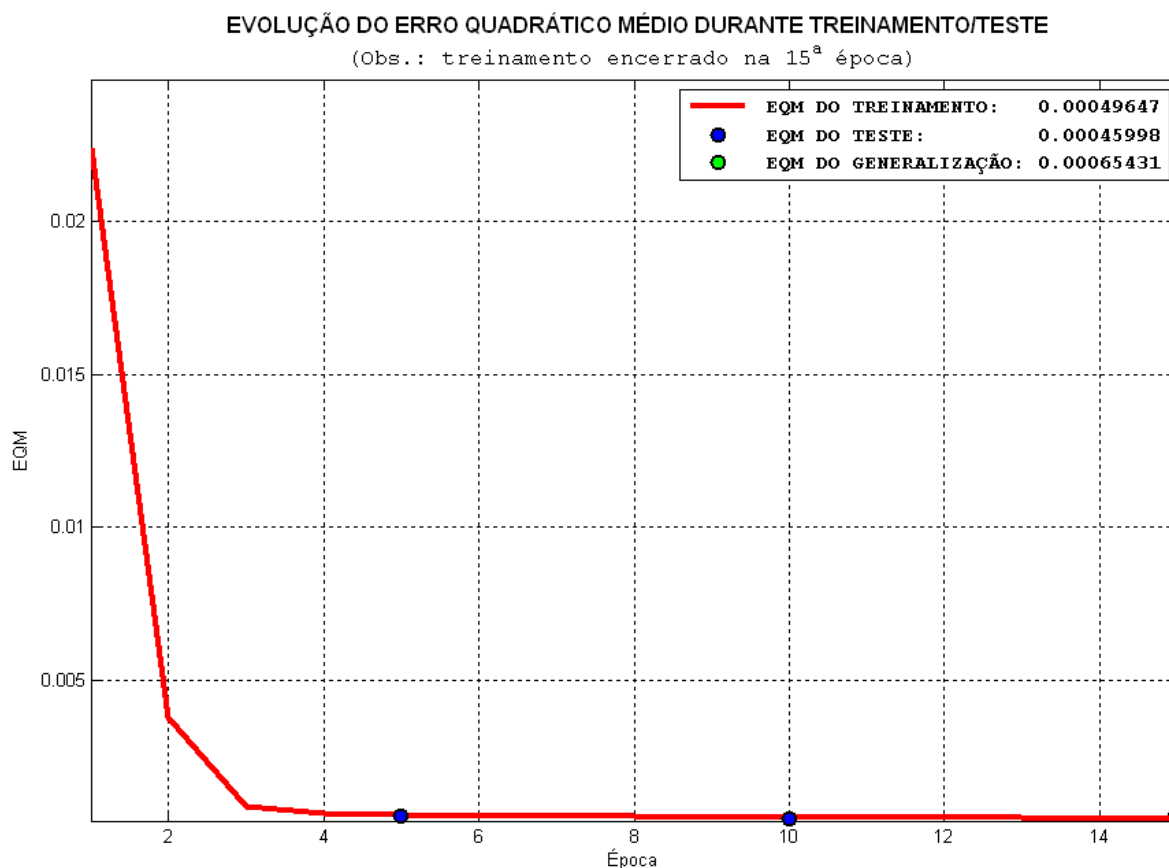


Figura III.7 – EQMs obtidos para a simulação N<sup>o</sup> 2 de saturação com zona morta.

Conforme o gráfico da Figura III.7, o EQM do treinamento estabilizou a partir da 8<sup>a</sup> época, tendo obtido valor abaixo de 0.001 já na 3<sup>a</sup> época. Os resultados obtidos pelo EQM de teste nas épocas 5, 10 e 15 possibilitaram antecipar o término do treinamento.



A Figura III.8 traz uma comparação entre os cem últimos sinais emitidos pela RNA com os sinais ideais da não-linearidade de saturação com zona morta, atinentes à simulação N<sup>o</sup> 2.

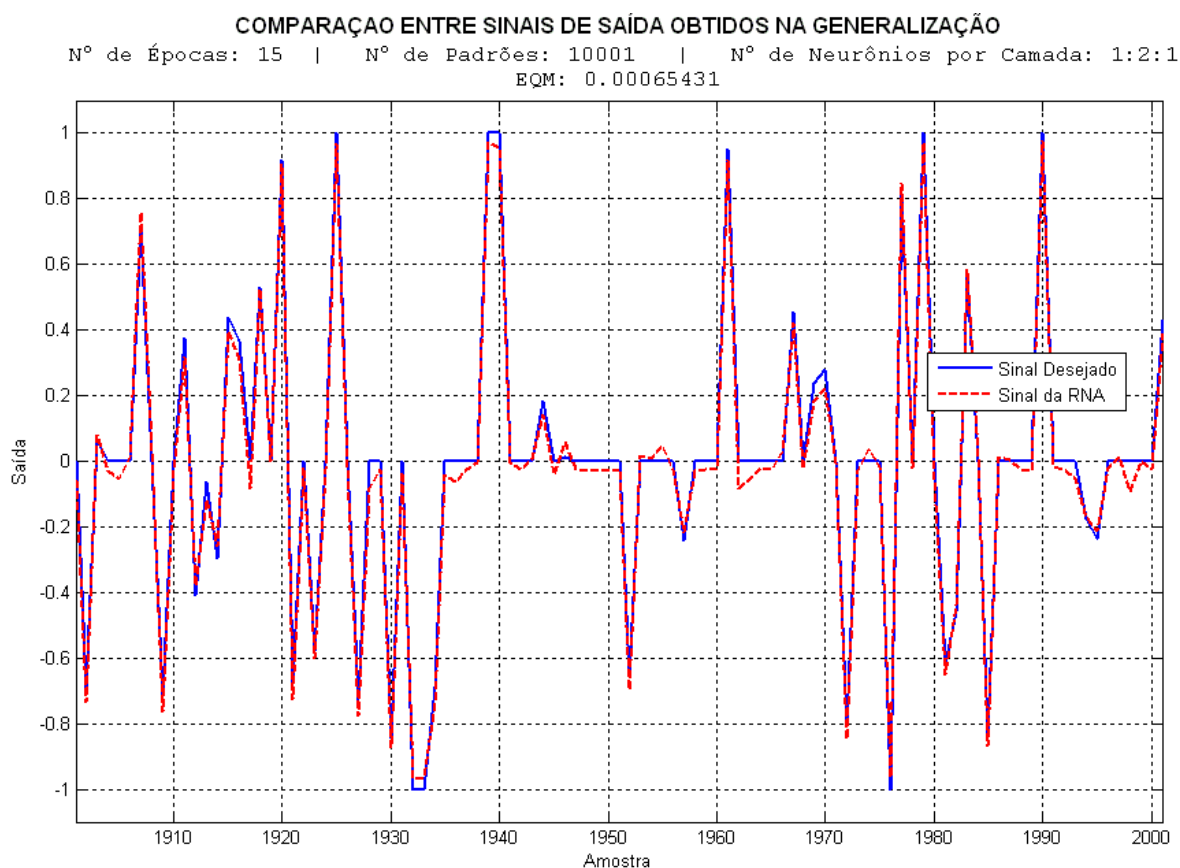


Figura III.8 – Amostra comparativa dos sinais desejados e emitidos pela RNA, referentes à fase de generalização da simulação N<sup>o</sup> 2 de saturação com zona morta.

Apesar do bom desempenho demonstrado pela Figura III.8, corroborado pelo EQM de generalização de 0.000654306, o sinal da rede não é suficientemente seguro para referenciais de zona morta.

### III.1.5– Simulação de liga-desliga

Na Tabela III.5 são apresentados os resultados obtidos para a simulação de liga-desliga.

Tabela III.5 – Resultados obtidos para a simulação de liga-desliga.

Nº DA SIMULAÇÃO	Nº DE PADRÕES	Nº DE NEURÔNIOS POR CAMADA	Nº DE ÉPOCAS	CRITÉRIO DE PARADA	EQM DE TREINAMENTO	EQM DE TESTE	EQM DE GENERALIZAÇÃO	TEMPO DA SIMULAÇÃO
1	5001	1:3:1	100	1	0.008261843	0.013372284	0.016728524	00m 34s
2	10001	1:2:1	100	1	0.007038208	0.002589874	0.003204018	01m 08s
3	10001	1:3:1	100	1	0.008508805	0.008728106	0.006363916	01m 08s
4	10001	1:4:1	100	1	0.008229589	0.007242052	0.008444819	01m 09s
5	10001	1:10:1	100	1	0.007668122	0.008498843	0.009846905	01m 12s
6	20001	1:3:1	100	1	0.006587737	0.011033060	0.009755371	02m 24s

De acordo com as informações contidas na Tabela III.5, é possível afirmar que:

- A simulação Nº 2 obteve o menor EQM de generalização;
- O maior EQM de generalização pertence à simulação Nº 1;
- As simulações Nºs 5 e 6, com o maior número de neurônios na camada escondida e o maior conjunto de exemplos para treinamento, respectivamente, apresentaram EQM de generalização superior aos obtidos pelas simulações Nºs 2, 3 e 4;
- Nenhuma configuração apresentou EQM de generalização inferior a 0.001;
- Todas as simulações utilizaram cem épocas de treinamento.

A evolução da curva de EQM de treinamento e os EQMs de teste, atinentes à simulação Nº 2 da não-linearidade de liga-desliga, estão dispostos na Figura III.9.

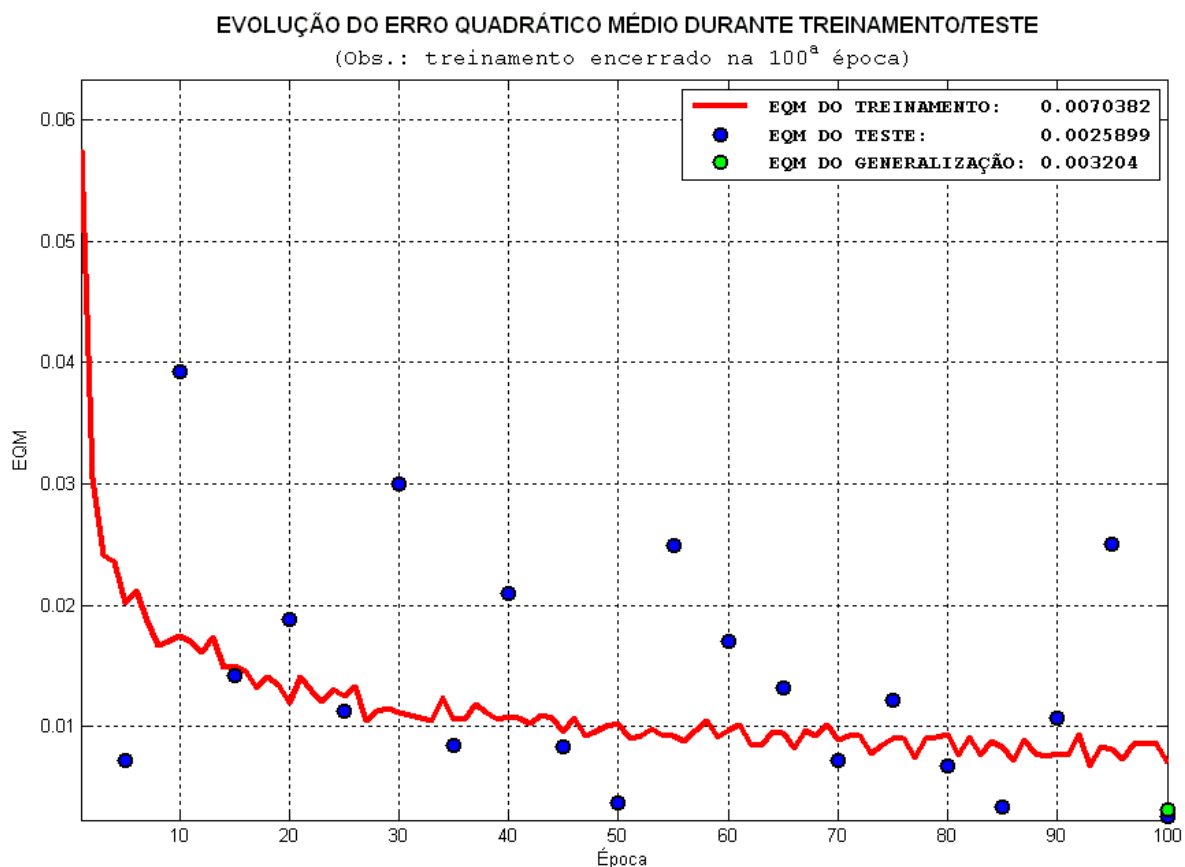


Figura III.9 – EQMs obtidos para a simulação Nº 2 de liga-desliga.

O curva do EQM de treinamento, disposta na Figura III.9, denota oscilação do processo de aprendizagem e número de épocas insuficiente para assegurar a convergência da rede. Segundo o próprio gráfico, 50% dos EQMs de teste tiveram valores acima dos obtidos pelo treinamento.

Uma amostra comparativa, entre os cem últimos sinais desejados e emitidos pela fase de generalização, referentes à simulação N<sup>o</sup> 2 da não-linearidade de liga-desliga, é demonstrada na Figura III.10.

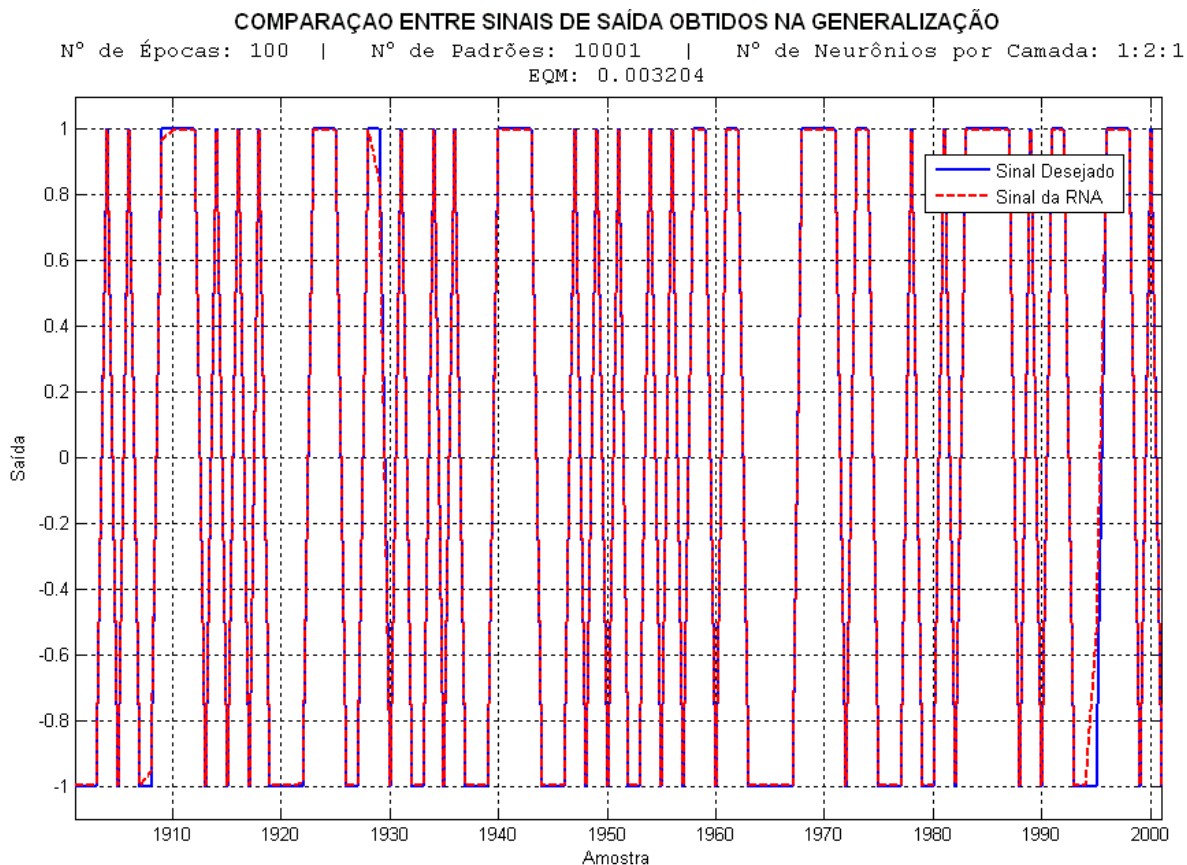


Figura III.10 – Amostra comparativa dos sinais desejados e emitidos pela RNA, referentes à fase de generalização da simulação N<sup>o</sup> 2 de liga-desliga.

### III.1.6– Simulação de liga-desliga com histerese

Na Tabela III.6 são apresentados os resultados obtidos para a simulação de liga-desliga com histerese.

Tabela III.6 – Resultados obtidos para a simulação de liga-desliga com histerese.

Nº DA SIMULAÇÃO	Nº DE PADRÕES	Nº DE NEURÓNIOS POR CAMADA	Nº DE ÉPOCAS	CRITÉRIO DE PARADA	EQM DE TREINAMENTO	EQM DE TESTE	EQM DE GENERALIZAÇÃO	TEMPO DA SIMULAÇÃO
1	5001	1:3:1	100	1	0.003403256	0.002166685	0.002436130	01m 14s
2	10001	1:2:1	100	1	0.002479428	0.002400555	0.002384270	02m 23s
3	10001	1:3:1	100	1	0.002583429	0.002252497	0.001534790	02m 32s
4	10001	1:4:1	100	1	0.002455037	0.002118496	0.002381125	02m 32s
5	10001	1:10:1	100	1	0.002092243	0.001809074	0.003130647	02m 40s
6	20001	1:3:1	100	1	0.001667690	0.002059011	0.001886361	05m 28s

De acordo com os resultados constantes da Tabela III.6, tem-se as seguintes considerações principais:

- A simulação Nº 3 tem a configuração que mais se aproximou do EQM de generalização esperado;
- A simulação Nº 5 obteve o maior EQM de generalização;
- A simulação Nº 1 atingiu EQM de generalização próximo aos das simulações Nºs 2 e 4;
- O segundo melhor EQM pertence à simulação Nº 6;
- Nenhuma das simulações alcançou EQM de 0.001;
- Todas as simulações utilizaram cem épocas de treinamento.

A Figura III.11 exibe a evolução da curva do EQM durante a fase de treinamento e marca os EQMs de teste referentes à simulação N<sup>o</sup> 3 para liga-desliga com histerese.

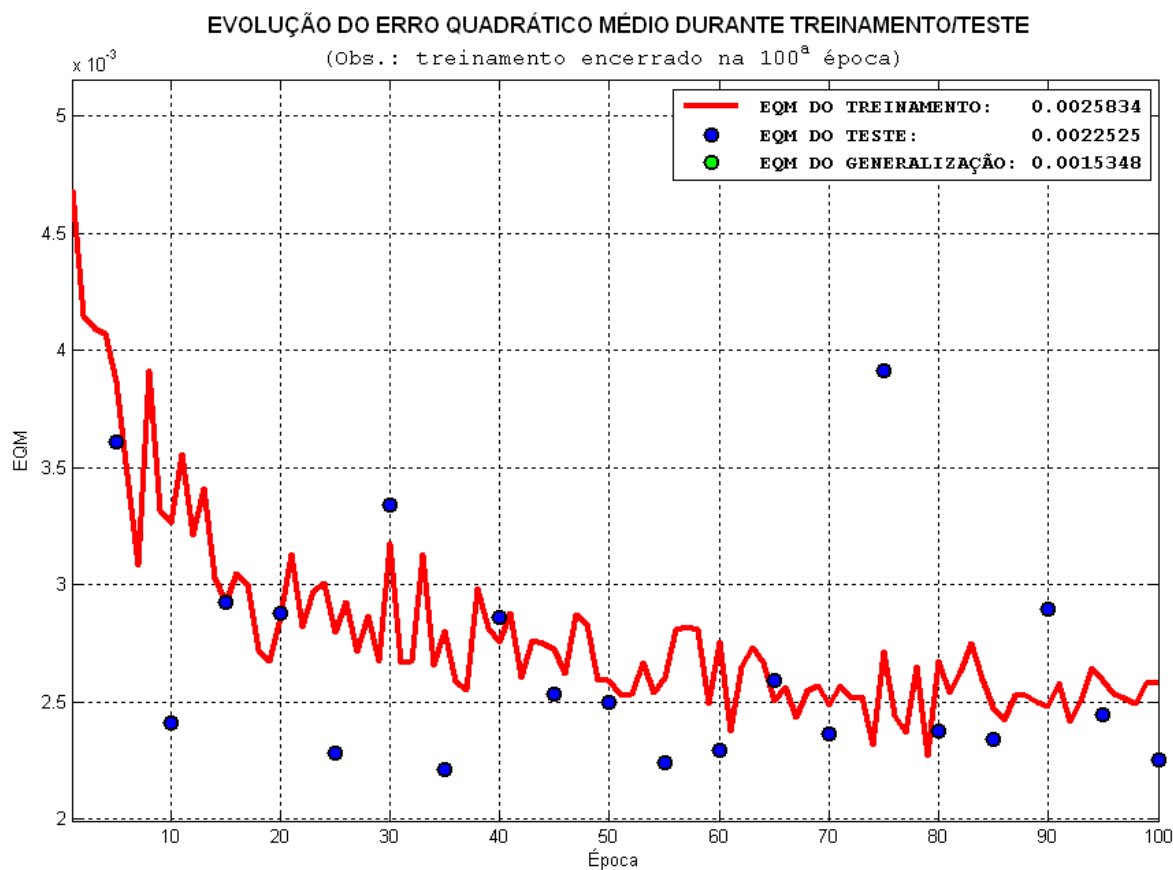


Figura III.11 – EQMs obtidos para a simulação N<sup>o</sup> 3 de liga-desliga com histerese.

De acordo com a Figura III.11, a convergência indefinida da rede está indefinida e o processo de treinamento oscilou para todas as épocas. Também é possível observar que 80% dos EQMs de teste apresentam valores menores que os alcançados pelo treinamento.

O gráfico da Figura III.12 exibe uma amostragem do desempenho da rede em representar o efeito não-linear de liga-desliga com histerese, contudo uma avaliação otimista é prejudicada pelo processo instável da aprendizagem aliado à indeterminação da convergência da rede.

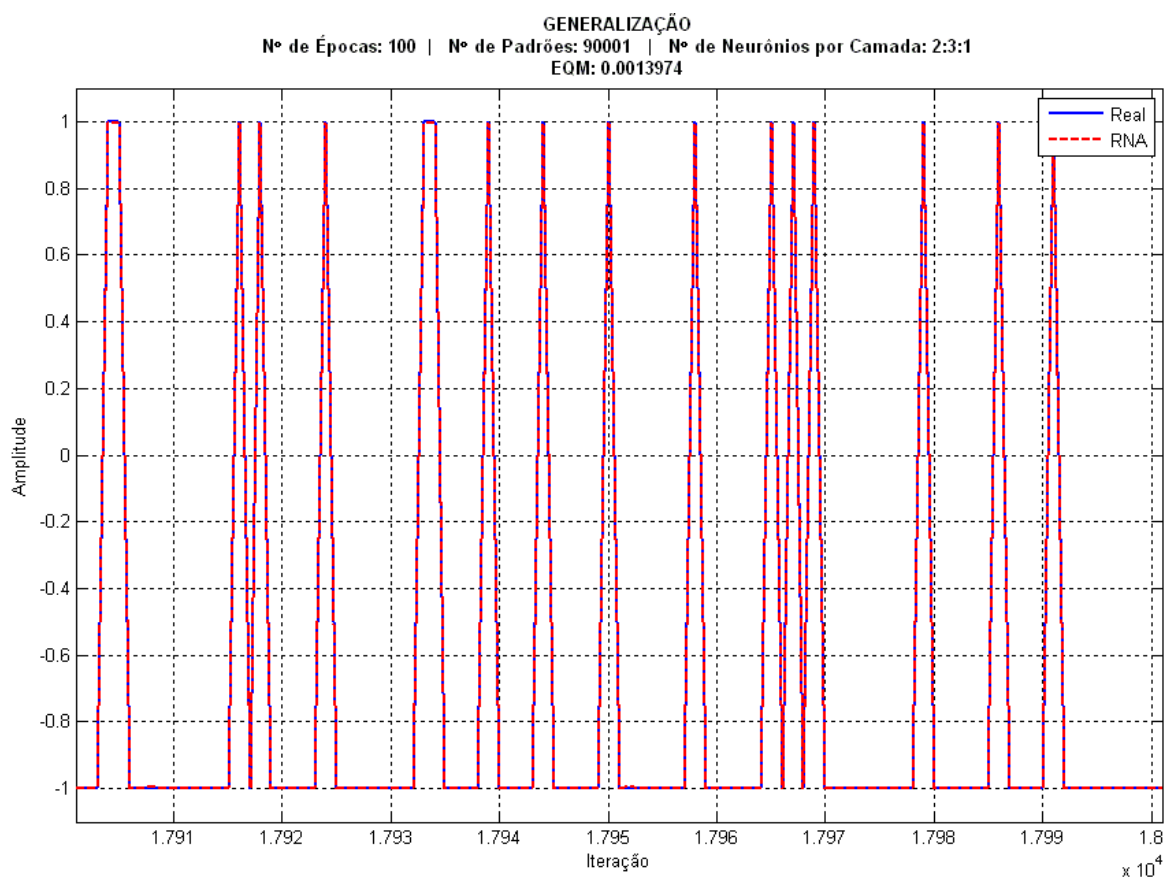


Figura III.12 – Amostra comparativa dos sinais desejados e emitidos pela RNA, referentes à fase de generalização da simulação N<sup>o</sup> 3 de liga-desliga com histerese.

### III.1.7– Simulação de liga-desliga com zona morta

Na Tabela III.7 são apresentados os resultados obtidos para a simulação de liga-desliga com zona morta.

Tabela III.7 – Resultados obtidos para a simulação de liga-desliga com zona morta.

Nº DA SIMULAÇÃO	Nº DE PADRÕES	Nº DE NEURÓNIOS POR CAMADA	Nº DE ÉPOCAS	CRITÉRIO DE PARADA	EQM DE TREINAMENTO	EQM DE TESTE	EQM DE GENERALIZAÇÃO	TEMPO DA SIMULAÇÃO
1	5001	1:3:1	100	1	0.003976188	0.003136050	0.003937831	00m 34s
2	10001	1:2:1	100	1	0.003557076	0.002596541	0.002577695	01m 07s
3	10001	1:3:1	100	1	0.003481406	0.006512117	0.006409846	01m 08s
4	10001	1:4:1	100	1	0.058428907	0.053885112	0.052121737	01m 08s
5	10001	1:10:1	90	3	0.056224091	0.063831756	0.066052688	01m 03s
6	20001	1:3:1	70	3	0.002993114	0.003434745	0.013503198	01m 41s

Segundo os resultados da Tabela III.7, pode-se deduzir que:

- A simulação que obteve o menor EQM de generalização é a de Nº 2;
- O maior EQM foi determinado pela simulação Nº 5, tendo sido o critério de parada estabelecido pela tendência de crescimento dos EQMs de teste nas épocas 80, 85 e 90;
- A simulação Nº 3 apresentou rendimento superior ao obtido pela simulação Nº 6;
- O critério de parada da simulação Nº 6 foi determinado pela tendência de crescimento dos EQMs de teste, ocorridos nas épocas 60, 65 e 70;
- As simulações Nºs 1, 2, 3 e 4 utilizaram cem épocas de treinamento.



A Figura III.13 exibe a evolução da curva do EQM durante a fase de treinamento junto com os EQMs de teste, referentes à simulação N° 2, para a não-linearidade de liga-desliga com zona morta.

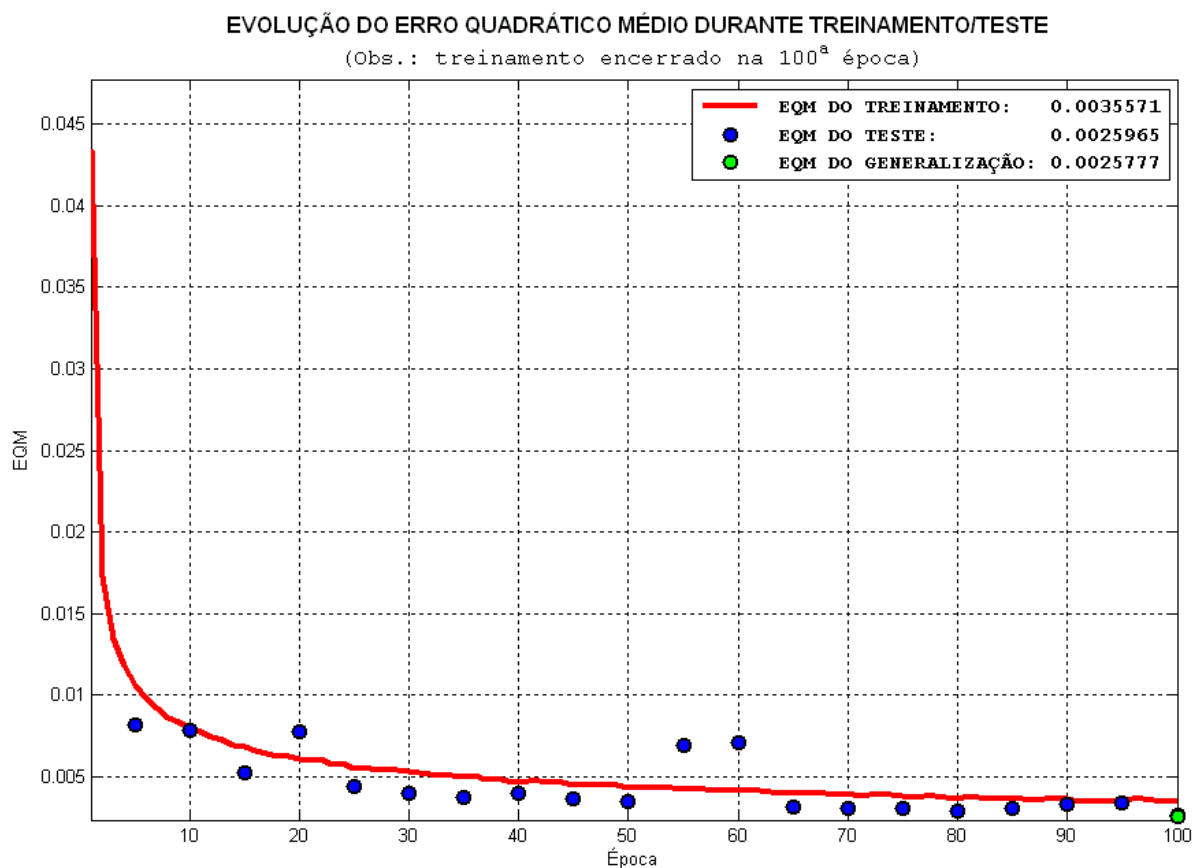


Figura III.13 – EQMs obtidos para a simulação N° 2 de liga-desliga com zona morta.

O gráfico da Figura III.13 demonstra que a utilização de cem épocas não foram suficientes para garantir a convergência da rede. De acordo com a mesmo gráfico, 85% dos EQMs de teste tiveram valores inferiores aos obtidos pelo treinamento.

A Figura III.14 apresenta uma comparação entre os cem últimos sinais desejados e emitidos pela RNA, referentes à simulação N° 2, da não-linearidade de liga-desliga com zona morta.

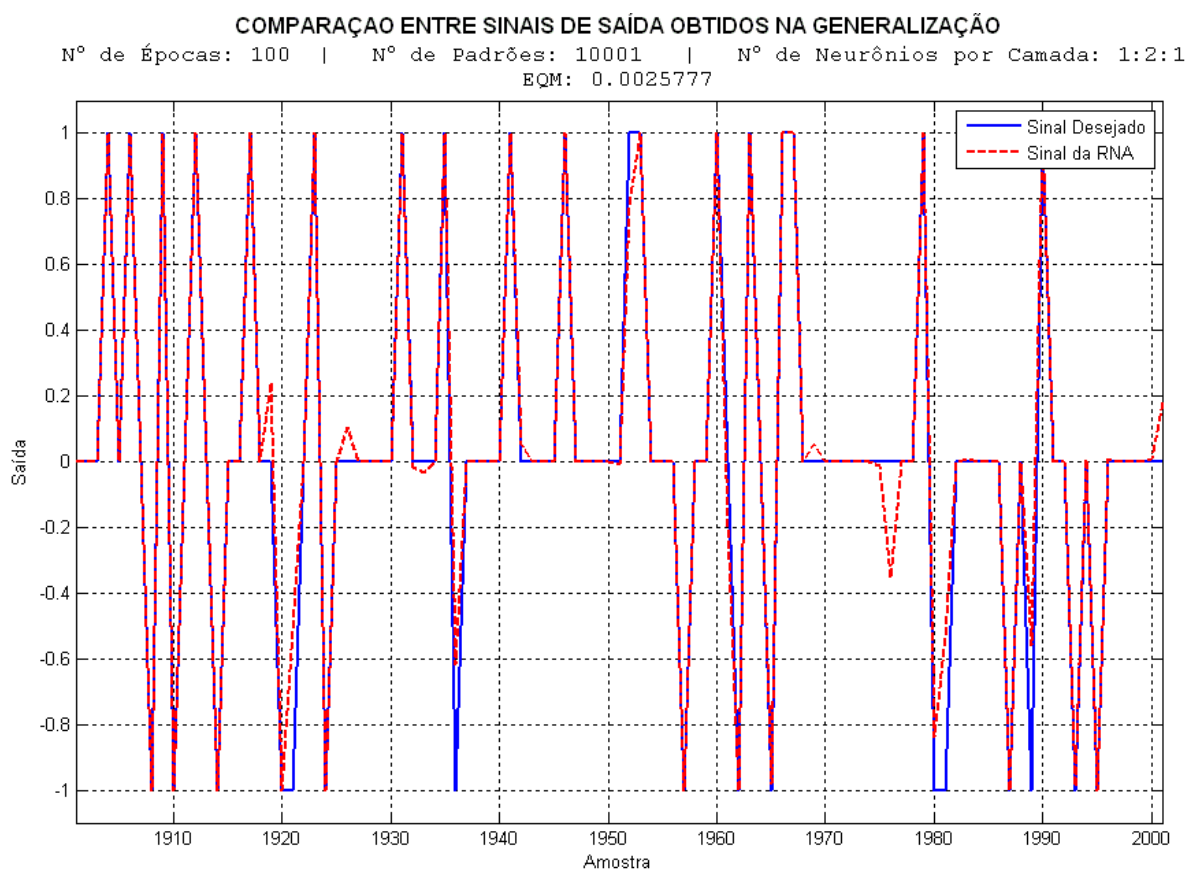


Figura III.14 – Amostra comparativa dos sinais desejados e emitidos pela RNA, referentes à fase de generalização da simulação N° 2 de liga-desliga com zona morta.

### III.2 – SIMULAÇÕES COMPLEMENTARES

Nesta seção são efetuadas simulações complementares para as não-linearidades de zona morta tipos I e II, liga-desliga, liga-desliga com histerese e liga-desliga com zona morta, por não terem cumprido o requisito de EQM de generalização mínimo.

### III.2.1– Simulação complementar de zona morta tipo I

Dentre as simulações de zona morta tipo I, apresentadas na seção III.1.2, a de N<sup>o</sup> 2 obteve o menor EQM de generalização, utilizando a topologia composta por dois neurônios na camada escondida, 10001 pares de entrada-saída e cem épocas de treinamento.

De acordo a evolução do EQM de treinamento obtida pela simulação N<sup>o</sup> 2 (Figura III.3), deduz-se que a rede estacionou na proximidade da 50<sup>a</sup> época sem superar o EQM de generalização pretendido. E mais, a amostra entre sinais (Figura III.4) apontou que a zona morta ainda não estava bem representada. Então, para asseverar o ponto de convergência da rede e atenuar a curva de aprendizagem da rede, as simulações N<sup>os</sup> 7 e 8 utilizaram taxa de aprendizagem igual a 0.01, constante de momento nula e demais parâmetros conforme Tabela III.8.

Tabela III.8 – Resultados obtidos para as simulações complementares de zona morta tipo I.

N <sup>o</sup> DA SIMULAÇÃO	N <sup>o</sup> DE PADRÕES	N <sup>o</sup> DE NEURÔNIOS POR CAMADA	N <sup>o</sup> DE ÉPOCAS	CRITÉRIO DE PARADA	EQM DE TREINAMENTO	EQM DE TESTE	EQM DE GENERALIZAÇÃO	TEMPO DA SIMULAÇÃO
7	10001	1:2:1	5000	1	0.001977087	0.001969669	0.001950141	60m 37s
8	20001	1:2:1	5000	1	0.001951001	0.001944437	0.001978951	109m 06s

Conforme os resultados apresentados na Tabela III.8, o melhor EQM foi obtido pela simulação N<sup>o</sup> 7, mas sem atingir o valor desejado. De acordo com a curva de evolução do EQM de treinamento apresentada na Figura III.15, a convergência da rede ocorreu a partir da 700<sup>a</sup> época (desvio-padrão de  $4 \times 10^{-6}$ ). Cabe ressaltar que a simulação N<sup>o</sup> 8 serve para ratificar a

quantidade de 10001 padrões do conjunto de treinamento, tendo em vista que o acréscimo de 100% de exemplos não melhorou o EQM de generalização.

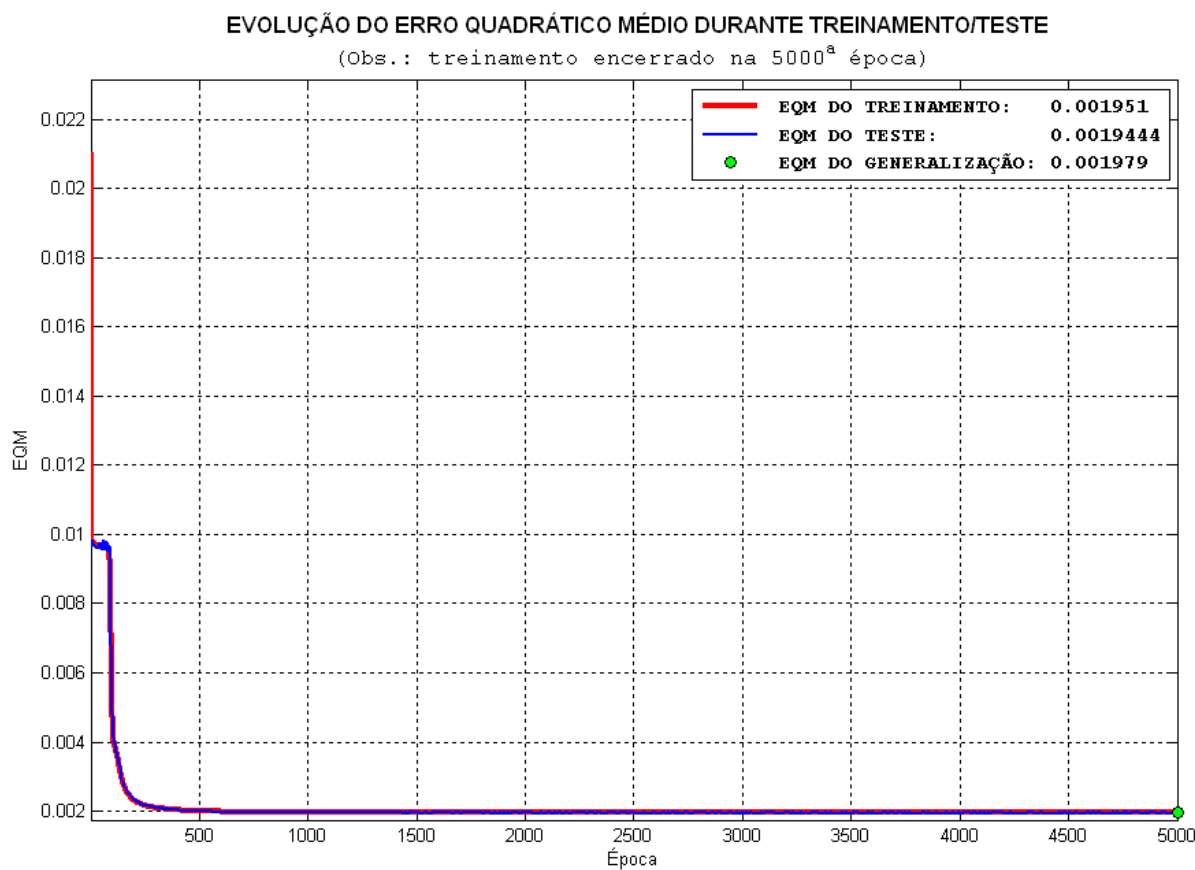


Figura III.15 – EQMs obtidos para a simulação Nº 7 de zona morta tipo I.

A Figura III.16 permite uma comparação entre os cem últimos sinais desejados com os determinados pela rede, referentes à simulação N<sup>o</sup> 8 para a não-linearidade zona morta tipo I.

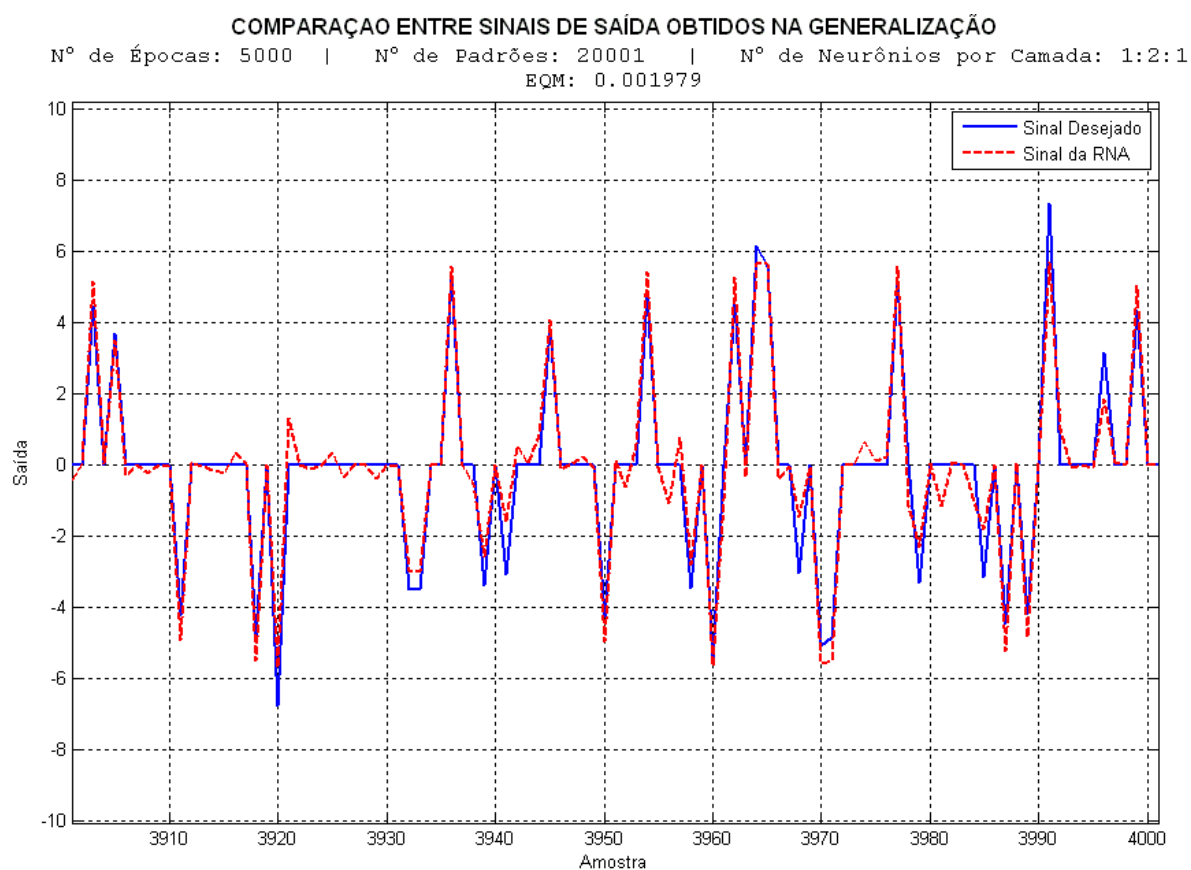


Figura III.16 – Amostra comparativa dos sinais desejados e emitidos pela RNA, referentes à fase de generalização da simulação N<sup>o</sup> 7 de zona morta tipo I.

### III.2.2– Simulação complementar de zona morta tipo II

Dentre as simulações de zona morta tipo II, apresentadas na seção III.3, a configuração Nº 3 obteve o menor EQM de generalização, utilizando a configuração formada por três neurônios na camada escondida, 10001 pares de entrada-saída de treinamento e cem épocas de treinamento.

Diante da evolução do EQM de treinamento para a simulação Nº 3 (Figura III.5), combinada com a amostra comparativa de sinais desejados e emitidos pela rede, exibidos na Figura III.6, a simulação complementar Nº 7 empenhou-se em minimizar a oscilação de aprendizagem e melhorar o sinal de resposta da rede. Para tanto, o conjunto de treinamento foi mantido com 10001 exemplos, a taxa de aprendizagem baixou para 0.01, sem constante de momento e o treinamento foi estendido por 2000 épocas. A partir destas alterações, a Tabela III.9 apresenta os resultados obtidos:

Tabela III.9 – Resultados obtidos para a simulação complementar de zona morta tipo II.

Nº DA SIMULAÇÃO	Nº DE PADRÕES	Nº DE NEURÓNIOS POR CAMADA	Nº DE ÉPOCAS	CRITÉRIO DE PARADA	EQM DE TREINAMENTO	EQM DE TESTE	EQM DE GENERALIZAÇÃO	TEMPO DA SIMULAÇÃO
7	10001	1:3:1	1165	2	0.000404910	0.000464364	0.000373998	12m 57s

Conforme os resultados da Tabela III.9, a simulação Nº 7 foi suficiente para oferecer um EQM abaixo do desejado, tendo finalizado o treinamento com 1165 épocas devido ao EQM de teste ter apresentado valores inferiores a 0.001 nas épocas 1155, 1160 e 1165. A respectiva curva de evolução de EQM, apresentada na Figura III.17 indica que o treinamento melhorou a partir da 1100ª época.

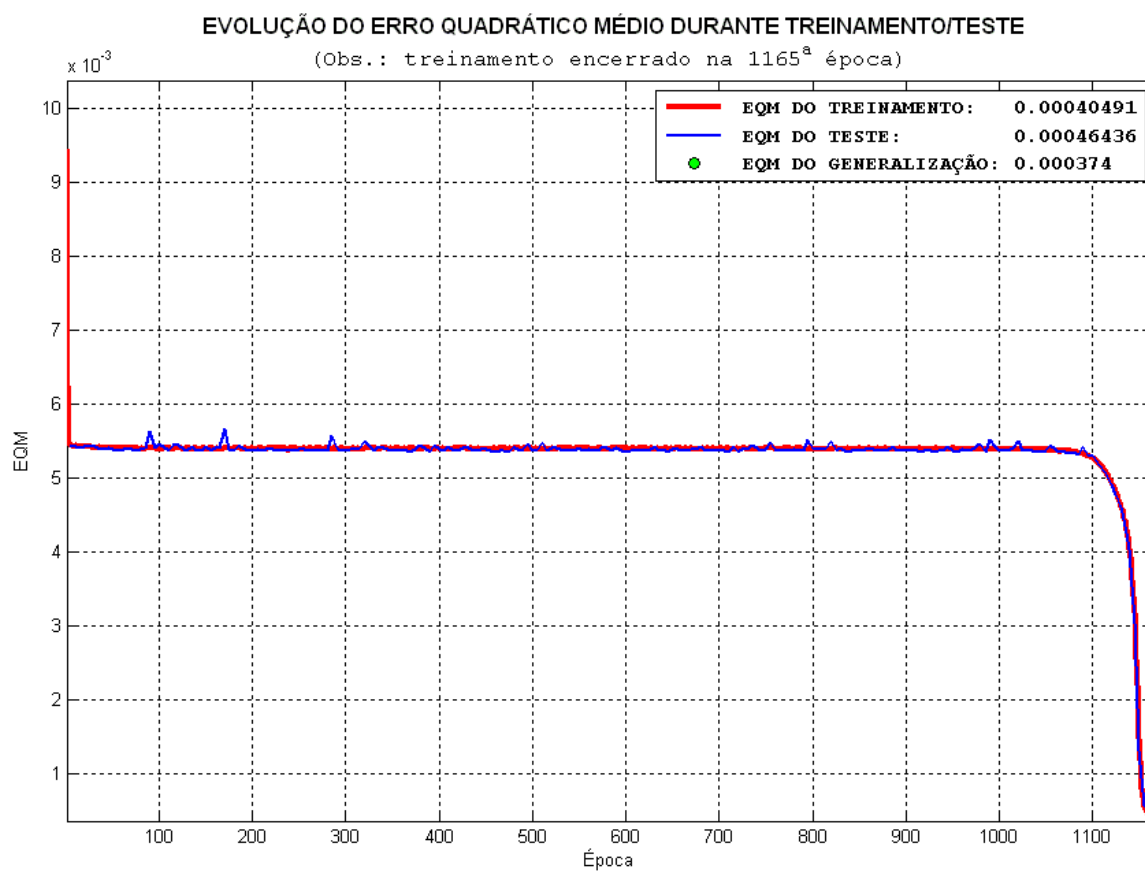


Figura III.17 – EQMs obtidos para a simulação N<sup>o</sup> 7 de zona morta tipo II.

A Figura III.18 permite uma comparação entre os cem últimos sinais desejados com os determinados pela rede, referentes à simulação N<sup>o</sup> 7 para a não-linearidade zona morta tipo II.

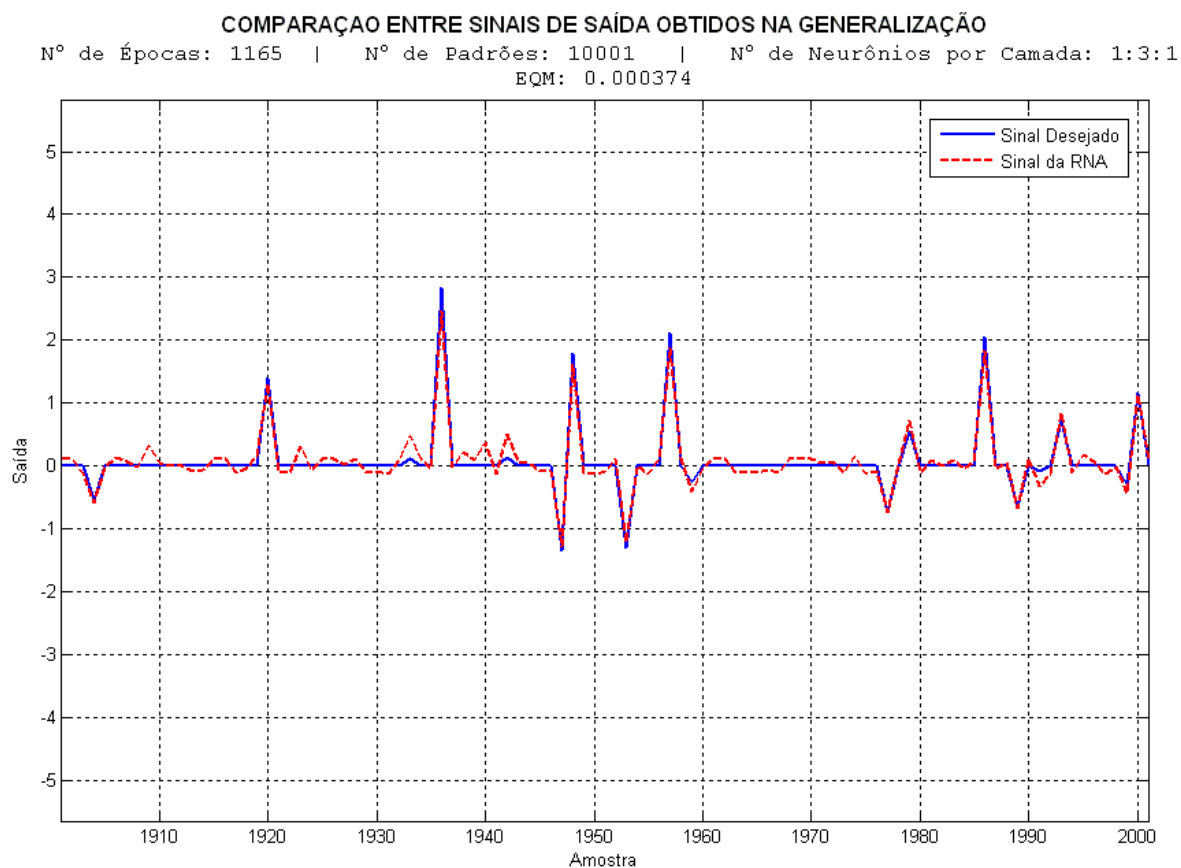


Figura III.18 – Amostra comparativa dos sinais desejados e emitidos pela RNA, referentes à fase de generalização da simulação N<sup>o</sup> 7 de zona morta tipo II.



### III.2.3– Simulação complementar de liga-desliga

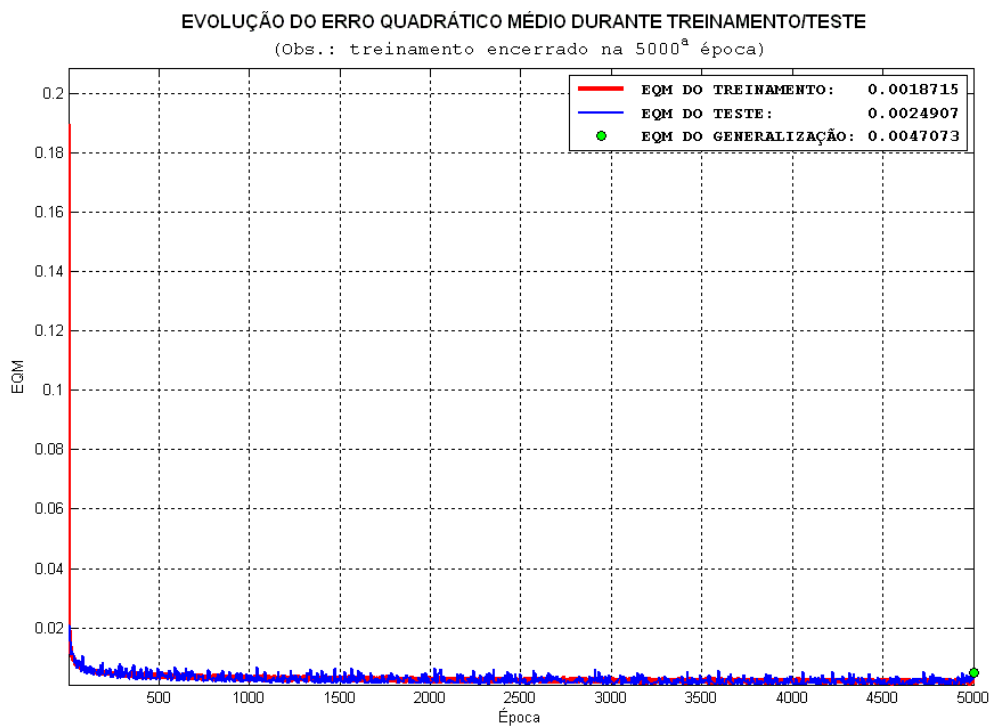
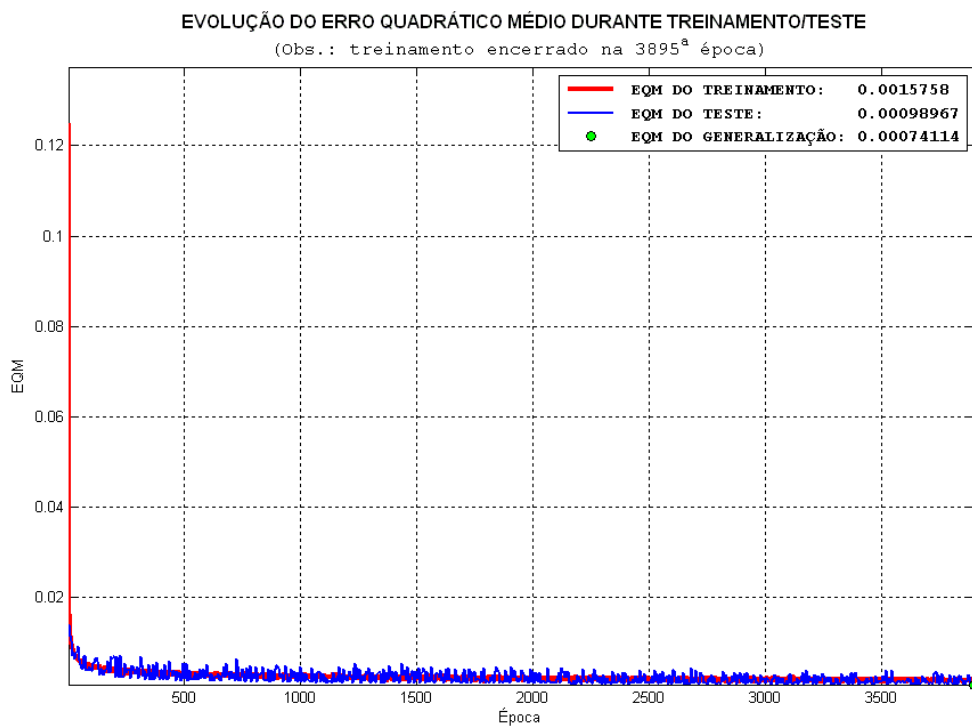
Dentre as simulações de liga-desliga, apresentadas na seção III.5, o menor EQM de generalização foi obtido pela configuração N<sup>o</sup> 2, por meio de dois neurônios na camada escondida, 10001 pares de exemplos e cem épocas de treinamento.

A fim de minimizar o EQM de generalização e prevenir a variação da curva de treinamento, foram feitas duas simulações, ambas com taxa de aprendizagem fixa em 0.01, constante de momento nula e 5000 épocas de treinamento, diferenciando-se apenas pelo número de pares do conjunto de treinamento, sendo os resultados dispostos na Tabela III.10.

Tabela III.10– Resultados obtidos para as simulações complementares de liga-desliga.

N <sup>o</sup> DA SIMULAÇÃO	N <sup>o</sup> DE PADRÕES	N <sup>o</sup> DE NEURÔNIOS POR CAMADA	N <sup>o</sup> DE ÉPOCAS	CRITÉRIO DE PARADA	EQM DE TREINAMENTO	EQM DE TESTE	EQM DE GENERALIZAÇÃO	TEMPO DA SIMULAÇÃO
7	10001	1:2:1	5000	1	0.001871454	0.002490680	0.004707259	54m 03s
8	20001	1:2:1	3895	2	0.001575829	0.000989671	0.000741143	84m 31s

As simulações N<sup>o</sup> 2 (seção III.5) e N<sup>o</sup> 7 têm o mesmo conjunto de treinamento, mas aquela utilizou cem épocas de treinamento, enquanto esta consumiu cinco mil épocas sem ser capaz de melhorar o EQM de generalização já obtido. A curva de treinamento disposta na Figura III.19 indica que o EQM de treinamento da simulação N<sup>o</sup> 7 estacionou a partir da 2000<sup>a</sup> época (desvio-padrão de  $3 \times 10^{-4}$ ).

Figura III.19 – EQMs obtidos para a simulação N<sup>o</sup> 7 de liga-desliga.Figura III.20 – EQMs obtidos para a simulação N<sup>o</sup> 8 de liga-desliga.

O menor EQM de generalização foi obtido pela simulação N<sup>o</sup> 8, com o valor de 0.000741143, abaixo do desejado, tendo utilizado um conjunto de treinamento composto por 20001 pares de exemplos, sendo o treinamento encerrado em razão dos EQMs de teste das épocas 3885, 3890 e 3895 terem sido inferiores a 0.001. A Figura III.20 apresenta a evolução dos EQMs de treinamento e teste para a simulação N<sup>o</sup> 8.

Uma comparação entre os cem últimos sinais desejados com os emitidos pela rede, referentes à simulação N<sup>o</sup> 8 para a não-linearidade liga-desliga, é exibida na Figura III.21.

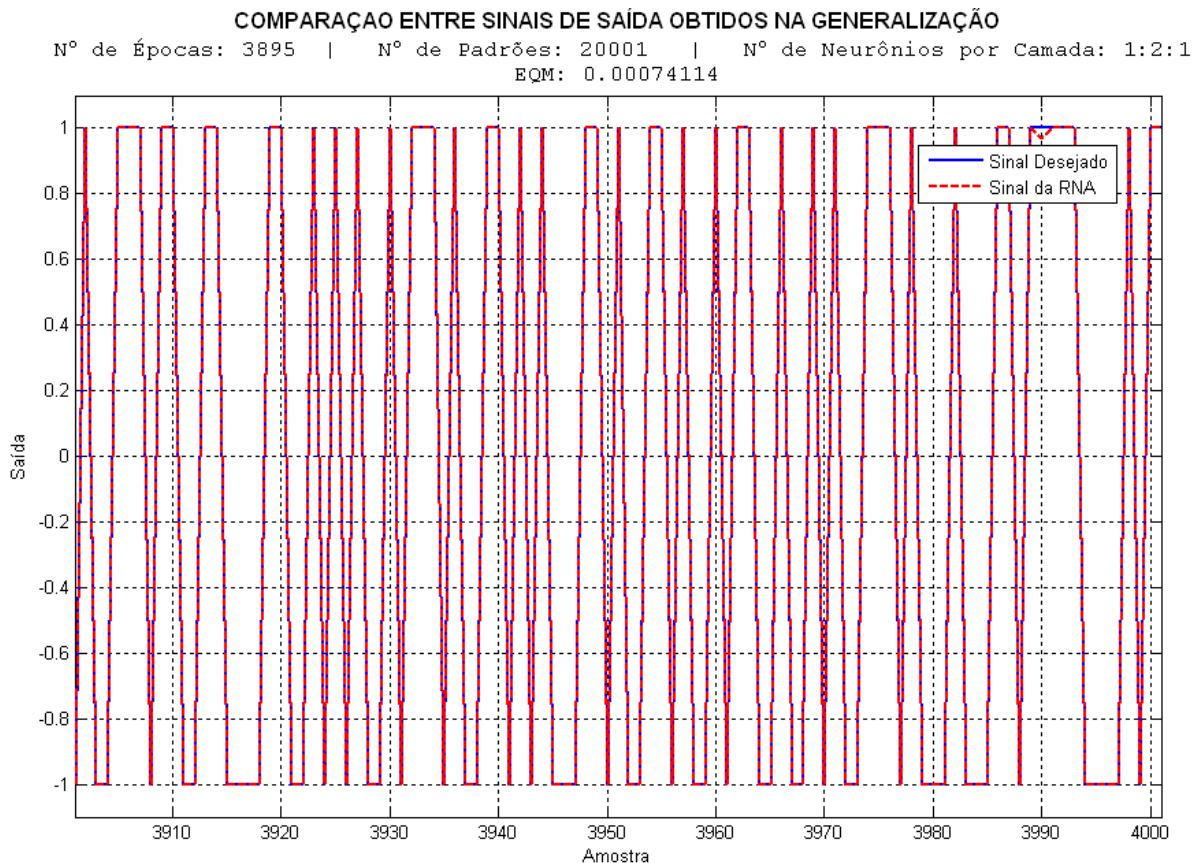


Figura III.21 – Amostra comparativa dos sinais desejados e emitidos pela RNA, referentes à fase de generalização da simulação N<sup>o</sup> 8 de liga-desliga.

### III.2.4– Simulação de liga-desliga com histerese

Dentre as simulações de liga-desliga com histerese, apresentadas na seção III.6, o melhor resultado foi conquistado pela simulação N<sup>o</sup> 3, com cinco neurônios na camada escondida, 20001 pares de exemplos e tendo consumido cem épocas de treinamento. A Figura III.11 sugere a necessidade de prevenir as oscilações do processo de treinamento bem como assegurar o ponto de convergência da rede.

Foram procedidas duas simulações complementares, ambas com taxa de aprendizagem fixa em 0.01, constante de momento nula e 3000 épocas de treinamento, diferenciando-se apenas pelo número de pares do conjunto de treinamento, cujos resultados obtidos constam na Tabela III.11.

Tabela III.11– Resultados obtidos para as simulações complementares de liga-desliga com histerese.

N <sup>o</sup> DA SIMULAÇÃO	N <sup>o</sup> DE PADRÕES	N <sup>o</sup> DE NEURÔNIOS POR CAMADA	N <sup>o</sup> DE ÉPOCAS	CRITÉRIO DE PARADA	EQM DE TREINAMENTO	EQM DE TESTE	EQM DE GENERALIZAÇÃO	TEMPO DA SIMULAÇÃO
7	21001	2:5:1	3000	1	0.002049744	0.001979507	0.001906812	71m 45s
8	42001	2:5:1	615	2	0.001019925	0.000957569	0.000942506	29m 39s

As simulações N<sup>os</sup> 3 (seção III.6) e 7 têm o mesmo conjunto de treinamento, mas após 3000 épocas de treinamento, esta simulação complementar apresentou EQM de generalização igual a 0.001906812, superior ao obtido pela N<sup>o</sup> 3. A curva de evolução disposta na Figura III.22 indica que o EQM de treinamento estacionou a partir da 500<sup>a</sup> época (desvio-padrão de  $4 \times 10^{-5}$ ).

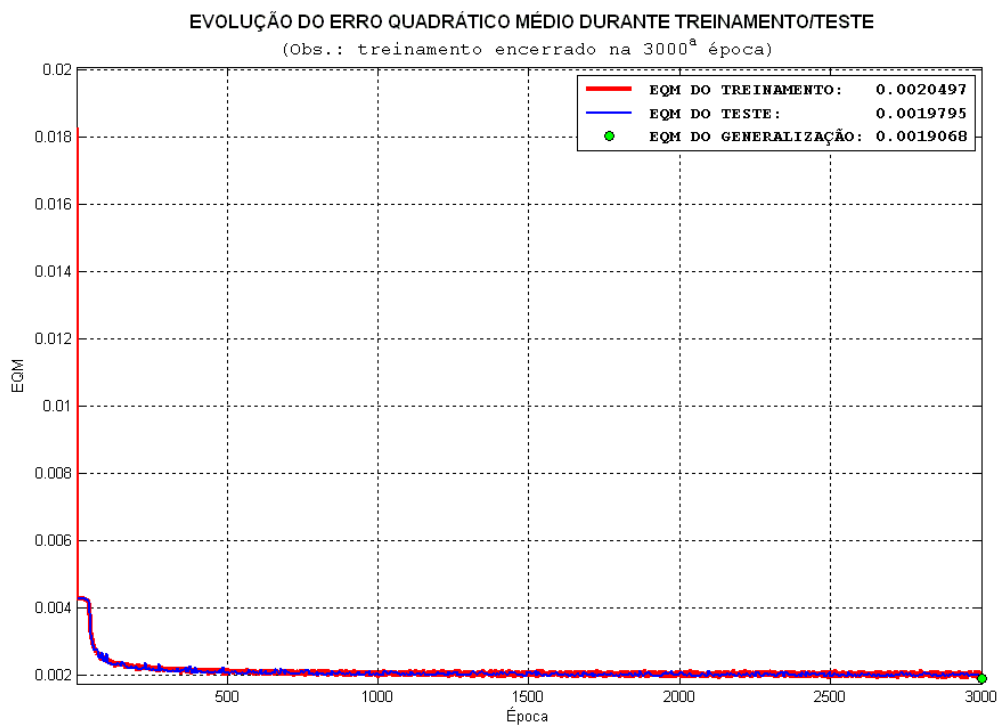


Figura III.22 – EQMs obtidos para a simulação N° 7 de liga-desliga com histerese.

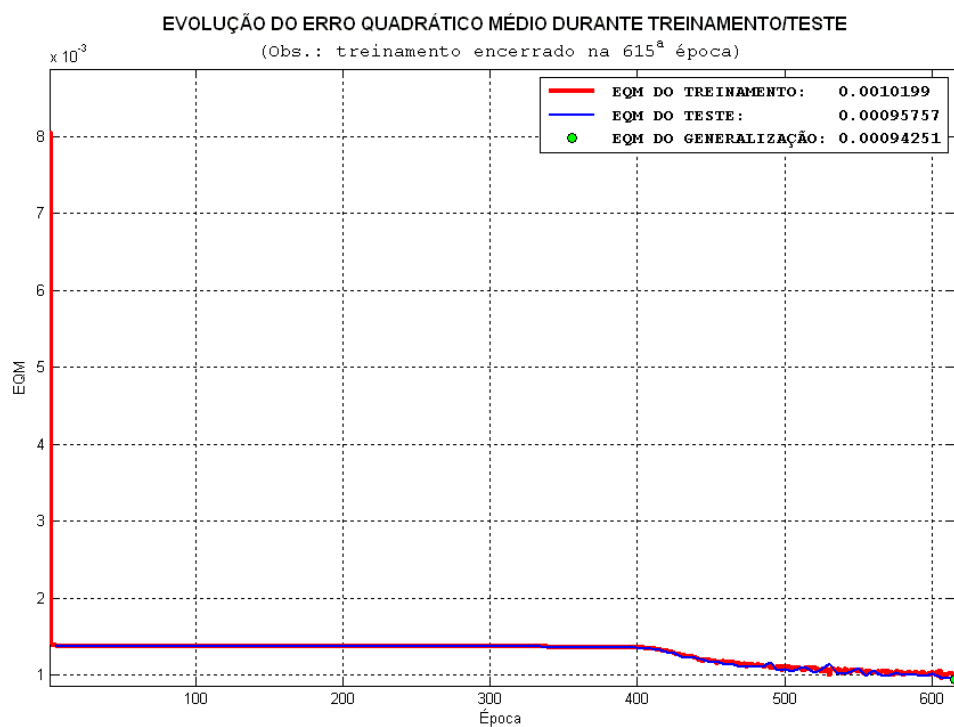


Figura III.23 – EQMs obtidos para a simulação N° 8 de liga-desliga com histerese.

Por sua vez, a simulação Nº 8 obteve EQM de generalização igual a 0.000942506, tendo utilizado um conjunto de treinamento composto por 42001 pares de exemplos com a aprendizagem encerrada na 615ª época, devido ao EQM de teste ter se mantido abaixo de 0.001 nas épocas 605, 610 e 615, conforme visto na Figura III.23.

A Figura III.24 representa uma comparação entre os cem últimos sinais desejados com os calculados pela rede, referentes à simulação Nº 8 para a não-linearidade liga-desliga com histerese.

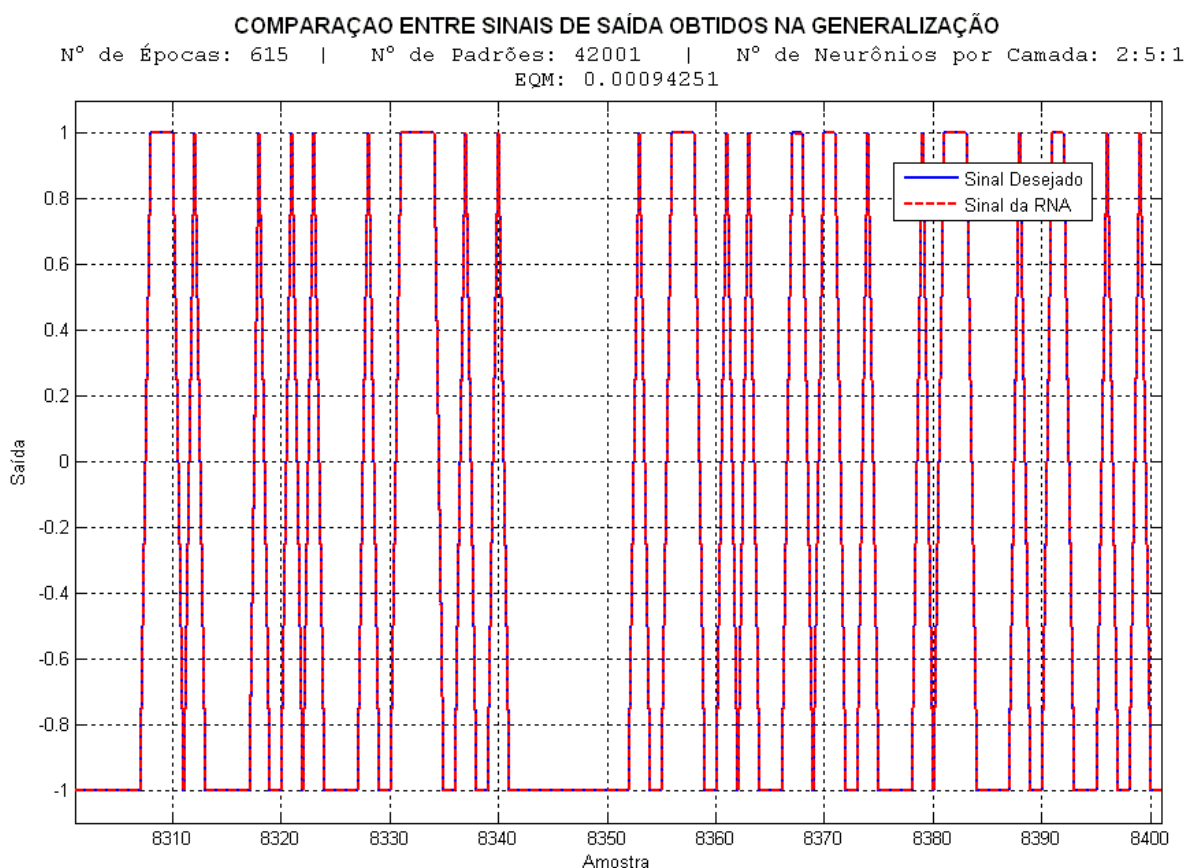


Figura III.24 – Amostra comparativa dos sinais desejados e emitidos pela RNA, referentes à fase de generalização da simulação Nº 8 de liga-desliga com histerese.

### III.2.5– Simulação de liga-desliga com zona morta

Dentre as simulações de liga-desliga com zona morta, tratadas na seção III.7, apresentou melhor desempenho a de N<sup>o</sup> 2, cujo arranjo é composto por dois neurônios na camada escondida, 10001 pares de exemplos no conjunto de treinamento e cem épocas de treinamento. De acordo com a curva de EQM de generalização, exposta na Figura III.13, as cem épocas utilizadas não foram suficientes para concluir quanto à convergência da rede.

Neste caso, a simulação proposta incrementou o número de épocas para 2000, e manteve os parâmetros de taxa de aprendizagem e constante de momento conforme a configuração padronizada. Após executada, a simulação N<sup>o</sup> 7 ofereceu os resultados dispostos na Tabela III.12.

Tabela III.12– Resultados obtidos para a simulação complementar de liga-desliga com zona morta.

N <sup>o</sup> DA SIMULAÇÃO	N <sup>o</sup> DE PADRÕES	N <sup>o</sup> DE NEURÔNIOS POR CAMADA	N <sup>o</sup> DE ÉPOCAS	CRITÉRIO DE PARADA	EQM DE TREINAMENTO	EQM DE TESTE	EQM DE GENERALIZAÇÃO	TEMPO DA SIMULAÇÃO
7	10001	1:2:1	1990	2	0.001294856	0.000997186	0.000748027	24m 33s

Segundo os resultados da Tabela III.12, pode-se concluir que a simulação N<sup>o</sup> 7 obteve EQM abaixo do desejado utilizando 1990 épocas.

A seguir, a Figura III.25 demonstra a evolução dos EQM de treinamento para a simulação complementar Nº 7, atinente ao efeito liga-desliga com zona morta.

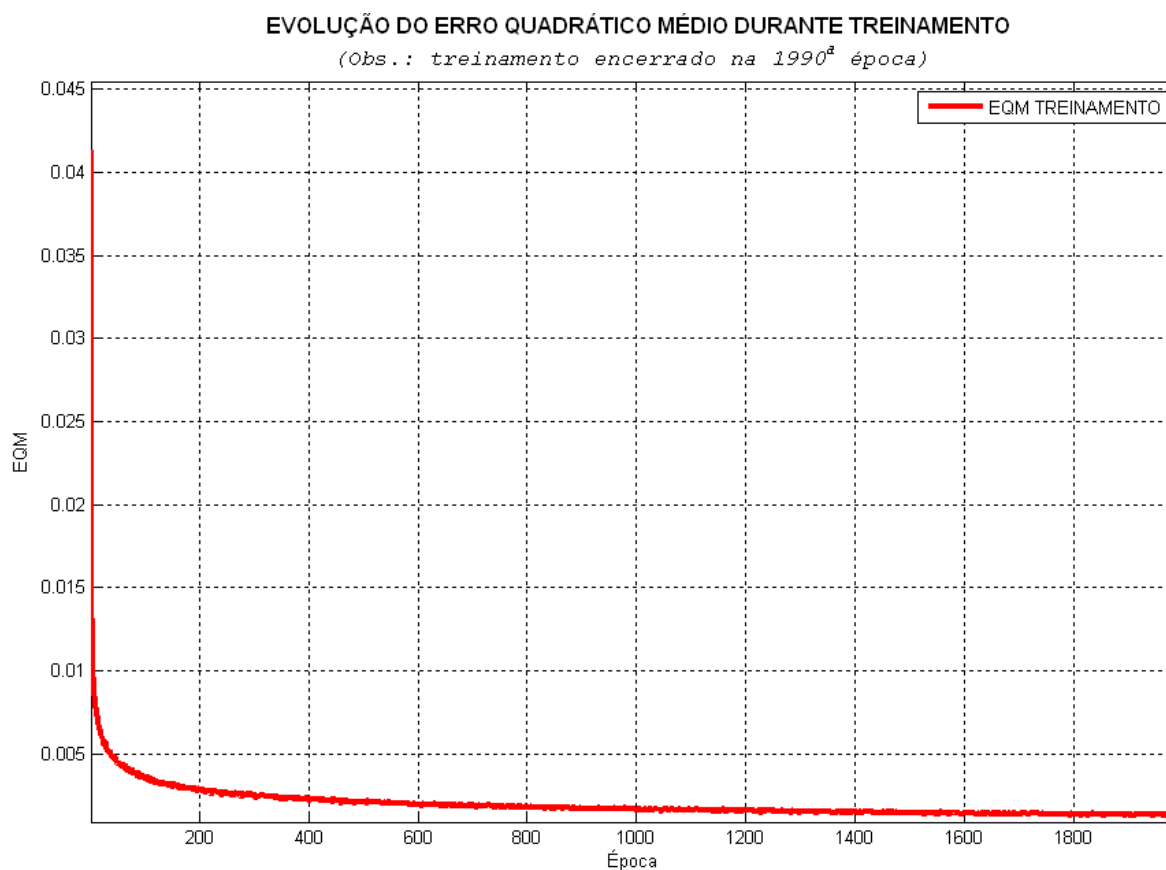


Figura III.25 – EQMs obtidos para a simulação Nº 7 de liga-desliga com zona morta.



A Figura III.26 permite uma comparação entre os cem últimos sinais desejados com os emitidos pela RNA, treinada com 1990 épocas, referentes à não-linearidade de liga-desliga com zona morta.

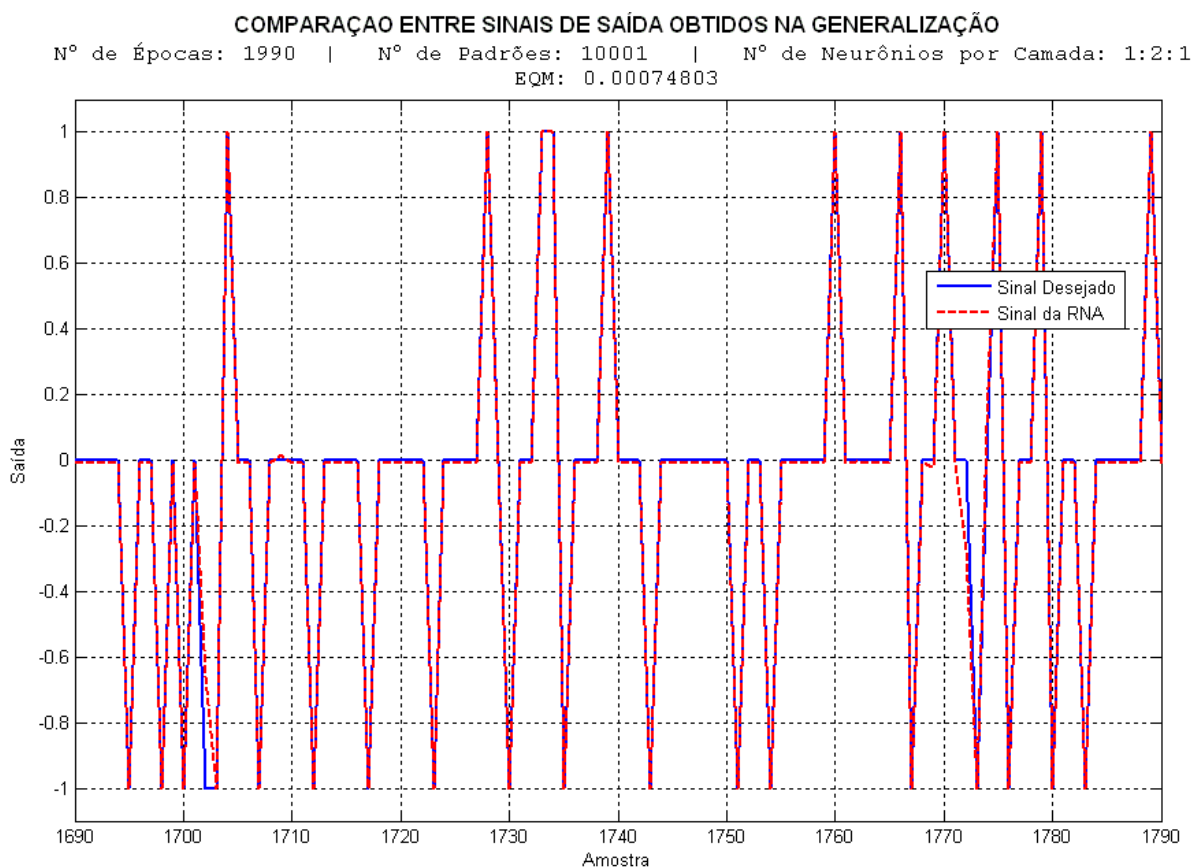


Figura III.26 – Amostra comparativa dos sinais desejados e emitidos pela RNA, referentes à fase de generalização da simulação Nº 7 de liga-desliga com zona morta.

A partir da amostra representada pelo gráfico da Figura III.25, apoiada pelo EQM de generalização em 0.000748027, depreende-se que o desempenho da rede em simular esta não-linearidade com a atual configuração melhorou em relação à simulação Nº 2.

## CAPÍTULO IV

### DISCUSSÃO DOS RESULTADOS

Dos resultados obtidos pelas simulações realizadas neste trabalho, utilizando redes neurais artificiais tipo MLP para representação de não-linearidades rígidas de saturação, zona morta, saturação com zona morta, liga-desliga, liga-desliga com histerese e liga-desliga com zona morta, alguns pontos merecem ser destacados:

- A não-linearidade de saturação foi representada satisfatoriamente ainda na primeira etapa de simulações. Das seis experiências realizadas, cinco tiveram o treinamento encerrado na 15ª ou 20ª época, com EQMs de generalização abaixo de 0.001. A única exceção refere-se à arquitetura composta por dez neurônios na camada escondida, que não atingiu o EQM desejado mesmo tendo consumido o número máximo de épocas. O teorema de Kolmogorov e a regra de Baum-Haussler contribuíram para a definição dos respectivos parâmetros da RNA. Simulações complementares não foram necessárias;
- Nenhuma das simulações de zona morta tipo I obteve o EQM de generalização determinado. Tendo em vista a curva de treinamento referente à simulação Nº 2, detentora do melhor resultado na primeira fase, ter estacionado a partir da 25ª época sem apresentar oscilações significativas (desvio-padrão de  $2 \times 10^{-5}$ ), as simulações complementares detiveram-se apenas em avaliar o número de épocas e o tamanho do conjunto de treinamento. Assim, na segunda etapa, a simulação complementar Nº 7 apenas prolongou o número de épocas para 5000, enquanto a de Nº 8 também ampliou o conjunto de treinamento para 20001 padrões, porém, ambas não foram capazes de obter o EQM de generalização admissível, cabendo o

melhor resultado geral à simulação N<sup>o</sup> 7, cujo processo de treinamento estabilizou a partir da 700<sup>a</sup> época, com desvio-padrão de  $4 \times 10^{-6}$ ;

- A representação da não-linearidade de zona morta tipo II tornou-se possível através da simulação N<sup>o</sup> 7, uma vez que nenhuma das simulações realizadas na primeira etapa reuniu os parâmetros necessários para cumprir a exigência de EQM mínimo de generalização. Naquela ocasião, a simulação N<sup>o</sup> 3, configurada segundo a regra de Baum-Haussler, combinada com o teorema de Kolmogorov, obteve o melhor resultado possível e, de acordo com a respectiva curva de treinamento, houve ligeira oscilação (desvio-padrão de  $4 \times 10^{-5}$ ). Por sua vez, a simulação complementar N<sup>o</sup> 7 atendeu o critério de generalização através da configuração na qual se adotou a taxa de aprendizagem de 0.01, sem momento, para minimizar possíveis oscilações do processo de aprendizagem. Por conseguinte, para comportar a dilatação do tempo necessário à convergência da rede, o número de épocas havia sido estendido para 2000, mas o treinamento foi encerrado na 1165<sup>a</sup> época;
- A representação da não-linearidade de saturação com zona morta obteve êxito na primeira etapa, através da simulação N<sup>o</sup> 2, cuja rede se compôs de dois neurônios na camada escondida e utilizou 10001 padrões de treinamento. O número de neurônios internos avizinhou-se com o proposto pelo teorema de Kolmogorov que, associado à regra de Baum-Haussler, nortearam o tamanho do vetor de treinamento. O referido processo de treinamento foi encerrado na 15<sup>a</sup> época e, portanto, simulações complementares não foram necessárias;
- A não-linearidade de de liga-desliga obteve representação suficiente por meio da 8<sup>a</sup> simulação, pois nenhuma das simulações realizadas na primeira etapa alcançou o EQM de generalização estipulado, sendo o melhor resultado possível atribuído à simulação N<sup>o</sup> 2, da qual se teve um processo de aprendizagem oscilatório (desvio-

padrão de  $6 \times 10^{-3}$ ) e incerto quanto à convergência da rede. O teorema de Kolmogorov e a regra de Baum-Haussler contribuíram na aproximação do melhor arranjo possível quanto à topologia da rede e o quantitativo de padrões de treinamento, respectivamente. Na fase seguinte, foram feitas duas simulações complementares, ambas com dois neurônios na camada escondida, taxa de aprendizagem de 0.01, constante de momento nula e 5000 épocas de treinamento, no máximo. A simulação Nº 7 não cumpriu a meta desejada, tendo utilizado o mesmo conjunto de treinamento da simulação Nº 2 e esgotado o limite máximo de épocas. Por sua vez, a simulação Nº 8, que utilizou 42001 padrões de treinamento, convergiu na 3895ª época. A taxa de aprendizagem menor minimizou a oscilação da aprendizagem, mas também ampliou o tempo necessário à definição do EQM desejado;

- A não-linearidade de liga-desliga com histerese destaca-se das demais por incorporar recorrência no seu aprendizado e foi mapeada adequadamente pela simulação Nº 8. As simulações realizadas na primeira fase não cumpriram o critério de EQM de generalização mínimo, cabendo a melhor resposta possível à simulação Nº 3, cuja configuração estava de acordo a regra de Baum-Haussler, combinada com o teorema de Kolmogorov. O respectivo processo de treinamento apresentou permanente oscilação (desvio-padrão de  $4 \times 10^{-4}$ ) e não asseverou a convergência da rede. Em seguida, foram procedidas duas simulações complementares, diferenciadas apenas pela quantidade de pares entrada-saída de treinamento, e que tiveram em comum os cinco neurônios na camada escondida, a taxa de aprendizagem decrescida para 0.01, com anulação da constante de momento, e o treinamento limitado em 3000 épocas. Desse modo, a simulação Nº 7, que utilizou conjunto idêntico à simulação Nº 3, rendeu EQM de generalização ainda maior que o já alcançado. Por sua vez, a 8ª simulação, que se fez com 42001 padrões, cumpriu o EQM admissível na 615ª época. Em ambas as simulações, a taxa de

aprendizagem reduzida possibilitou minimizar a oscilação acima mencionada, porém demandou mais épocas de treinamento para se obter o EQM mínimo;

- A não-linearidade de liga-desliga com zona morta foi representada adequadamente na segunda etapa de simulações. Encerrada a primeira fase, a simulação N<sup>o</sup> 2 deteve o melhor resultado possível, porém não assegurou a convergência da rede dentro do número de épocas estabelecido, conforme observada a respectiva curva de treinamento. Assim, a simulação complementar N<sup>o</sup> 7 apenas ampliou o número de épocas, mantendo os demais parâmetros, tendo obtido o EQM de generalização admissível na 1990<sup>a</sup> época. Outrossim, o número de neurônios da camada escondida avizinha-se com o estimado pelo teorema de Kolmogorov, enquanto o conjunto de treinamento é determinado conforme a regra de Baum-Haussler.

## CONCLUSÃO

Os resultados obtidos e analisados neste trabalho permitiram as seguintes conclusões:

A rede neural artificial do tipo MLP com treinamento supervisionado por retropropagação de erro, utilizando função tangente hiperbólica nas camadas escondida e de saída, demonstrou capacidade para aprender e generalizar as não-linearidades rígidas apresentadas.

A avaliação da capacidade da RNA em identificar as não-linearidades deteve-se ao cumprimento do EQM de generalização mínimo. Assim:

- As não-linearidades de saturação e saturação com zona morta cumpriram o requisito de generalização ainda na primeira etapa de simulações. A ocorrência da convergência por volta da 20ª época de treinamento evidencia a capacidade da rede em aprender e generalizar este tipo de mapeamento não-linear;
- A não-linearidade de zona morta tipo I não atendeu o critério de EQM de generalização mínimo em nenhuma das simulações realizadas. Entretanto, o melhor resultado possível refere-se a um erro de 0.2%, o qual pode ser considerado aceitável em determinados sistemas de controle;
- A não-linearidade de zona morta tipo II exigiu simulação complementar para atender o EQM de generalização aceitável. Para tanto, a configuração padrão sofreu redução da taxa de aprendizagem, anulação da constante de momento e incremento do número de épocas;
- As não linearidades de liga-desliga e liga-desliga com histerese também solicitaram a segunda fase de simulações para satisfazer o EQM de generalização desejado. Nestes casos, foram alterados os parâmetros de taxa de aprendizagem, constante

de momento, número de épocas e tamanho do conjunto de treinamento, em relação à configuração padronizada;

- A não-linearidade de liga-desliga com zona morta também fez uso da segunda fase de simulações e atingiu o EQM de generalização apenas incrementando o número de épocas da configuração padrão.

A partir da análise comparativa dos EQMs é possível inferir quanto aos parâmetros de número de neurônios da camada interna e tamanho do conjunto de treinamento da RNA:

- O número superestimado de neurônios da camada interna aumentou a complexidade da rede e não melhorou o processo de minimização do erro;
- A rede neural, em geral, não generalizou eficientemente quando treinada com número insuficiente de pares de entrada-saída;
- O conjunto de treinamento superestimado favoreceu o treinamento, mas prejudicou a generalização da rede, demonstrando que a rede “decorou” os pares sem mapear a função pretendida.

Estes resultados serviram para confirmar a utilização de redes neurais na área de controle e automação, devido aos resultados promissores que oferecem em termos de simplicidade e qualidade dos resultados.

As redes neurais artificiais gozam de melhor desempenho quando heurísticas são minimamente empregadas, ao invés da simples tentativa-e-erro. Especificamente, o teorema de Kolmogov, utilizado na estimativa do número de neurônios da camada escondida, e a regra de Baum-Haussler, para limitar o tamanho do conjunto de treinamento, permitiram determinar a melhor solução em 71% das não-lineares estudadas.

## SUGESTÕES DE CONTINUIDADE

Como ainda há muitas abordagens que podem ser contempladas em estudos futuros com vistas a garantir o mapeamento de não-linearidades rígidas, surgem como propostas para trabalhos futuros:

- Identificação de outras não-linearidades rígidas, tais como folga, atrito, histerese etc.;
- Representação das não-linearidades por RNAs com funções de ativação de base radial;
- Utilização de taxa de aprendizagem dinâmica no treinamento da RNA;
- Seleção dos exemplos de treinamento;
- Substituição de um controlador físico pela RNA.



## REFERÊNCIAS BIBLIOGRÁFICAS

- [1] SLOTINE, J. J. E.; LI, W.; *Applied Nonlinear Control*, New Jersey, Prentice-Hall, 1991.
- [2] KHALIL, H. K.; *Nonlinear Systems*, London, Macmillan Publishing Co, 1992.
- [3] DAVID M. A.; YASUNDO T.; MICHAEL J. R.; *Introducing Systems and Control*, USA, McGraw-Hill, 1974.
- [4] GIBSON, J. E.; *Nonlinear Automatic Control*, New York, McGraw-Hill, 1963.
- [5] OGATA, K.; *Engenharia de Controle Moderno*, 2ed., Rio de Janeiro, Prentice-Hall do Brasil, 1993.
- [6] SILVA, G.V.M.; *Controlo Não Linear*, Setúbal, Instituto Politécnico de Setúbal, 2003.
- [7] CAMPOS, M. M.; KAKU, S.; *Sistemas Inteligentes em Controle e Automação de Processos*, Rio de Janeiro, Ciência Moderna, 2004.
- [8] SHAW, I. S.; SIMÕES, M. G.; *Controle e Modelagem Fuzzy*, São Paulo, Edgard Blücher, 2001.
- [9] CARDENUTO, N. C.; *Aeronaves Configuradas por Controle do Tipo Preditivo Neural*, Tese de D. Sc., INPE, São José dos Campos, SP, Brasil, 2003.
- [10] TAFNER, M. A.; As Redes Neurais Artificiais: Aprendizado e Plasticidade. *Cérebro & Mente: Revista eletrônica de divulgação científica em neurociências*, n. 5. Disponível em <<http://www.epub.org.br/cm/n05/tecnologia/plasticidade2.html>>. Acesso em: 7 abr. 2007.
- [11] ALEKSANDER I.; MORTON H.; *An Introduction to Neural Computing*, 2ed., London, International Thomson Computer Press, 1995.
- [12] ENDER, L.; *Redes Neurais Aplicadas em Estratégias de Controle Não-linear*, Tese de D. Sc., FEQ/UNICAMP, Campinas, SP, Brasil, 2002.
- [13] MENDES, P. P. C.; *Aplicação de Redes Neurais Artificiais na Análise em Tempo Real de Estabilidade de Tensão de Regime Permanente de Sistemas Elétricos de Potência*, Tese de D. Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, 1999.
- [14] REAL, J. A. F.; *Controlador Adaptativo de Modo Dual com Redes Gaussianas*, Tese de D. Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, 1999.
- [15] COMERLATO, C. A.; *Redes Neurais Artificiais em Sistemas Críticos quanto à Segurança: Uma abordagem para Certificação*, Tese de D. Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, 2002.
- [16] KANO, R. Y. C.; *Uso das Redes Neurais como Ferramenta na Medição de Empuxo de Foguetes*, Dissertação de Mestrado em Tecnologia, PPTEC/CEFET/RJ, Rio de Janeiro, RJ, Brasil, 2004.

- [17] AZEVEDO, M. P.; *Identificação de Padrões no Sinal Ultra-sônico de Coagulação Sangüínea utilizando Redes Neurais*, Dissertação de Mestrado em Tecnologia, PPTEC/CEFET/RJ, Rio de Janeiro, RJ, Brasil, 2004.
- [18] CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA CELSO SUCKOW DA FONSECA. Diretoria de Pesquisa e Pós-Graduação, Rio de Janeiro, 2005. 5º Seminário de iniciação científica – Livro de Resumos. Disponível em <[http://www.cefet-rj.br/pesquisa/COPET/5\\_SIC\\_CEFET-RJ\\_2005.pdf](http://www.cefet-rj.br/pesquisa/COPET/5_SIC_CEFET-RJ_2005.pdf)> Acesso em: 17 ago. 2006.
- [19] HUNT, K. J.; SBARBARO, D.; ZBIKOWSKI, R.; GAWTHROP, P. J.; “Neural networks for control systems: a survey”, *Automatica*, v.28, n.6, pp. 1083-1112, 1992.
- [20] NARENDRA, K. S.; “Neural Networks for Control: Theory and Practices”, *Proceedings of the IEEE*, v. 84, n.10, pp. 1385-1406, 1996.
- [21] CYBENKO, G.; “Continuous Valued Neural Networks with Two Hidden Layers are Sufficient”, *Technical report, Department of Computer Science*, Tufts University, Medford, MA, 1988.
- [22] CYBENKO, G.; “Approximation by Superpositions of a Sigmoid Function”, *Mathematics of Control, Signals and Systems*, v.2, pp. 303–314, 1989.
- [23] TENG, F. C.; “Review on Application of Neural Networks to Adaptive Control of Nonlinear Systems”, *Journal Robust and Nonlinear Control*, v.11, n.9, pp. 879-880, 2001.
- [24] MILLER, W. T.; HEWES, R. P.; GLANZ, F. H.; KRAFT, L. G.; "Real-time Dynamic Control of an Industrial Manipulator using a Neural Network based Learning Controller," *IEEE Transactions on Robotics and Automation*, v.6, n.1-9 , 1990.
- [25] ZUBEN, F. J. V.; *Redes Neurais Aplicadas ao Controle de Máquina de Indução*, Dissertação de M.Sc., FEEC/UNICAMP, Campinas, SP, Brasil, 1993.
- [26] KOVACS, Z. L.; *Redes Neurais Artificiais Fundamentos e Aplicações*, São Paulo, Collegium Cognitivo e Edição Acadêmica, 1996.
- [27] HAYKIN, S.; *Redes Neurais: Princípios e Práticas*, 2ed., Porto Alegre, Bookman, 2001.
- [28] BRAGA, A. P.; LUDEMIR, T. B.; CARVALHO, A. P.; *Redes Neurais Artificiais: Teoria e Aplicações*, 2ed., Rio de Janeiro, LTC, 2001
- [29] MCCULLOCH, J.; PITTS, W.; “A Logical Calculus of the Ideas Immanent in Nervous Activity”, *Bulletin of Mathematical Biophysics*, v.7, pp. 115-133, 1943.
- [30] HEBB, D.; *The Organization of Behavior*, New York, N.Y. Wiley, 1949.
- [31] ROSENBLATT, F.; “The Perceptron: A Probabilistic model for information storage and organization in the brain”, *Psychology. Review*, v. 65, pp. 386-408, 1958.
- [32] WIDROW, B; HOFF, M.E.; "Adaptive switching circuits," *IRE WESCON Convention Record*, v.4, pp. 96-104, 1960.

- [33] MINSKY, M. L.; PAPERT, S. A.; *Perceptrons: an Introduction to Computational Geometry*, 3ed., Massachussets, The MIT Press, 1988. Impressão modificada do original de 1969.
- [34] HOPFIELD, J.; “Hopfield Neural Networks for Timetabling: Formulations, Methods, and Comparative Results”. *Computers and Industrial Engineering archive*, v.44, n.2, pp. 283-35, 2003.
- [35] RUMELHART, D. E.; McCLELLAND, J. L.; *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, 3ed., Massachussets, The MIT Press, 1986.
- [36] PAULA, M. D.; *Estudo dos Tons e suas Características para Utilização em um Classificador de Navios Baseado em Redes Neurais*, Dissertação de M.Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, 2007.
- [37] GORI, M.; SCARSELLI, F.; TSOI, A.C.; “On the Closure of the Set of Functions that can be Realized by a given Multilayer Perceptron”, *Journal Article, IEEE Transaction Neural Networks*, v. 9, pp. 1086-98, 1998.
- [38] ONAT, A.; KITA, H.; NISHIKAWA, Y.; “Recurrent Neural Networks for Reinforcement Learning: Architecture, Learning and Internal Representation”, *Proceedings of the IEEE - World Congress on Computational Intelligence*, pp. 2010–2015, 1998.
- [39] Kaelbling, L. P.; Littman, M. L.; Moore, A. W.; “Reinforcement Learning: A Survey”, *Journal of Artificial Intelligence Research*, v.4, pp. 237–285, 1996.
- [40] Kolmogorov, A. N.; “On the Representation of Continuous Functions of Many Variables by Superposition of Continuous Functions of one Variable and Addition”. *Dokl. Akad. Nauk SSSR*, v.114, pp. 953–956.
- [41] Baum, E.B.; Haussler, D.; “What Size Net Gives Valid Generalization?”, *Neural Computation*, v.1 pp. 151 – 160, 1990.
- [42] Ballini, R.; *Análise e Previsão de Vazões Utilizando Modelos de Série Temporais, Redes Neurais e Redes Neurais Nebulosas*, Tese de Doutorado, FEEC/UNICAMP, Campinas, SP, Brasil, 2000.
- [43] Rodrigues, G. C.; *Utilização de Redes Neurais para Previsão de Ventos no Horizonte de 24 Horas*, Dissertação de M. Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, 2007.
- [44] THE MATHWORKS. Apresenta ferramentas computacionais de modelagem de sistemas. Disponível em <<http://www.mathworks.com>>. Acesso em: 10 abr. 2008.
- [45] Chapman, S. J.; *Matlab Programming for Engineers*, Thomson Engineering, 2005.
- [46] Barreto, G. A.; “Perceptron Multicamadas e o Algoritmo de Retropropagação do Erro”. DET/PPGETI/UFC, PE, Brasil, 2007. Disponível em <<http://www.deti.ufc.br/~guilherme/>>. Acesso em: 18 mar. 2008.
- [47] Barreto, G. A.; Apresenta textos e trabalhos acadêmicos sobre RNA. Disponível em <<http://www.deti.ufc.br/~guilherme/TI016/>>. Acesso em: 06 out. 2007.

## Apêndice 1

Código-fonte em MATLAB da função geradora de padrões referentes a não-linearidade de saturação

```

1: % Geração de padrões de entrada para a rede MLP (backpropagation)
2: % Simulação de Saturação
3: % x = sinal de entrada
4: % d = saída desejada
5:
6: clear; close all;
7:
8: C = 0;
9: while C == 0
10:     clc;
11:
12:     NI = input('No. de Iterações [Padrao 100]: ');
13:     if isempty(NI)
14:         NI = 100;
15:     end
16:
17:     TA = input('Taxa de Amostragem/Iteração [Padrao 100]: ');
18:     if isempty(TA)
19:         TA = 100;
20:     end
21:
22:     if NI/TA > 1
23:         P_repete = input('Haverá padrões repetidos! Continuar? S/N [Padrão N]: ','s');
24:         if (P_repete == 'S' | P_repete == 's')
25:             C = 1;
26:         end
27:     else C = 1;
28:     end
29: end
30: TA=1/TA;
31:
32: t = 0:TA:NI;
33:
34: for i=1:length(t)
35:     x(1,i) = 2.0 .* (cos(2*pi*3.5*t(i))*sin(2*pi*5*t(i))+cos(2*pi*29*t(i))+sin(2*pi*.33*t(i))+sin(2*pi*3*t(i))+sin(2*pi*5*t(i)));
36:     if x(1,i) < -6
37:         d(1,i) = -1;
38:     elseif x(1,i) > 6
39:         d(1,i) = 1;
40:     else d(1,i) = x(1,i)/6;
41:     end
42:     fprintf('Iteração %d de %d\n', i, length(t));
43: end
44:
45: P = x;
46: T = d;
47:
48: save P P;
49: save T T;

```

## Apêndice 2

Código-fonte em MATLAB da função geradora de padrões referentes a não-linearidade de zona morta tipo I

```

1: % Geração de padrões de entrada para a rede MLP (backpropagation)
2: % Simulação de Zona Morta Tipo I
3: % x = sinal de entrada
4: % d = saída desejada
5:
6: clear; close all;
7:
8: C = 0;
9: while C == 0
10:     clc;
11:
12:     NI = input('No. de Iterações [Padrao 100]: ');
13:     if isempty(NI)
14:         NI = 100;
15:     end
16:
17:     TA = input('Taxa de Amostragem/Iteração [Padrao 100]: ');
18:     if isempty(TA)
19:         TA = 100;
20:     end
21:
22:     if NI/TA > 1
23:         P_repete = input('Haverá padrões repetidos! Continuar? S/N [Padrão N]: ','s');
24:         if (P_repete == 'S' | P_repete == 's')
25:             C = 1;
26:         end
27:     else C = 1;
28:     end
29: end
30: TA=1/TA;
31:
32: t = 0:TA:NI;
33:
34: for i=1:length(t)
35:     x(1,i) = 2.0 .* (cos(2*pi*3.5*t(i))*sin(2*pi*5*t(i))+cos(2*pi*29*t(i))+sin(2*pi*.33*t(i))+sin(2*pi*3*t(i))+sin(2*pi*5*t(i)));
36:     if abs(x(1,i)) <= 3
37:         d(1,i) = 0;
38:     elseif abs(x(1,i)) > 3;
39:         d(i) = x(1,i);
40:     end
41:     fprintf('Iteração %d de %d\n', i, length(t));
42: End
43:
44: P = x;
45: T = d;
46:
47: save P P;
48: save T T;

```

### Apêndice 3

Código-fonte em MATLAB da função geradora de padrões referentes a não-linearidade de zona morta tipo II

```

1: % Geração de padrões de entrada para a rede MLP (backpropagation)
2: % Simulação de Zona Morta Tipo II
3: % x = sinal de entrada
4: % d = saída desejada
5:
6: clear; close all;
7:
8: C = 0;
9: while C == 0
10:     clc;
11:
12:     NI = input('No. de Iterações [Padrao 100]: ');
13:     if isempty(NI)
14:         NI = 100;
15:     end
16:
17:     TA = input('Taxa de Amostragem/Iteração [Padrao 100]: ');
18:     if isempty(TA)
19:         TA = 100;
20:     end
21:
22:     if NI/TA > 1
23:         P_repete = input('Haverá padrões repetidos! Continuar? S/N [Padrão N]: ');
24:         if (P_repete == 'S' | P_repete == 's')
25:             C = 1;
26:         end
27:         else C = 1;
28:     end
29: end
30: TA=1/TA;
31:
32: t = 0:TA:NI;
33:
34: for i=1:length(t)
35:     x(1,i) = 2.0 .* (cos(2*pi*3.5*t(i))*sin(2*pi*5*t(i))+cos(2*pi*29*t(i))+sin(2*pi*.33*t(i))+sin(2*pi*3*t(i))+sin(2*pi*5*t(i)));
36:     if x(1,i) <= -4
37:         d(1,i) = x(1,i) + 4;
38:     elseif x(1,i) > 4;
39:         d(1,i) = x(1,i) - 4;
40:     else d(1,i) = 0;
41:     end
42:     fprintf('Iteração %d de %d\n', i, length(t));
43: end
44:
45: P = x;
46: T = d;
47:
48: save P P;
49: save T T;

```

## Apêndice 4

Código-fonte em MATLAB da função geradora de padrões referentes a não-linearidade de saturação com zona morta

```

1: % Geração de padrões de entrada para a rede MLP (backpropagation)
2: % Simulação de Saturação com Zona Morta
3: % x = sinal de entrada
4: % d = saída desejada
5:
6: clear; close all;
7:
8: C = 0;
9: while C == 0
10:     clc;
11:
12:     NI = input('No. de Iterações [Padrao 100]: ');
13:     if isempty(NI)
14:         NI = 100;
15:     end
16:
17:     TA = input('Taxa de Amostragem/Iteração [Padrao 100]: ');
18:     if isempty(TA)
19:         TA = 100;
20:     end
21:
22:     if NI/TA > 1
23:         P_repete = input('Haverá padrões repetidos! Continuar? S/N [Padrão N]: ');
24:         if (P_repete == 'S' | P_repete == 's')
25:             C = 1;
26:         end
27:         else C = 1;
28:     end
29: end
30: TA=1/TA;
31:
32: t = 0:TA:NI;
33:
34: for i=1:length(t)
35:     x(1,i) = 2.0 .* (cos(2*pi*3.5*t(i))*sin(2*pi*5*t(i))+cos(2*pi*29*t(i))+sin(2*pi*.33*t(i))+sin(2*pi*3*t(i))+sin(2*pi*5*t(i)));
36:     if x(1,i) > 6
37:         d(1,i) = 1;
38:     elseif x(1,i) < -6
39:         d(1,i) = -1;
40:     elseif (x(1,i) >= -6 & x(1,i) <= -3)
41:         d(1,i) = x(1,i)/3 + 1;
42:     elseif (x(1,i) >= 3 & x(1,i) <= 6)
43:         d(1,i) = x(1,i)/3 - 1;
44:     else d(1,i) = 0;
45:     end
46:     fprintf('Iteração %d de %d\n', i, length(t));
47: end
48:
49: P = x;
50: T = d;
51:
52: save P P;
53: save T T;

```

## Apêndice 5

Código-fonte em MATLAB da função geradora de padrões referentes a não-linearidade de liga-desliga

```

1: % Geração de padrões de entrada para a rede MLP (backpropagation)
2: % Simulação de On/Off
3: % x = sinal de entrada
4: % d = saída desejada
5:
6: clear; close all;
7:
8: C = 0;
9: while C == 0
10:     clc;
11:
12:     NI = input('No. de Iterações [Padrao 100]: ');
13:     if isempty(NI)
14:         NI = 100;
15:     end
16:
17:     TA = input('Taxa de Amostragem/Iteração [Padrao 100]: ');
18:     if isempty(TA)
19:         TA = 100;
20:     end
21:
22:     if NI/TA > 1
23:         P_repete = input('Haverá padrões repetidos! Continuar? S/N [Padrão N]: ','s');
24:         if (P_repete == 'S' | P_repete == 's')
25:             C = 1;
26:         end
27:         else C = 1;
28:     end
29: end
30: TA=1/TA;
31:
32: t = 0:TA:NI;
33:
34: for i=1:length(t)
35:     x(1,i) = 2.0 .* (cos(2*pi*3.5*t(i))*sin(2*pi*5*t(i))+cos(2*pi*29*t(i))+sin(2*pi*.33*t(i))+sin(2*pi*3*t(i))+sin(2*pi*5*t(i)));
36:     if x(i) > 0
37:         d(i) = 1;
38:     else
39:         d(i) = -1;
40:     end
41:     fprintf('Iteração %d de %d\n', i, length(t));
42: end
43:
44: P = x;
45: T = d;
46:
47: save P P;
48: save T T;

```



## Apêndice 6

Código-fonte em MATLAB da função geradora de padrões referentes a não-linearidade de liga-desliga com histerese

```

1: % Geração de padrões de entrada para a rede MLP (backpropagation)
2: % Simulação de On/Off com Histerese
3: % x = sinal de entrada
4: % d = saída desejada
5:
6: clear; close all;
7:
8: C = 0;
9: while C == 0
10:     clc;
11:
12:     NI = input('No. de Iterações [Padrao 105]: ');
13:     if isempty(NI)
14:         NI = 105;
15:     end
16:
17:     TA = input('Taxa de Amostragem/Iteração [Padrao 200]: ');
18:     if isempty(TA)
19:         TA = 200;
20:     end
21:
22:     if NI/TA > 1
23:         P_repete = input('Haverá padrões repetidos! Continuar? S/N [Padrão N]: ','s');
24:         if (P_repete == 'S' | P_repete == 's')
25:             C = 1;
26:         end
27:         else C = 1;
28:     end
29: end
30: TA=1/TA;
31:
32: t = 0:TA:NI;
33:
34: dtemp = 1;
35: for i=1:length(t)
36:     u(1,i) = 2.0 .* (cos(2*pi*3.5*t(i))*sin(2*pi*5*t(i))+cos(2*pi*29*t(i))+sin(2*pi*.33*t(i))+sin(2*pi*3*t(i))+sin(2*pi*5*t(i)));
37:     u(2,i) = dtemp;
38:     if u(1,i) >= 8 & u(2,i) == -1
39:         d(i) = 1;
40:     elseif u(1,i) <= -8 & u(2,i) == 1
41:         d(i) = -1;
42:     else d(1,i) = dtemp;
43:     end
44:     dtemp = d(1,i);
45:     fprintf('Iteração %d de %d\n', i, length(t));
46: end
47:
48: x(1,:) = (u(1,:));
49: x(2,:) = (u(2,:));
50:
51: P = x;
52: T = d;
53:
54: save P P;
55: save T T;

```

## Apêndice 7

Código-fonte em MATLAB da função geradora de padrões referentes a não-linearidade de liga-desliga com zona morta

```

1: % Geração de padrões de entrada para a rede MLP (backpropagation)
2: % Simulação de On/Off com Zona Morta
3: % x = sinal de entrada
4: % d = saída desejada
5:
6: clear; close all;
7:
8: C = 0;
9: while C == 0
10:     clc;
11:
12:     NI = input('No. de Iterações [Padrao 100]: ');
13:     if isempty(NI)
14:         NI = 100;
15:     end
16:
17:     TA = input('Taxa de Amostragem/Iteração [Padrao 100]: ');
18:     if isempty(TA)
19:         TA = 100;
20:     end
21:
22:     if NI/TA > 1
23:         P_repete = input('Haverá padrões repetidos! Continuar? S/N [Padrão N]: ','s');
24:         if (P_repete == 'S' | P_repete == 's')
25:             C = 1;
26:         end
27:         else C = 1;
28:     end
29: end
30: TA=1/TA;
31:
32: t = 0:TA:NI;
33:
34: for i=1:length(t)
35:     x(1,i) = 2.0 .* (cos(2*pi*3.5*t(i))*sin(2*pi*5*t(i))+cos(2*pi*29*t(i))+sin(2*pi*.33*t(i))+sin(2*pi*3*t(i))+sin(2*pi*5*t(i)));
36:     if x(i) < -3
37:         d(i) = -1;
38:     elseif x(i) > 3
39:         d(i) = 1;
40:     else d(i) = 0;
41:     end
42:     fprintf('Iteração %d de %d\n', i, length(t));
43: end
44:
45: P = x;
46: T = d;
47:
48: save P P;
49: save T T;

```

## Apêndice 8

### Código-fonte em MATLAB para simulação da RNA

```

1: % Implementação da Rede MLP (backpropagation com gradiente descendente)
2: % Usando as funções built-in (internas) do Matlab
3: %
4: % X = Vetor de entrada
5: % d = Saída desejada (escalar)
6: % WW = Matriz de Pesos Entrada -> Camada Oculta
7: % MM = Matriz de Pesos Camada Oculta -> Camada de Saída
8: % eta = Taxa de Aprendizagem
9: % mom = Constante de Momento
10:
11: clear; close all;
12:
13: tic; % Inicia Contador de Tempo
14:
15: % Carrega DADOS
16: %=====
17: load P;
18: load T;
19:
20: dados=P; % Vetores (padrões) de entrada
21: alvos=T; % Saídas desejadas correspondentes
22:
23: clear P; % Libera espaço em memória
24: clear T;
25:
26: % Determina o tamanho dos vetores
27: [LinD ColD]=size(dados);
28: [LinA ColA]=size(alvos);
29:
30: time1=toc; % Interrompe Contador de Tempo;
31:
32: % MENU DE OPÇÕES PARA SIMULAÇÃO
33: %=====
34: clc;
35:
36: fprintf('No. de Neuronios da Camada de Entrada: %d\n', LinD);
37:
38: fprintf('No. de Neuronios da Camada de Saída: %d\n', LinA);
39: No = LinA;
40:
41: Nh = input('No. de Neuronios da Camada Escondida [Padrao: Regra de Kolmogorov]: ');
42: if isempty(Nh)
43:     Nh = (2 * LinD) + 1;
44:     disp(Nh);
45: end
46:
47: eta = input('Taxa de Aprendizagem [Padrao 0.1]: ');
48: if isempty(eta)
49:     eta = 0.1;
50: end
51:
52: mom = input('Constante de Momento [Padrao 0.75]: ');

```

```

53:   if isempty(mom)
54:       mom = 0.75;
55:   end
56:
57:   Ne = input('No. de Epocas (> 5) [Padrao 100]: ');
58:   if isempty(Ne)
59:       Ne = 100;
60:   end
61:   % No. de épocas superior a 5 devido ao primeiro teste
62:
63:   ptrn = input('Porcentagem para Treino [Padrao 80]: ');
64:   if isempty(ptrn)
65:       ptrn = .80;
66:   else ptrn = ptrn/100;
67:   end
68:
69:   op_salvarel = input('Salvar Relatorio da Simulacao? S/N [Padrão S]: ','s');
70:   if (op_salvarel == 'N' | op_salvarel == 'n')
71:       salvarel = 0;
72:       arqrel = 'temp';
73:   else salvarel = 1;
74:       arqrel = input('Digite o Nome do Arquivo de Relatorio: ','s');
75:   end
76:
77:   op_salvados = input('Salvar Dados da Simulacao? S/N [Padrão S]: ','s');
78:   if (op_salvados == 'N' | op_salvados == 'n')
79:       salvados = 0;
80:   else salvados = 1;
81:       C = strcmp(arqrel,'temp');
82:       if C == 1
83:           arqdados = input('Digite o Nome do Arquivo de dados: ','s');
84:       else arqdados = arqrel;
85:       end
86:   end
87:
88:   ext = '.txt'; % Salva relatório final em arquivo texto
89:   arqrel = strcat(arqrel, ext);
90:
91:   %=====
92:   tic; % Reinicia Contador de Tempo
93:
94:   % Normaliza componentes para máximo +1 e mínimo -1
95:   for i=1:LinD
96:       Dmax=max(dados(i,:));
97:       Dmin=min(dados(i,:));
98:       Dn(i,:)=[dados(i,:)-((Dmax+Dmin)/2)]./[(Dmax-Dmin)/2];
99:   end
100:  for i=1:LinA
101:      Amax=max(alvos(i,:));
102:      Amin=min(alvos(i,:));
103:      An(i,:)=[alvos(i,:)-((Amax+Amin)/2)]./[(Amax-Amin)/2];
104:  end
105:
106:  I=randperm(ColD);
107:  Dn=Dn(:,I);
108:  An=An(:,I); % Embaralha saídas desejadas p/ manter correspondência com vetor de entrada

```

```

109:
110: % Define tamanho dos conjuntos de teste
111: ptst=1-ptrn; % Porcentagem usada para teste
112:
113: J=floor(ptrn*ColD);
114:
115: % Vetores para treinamento e saídas desejadas correspondentes
116: P = Dn(:, 1:J); T1 = An(:, 1:J);
117: [LinP ColP]=size(P); % Tamanho da matriz de vetores de treinamento
118:
119: % Vetores para teste e saídas desejadas correspondentes
120: Q = Dn(:, J+1:end); T2 = An(:, J+1:end);
121: [LinQ ColQ]=size(Q); % Tamanho da matriz de vetores de teste
122:
123: % DEFINE ARQUITETURA DA REDE
124: %=====
125:
126: % Inicia matrizes de pesos
127: WW=0.1*rand(Nh,LinP+1); % Pesos entrada -> camada oculta
128: WW_old=WW; % Necessário para termo de momento
129:
130: MM=0.1*rand(No,Nh+1); % Pesos camada oculta -> camada de saída
131: MM_old = MM; % Necessário para termo de momento
132:
133: Bias=1; % Bias positivo
134:
135: if Bias == 0
136:     Np = (LinD*Nh)+(Nh*No)
137:     else Np = (LinD*Nh+Nh)+(Nh*No+No);
138: end
139:
140: %% ETAPA DE TREINAMENTO
141:
142: FIM = 0; %Controla o LOOP principal
143: CE = 0; % Inicia contador de EQM de teste < 0.001
144:
145: while FIM == 0; % Inicio de LOOP principal
146:
147:     Nr=1; %Inicia contador de testes
148:
149:     TO=0; %Inicia contador de tendência de crescimento do EQM de teste
150:
151:     EQMMin=1000; % Inicia variáveis de menor EQM entre treinamento e teste
152:     EQM1Min=1000;
153:
154:     %WWL=[]; % Inicia matrizes de pesos e bias (última rodada somente)
155:     %MML=[];
156:
157:     for t=1:Ne, % Inicio de LOOP de treinamento
158:
159:         Epoca=t,
160:
161:         if Epoca == Ne % Condição para sair do LOOP principal
162:             MOT = 1;
163:             FIM = 1; % Assume critério de parada por término de épocas
164:         end

```

```

165:
166: I=randperm(CoIP); P=P(:,I); T1=T1(:,I); % Embaralha vetores de treinamento e saídas desejadas
167:
168: EQ=0;
169: for tt=1:CoIP, % Inicia LOOP de época de treinamento
170:     % CAMADA OCULTA
171:     X=[Bias; P(:,tt)]; % Constrói vetor de entrada com adição da entrada x0
172:     Ui = WW * X; % Ativação (net) dos neurônios da camada oculta
173:     Yi = (1-exp(-Ui))./(1+exp(-Ui)); % Saída entre [-1,1] -> função tangente hiperbólica
174:     %WWL(:,,tt)=[WW]; % Armazena evolução de todos os pesos e bias da última época válida
175:
176:     % CAMADA DE SAÍDA
177:     Y=[Bias; Yi]; % Constrói vetor de entrada (desta camada) com adição da entrada y0
178:     Uk = MM * Y; % Ativação (net) dos neurônios da camada de saída
179:     Ok = (1-exp(-Uk))./(1+exp(-Uk)); % Saída entre [-1,1] -> função tangente hiperbólica
180:     %MML(:,,tt)=[MM]; % Armazena evolução de todos os pesos e bias da última época válida
181:
182:     % Desnormalização das saídas p/ plotagem de gráfico
183:     T1Des(tt) = [T1(:,tt).*((Amax-Amin)/2)].+[(Amax+Amin)/2];
184:     T1Sai(tt) = [Ok.*((Amax-Amin)/2)].+[(Amax+Amin)/2];
185:
186:     % CALCULO DO ERRO
187:     Ek = T1(:,tt) - Ok; % Erro entre a saída desejada e a saída da rede
188:     EQ = EQ + 0.5*sum(Ek.^2); % soma do erro quadrático de todos os neurônios p/ VETOR DE ENTRADA
189:
190:     %%% CALCULO DOS GRADIENTES LOCAIS
191:     Dk = (1 - Ok.*Ok); % derivada da tangente hiperbólica (camada de saída)
192:     DDK = Ek.*Dk; % gradiente local (camada de saída)
193:
194:     Di = (1 - Yi.*Yi); % derivada da tangente hiperbólica (camada oculta)
195:     DDi = Di.*(MM(:,2:end))*DDk; % gradiente local (camada oculta)
196:
197:     % AJUSTE DOS PESOS - CAMADA DE SAÍDA
198:     MM_aux=MM;
199:     MM = MM + eta*DDk*Y' + mom*(MM - MM_old);
200:     MM_old=MM_aux;
201:
202:     % AJUSTE DOS PESOS - CAMADA OCULTA
203:     WW_aux=WW;
204:     WW = WW + eta*DDi*X' + mom*(WW - WW_old);
205:     WW_old=WW_aux;
206:
207:     %{
208:     if tt == CoIP % Armazena matrizes dos últimos pesos e bias de cada época
209:         WWLE(:,,t)=[WW];
210:         MMLE(:,,t)=[MM];
211:     end
212:     %}
213:
214: end % Fim de LOOP de época de treinamento
215:
216: % MÉDIA DO ERRO QUADRÁTICO P/ ÉPOCA
217: EQM(t)=EQ/CoIP;
218:
219: if EQM (t) < EQMMin
220:     EQMMin = EQM(t);

```



```

277:         end
278:         else CE = 0;
279:     end
280:
281:     % Critério de crescimento do EQM (3)
282:     if Nr > 1
283:         if (EQM1(Nr) > EQM1(Nr-1)) & (EQM1(Nr) > EQM(t))
284:             TO=TO+1;
285:             if TO == 3;
286:                 MOT = 3;
287:                 FIM = 1;
288:                 break;
289:             end
290:         else TO=0;
291:         end
292:     end
293:     Nr=Nr+1; % Incrementa Contador de Testes
294:
295:     end % Fim da condição para TESTE
296:
297:     %{
298:     if Epoca < Ne
299:         WWL=[]; % Zera matrizes de pesos caso não sejam da última rodada
300:         MML=[];
301:     end
302:     %}
303:
304:     end % Fim do LOOP de treinamento
305:
306:     %% ETAPA DE GENERALIZAÇÃO %%
307:     EQ2=0;
308:     OUT2=[];
309:     for tt=1:ColQ,
310:         % CAMADA OCULTA
311:         X=[Bias; Q(:,tt)]; % Constrói vetor de entrada com adição da entrada x0
312:         Ui = WW * X; % Ativação (net) dos neurônios da camada oculta
313:         Yi = (1-exp(-Ui))./(1+exp(-Ui)); % Saída entre [-1,1] - função tangente hiperbólica
314:
315:         % CAMADA DE SAÍDA
316:         Y=[Bias; Yi]; % Constrói vetor de entrada (desta camada) com adição da entrada y0
317:         Uk = MM * Y; % Ativação (net) dos neurônios da camada de saída
318:         Ok = (1-exp(-Uk))./(1+exp(-Uk)); % Saída entre [-1,1] - função tangente hiperbólica
319:         OUT2=[OUT2 Ok]; % Armazena saída da rede
320:
321:         GDes(tt) = [T2(:,tt).*((Amax-Amin)/2)].+[(Amax+Amin)/2]; % Desnormaliza saídas
322:         GSai(tt) = [Ok.*((Amax-Amin)/2)].+[(Amax+Amin)/2];
323:
324:         % Gradiente local da camada de saída
325:         Ek = T2(:,tt) - Ok; % Erro entre a saída desejada e a saída da rede
326:
327:         Dk = (1 - Ok.*Ok); % derivada da tangente hiperbólica
328:         Ddk = Ek.*Dk; % gradiente local
329:
330:         % ERRO QUADRATICO GLOBAL (todos os neurônios) POR VETOR DE ENTRADA
331:         EQ2 = EQ2 + 0.5*sum(Ek.^2);
332:

```



```

333:         % Gradiente local da camada oculta
334:         Di = (1 - Yi.*Yi); % derivada da tangente hiperbólica
335:         DDi = Di.*(MM(:,2:end))*DDk);
336:     end
337:
338:     % MÉDIA DO ERRO QUADRÁTICO COM REDE TREINADA (USANDO DADOS DE TREINAMENTO)
339:     EQM2=EQ2/ColQ;
340:
341:     % Verifica se deve sair do LOOP principal
342:     if MOT == 3
343:         %WW = WWLE(:,t-(3*It)); % Retorna com pesos e bias imediatamente antes do crescimento do EQM de teste
344:         %MM = MMLE(:,t-(3*It));
345:         break;
346:     else if MOT == 2 & EQM2 < 0.001
347:         break;
348:     else if Epoca < Ne % Retorna para LOOP principal
349:         CE = 0;
350:         %WWL=[];
351:         %MML=[];
352:     end
353: end
354: end
355:
356: end % Fim do LOOP principal
357:
358: % CALCULA TAXA DE ACERTO
359: count_OK=0; % Contador de acertos
360: for t=1:ColQ,
361:     [T2max iT2max]=max(T2(:,t)); % Índice da saída desejada de maior valor
362:     [OUT2_max iOUT2_max]=max(OUT2(:,t)); % Índice do neurônio cuja saída é a maior
363:     if iT2max==iOUT2_max, % Conta acerto se os dois índices coincidem
364:         count_OK=count_OK+1;
365:     end
366: end
367: %Taxa de acerto global
368: TX_OK=100*(count_OK/ColQ);
369:
370: time2=toc; % Finaliza Contador de Tempo;
371: s=time1+time2; % Acumula em s os tempos de processamento
372:
373: % Conversão de s em h, m e s.
374: hora=fix(s/3600);
375: min=fix(mod(s,3600)/60);
376: seg=fix(mod(mod(s,3600),60));
377:
378: %% RELATÓRIO FINAL %%
379: %clc;
380: fid = fopen(arqrel,'wt');
381: fprintf(fid,'===== R E L A T Ó R I O =====\n');
382:
383: if salvarel == 1
384:     fprintf(fid,'\nNome do Arquivo de Relatório: ... %s\n',arqrel);
385: end
386:
387: if salvadados == 1
388:     fprintf(fid,'\nNome do Arquivo de Dados: ..... %s.mat\n',arqdados);

```

```

389: end
390:
391: fprintf(fid,'\nConfiguracao da Rede: ... %d : %d : %d\n', LinD, Nh, No);
392: fprintf(fid,'No. de Parametros Livres: %d\n', Np);
393: fprintf(fid,'Taxa de Aprendizagem: ... %2.4f\n', eta);
394: fprintf(fid,'Constante de Momento: ... %2.4f\n', mom);
395: fprintf(fid,'No. de Epocas: ..... %d\n', Epoca);
396: fprintf(fid,'No. de Testes: ..... %d\n', Nr-1);
397: fprintf(fid,'\n');
398: fprintf(fid,'Tamanho do vetor de padrões entradas-saidas: %d (100.0%%)\n', ColD);
399: fprintf(fid,'Tamanho do vetor de treinamento/teste: ..... %d (%4.1f%%)\n', ColP, ptrn*100);
400: fprintf(fid,'Tamanho do vetor de generalizacao: ..... %d (%4.1f%%)\n', ColQ, p1st*100);
401: fprintf(fid,'\nEQM de Treinamento: ..... %2.9f\n',EQM(Epoca));
402: fprintf(fid,'EQM de Teste: ..... %2.9f\n',EQM1(Nr-1));
403: fprintf(fid,'EQM de Generalizacao: ... %2.9f\n',EQM2);
404: criterio = ini2str(MOT);
405:     switch lower(criterio)
406:         case {'1'}
407:             fprintf(fid,'\nCritério de Parada: ..... (1) Número máximo de épocas\n');
408:         case '2'
409:             fprintf(fid,'\nCritério de Parada: ..... (2)EQM de teste < 0.001\n');
410:         case '3'
411:             fprintf (fid,'\nCritério de Parada: ..... (3)EQM de teste com tendencia de crescimento\n');
412:     end
413: fprintf(fid,'\nTempo da Simulação: ..... %dh %dm %ds\n', hora, min, seg);
414: fprintf(fid,'\n=====');
415: fclose(fid);
416: type(arqrel);
417:
418: if salvarel == 0
419:     delete temp.txt;
420: end
421:
422: if salvados == 1
423:     fprintf('Aguarde! Salvando dados... ');
424:     save (arqdados);
425:     fprintf('Pronto!\n');
426: end

```

## Apêndice 9

Tabela padronizada para disposição dos resultados obtidos pelas simulações

Nº DA SIMULAÇÃO	Nº DE PADRÕES	Nº DE NEURÔNIOS POR CAMADA	Nº DE ÉPOCAS	CRITÉRIO DE PARADA	EQM DE TREINAMENTO	EQM DE TESTE	EQM DE GENERALIZAÇÃO	TEMPO DA SIMULAÇÃO
1	5001	1:3:1						
2	10001	1:2:1						
3	10001	1:3:1						
4	10001	1:4:1						
5	10001	1:10:1						
6	20001	1:3:1						

## Apêndice 10

## Exemplo de relatório de parâmetros utilizados e resultados obtidos

```
===== R E L A T Ó R I O =====  
  
Nome do Arquivo de Relatório: ... satu_1.txt  
  
Nome do Arquivo de Dados: ..... satu_1.mat  
  
Configuracao da Rede: ... 1 : 3 : 1  
No. de Parametros Livres: 10  
Taxa de Aprendizagem: ... 0.1000  
Constante de Momento: ... 0.7500  
No. de Epocas: ..... 20  
No. de Testes: ..... 3  
  
Tamanho do vetor de padrões entradas-saidas: 5001 (100.0%)  
Tamanho do vetor de treinamento/teste: ..... 4000 (80.0%)  
Tamanho do vetor de generalizacao: ..... 1001 (20.0%)  
  
EQM de Treinamento: ..... 0.000105666  
EQM de Teste: ..... 0.000127878  
EQM de Generalizacao: ... 0.000114124  
  
Critério de Parada: ..... (2)EQM de teste < 0.001  
  
Tempo da Simulação: ..... 0h 0m 7s  
  
=====
```

## Apêndice 11

Código-fonte em MATLAB para plotagem dos EQMs da fase de treinamento/teste, com indicação do EQM de generalização

```

1: % Gráfico de evolução do EQM
2:
3: %%% PLOTA EQM DE TREINAMENTO
4: %%% MARCA EQM DE TESTE
5: %%% MARCA EQM DE GENERALIZAÇÃO
6:
7: x1=[1:1:Epoca];
8: x2=[5:5:Epoca];
9:
10: if EQMMin < EQM1Min
11:     MEQMMin = EQMMin;
12: else MEQMMin = EQM1Min;
13: End
14: MEQMMax = max(max(EQM),max(EQM1));
15:
16: figure(1);
17: hold off
18: plot(x1,EQM,'r-','LineWidth',3);
19: hold on
20: plot(x2,EQM1,'o','LineWidth',2,'MarkerSize',8,'MarkerEdgeColor','k','MarkerFaceColor','b');
21: plot(Epoca,EQM2,'go','LineWidth',2,'MarkerSize',8,'MarkerEdgeColor','k','MarkerFaceColor','g');
22: axis([1 Epoca .9*MEQMMin 1.1*MEQMMax]);
23: box on;
24: grid on;
25: title(['\bf \fontsize{12} EVOLUÇÃO DO ERRO QUADRÁTICO MÉDIO DURANTE TREINAMENTO/TESTE'];...
26:     ['\fontname{Courier New}\rm(Obs.: treinamento encerrado na ',int2str(Epoca),'^a época)']);
27: legend(['\bf \fontname{Courier New} \fontsize{11}EQM DO TREINAMENTO: ',num2str(EQM(Epoca))],...
28:     ['\bf \fontname{Courier New} \fontsize{11}EQM DO TESTE: ',num2str(EQM1(Nr-1))],...
29:     ['\bf \fontname{Courier New} \fontsize{11}EQM DO GENERALIZAÇÃO: ',num2str(EQM2)] );
30: xlabel('Época');
31: ylabel('EQM');
32: hold off

```

## Apêndice 12

Código-fonte em MATLAB para comparação entre o sinal de saída da não-linearidade e o sinal emitido pela RNA

```

1: % Geração de gráficos comparativos das fases de treinamento/teste e generalização
2:
3: % Define o no. de amostras a serem plotadas
4: Amostras = input('No. de Amostras para plotagem [Padrao 100]: ');
5: if isempty(Amostras)
6:     Amostras = 100;
7: end
8:
9: T1f = length(T1); % Final do conjunto de treinamento/teste
10: T1i = T1f - Amostras; % Início do conjunto de treinamento/teste
11:
12: T2f = length(T2); % Final do conjunto de generalização
13: T2i = T2f - Amostras; % Início do conjunto de generalização
14:
15: %%% PLOTA FASE DE TREINAMENTO
16:
17: figure(2);
18: plot(T1Des,'b-','LineWidth',2);
19: axis([T1i T1f Amin+.1*Amin Amax+.1*Amax]) % YMIN e YMAX acrescidos em 10%
20: hold on
21: plot (T1Sai,'r-','LineWidth',2);
22: title(['\bf \fontsize{12} COMPARAÇÃO ENTRE SINAIS DE SAÍDA OBTIDOS NO TREINAMENTO^{*}'];...
23:     ['\fontname{Courier New}\rm \Ncirc de Épocas: ',int2str(Epoca),' | ', '\Ncirc de Padrões: ',...
24:     int2str(CoLA),' | ', '\Ncirc de Neurônios por Camada: ',int2str(LinD),':',...
25:     int2str(Nh),':',int2str(LinA)];...
26:     ['EQM: ',num2str(EQM(Epoca))]);
27: legend('Sinal Desejado','Sinal da RNA');
28: xlabel('Amostra');
29: ylabel('Saída');
30: hold off
31: grid on
32: text(T1f-40, Amin*1.25,'*\fontsize{10}it Resultado da última época de treinamento');
33:
34: %%% PLOTA FASE DE TESTE
35: figure(3);
36: plot(T2Des,'b-','LineWidth',2);
37: axis([T1i T1f Amin+.1*Amin Amax+.1*Amax]) % YMIN e YMAX acrescidos em 10%
38: hold on
39: plot (T2Sai,'r-','LineWidth',2);
40: title(['\bf \fontsize{12} COMPARAÇÃO ENTRE SINAIS DE SAÍDA OBTIDOS NO TESTE^{*}'];...
41:     ['\fontname{Courier New}\rm \Ncirc de Épocas: ',int2str(Epoca),' | ', '\Ncirc de Padrões: ',...
42:     int2str(CoLA),' | ', '\Ncirc de Neurônios por Camada: ',int2str(LinD),':',...
43:     int2str(Nh),':',int2str(LinA)];...
44:     ['EQM: ',num2str(EQM1(Nr-1))]);
45: legend('Sinal Desejado','Sinal da RNA');
46: xlabel('Amostra');
47: ylabel('Saída');
48: hold off

```

```
49: grid on
50: text(T1f-35, Amin*1.25, '*\fontsize{10}it Resultado do último teste');
51:
52: %%% PLOTA FASE DE GENERALIZAÇÃO
53: figure(4);
54: plot(GDes, 'b-', 'LineWidth', 2);
55: axis([T2i T2f Amin+.1*Amin Amax+.1*Amax])
56: hold on
57: plot(GSai, 'r-', 'LineWidth', 2);
58: title({'\bf\fontsize{12}COMPARAÇÃO ENTRE SINAIS DE SAÍDA OBTIDOS NA GENERALIZAÇÃO};...
59:   [\fontname{Courier New}\rm N\circ de Épocas: ', int2str(Epoca), ' | ', '\N\circ de Padrões: ', ...
60:   int2str(CoIA), ' | ', '\N\circ de Neurônios por Camada: ', int2str(LinD), ':', ...
61:   int2str(Nh), ':', int2str(LinA)];...
62:   ['EQM: ', num2str(EQM2)]});
63: legend('Sinal Desejado', 'Sinal da RNA');
64: xlabel('Amostra');
65: ylabel('Saída');
66: hold off
67: grid on
```

### Apêndice 13

Código-fonte em MATLAB utilizado plotagem dos pesos e bias, referentes à última época de treinamento, para redes com 1 neurônio na camada de entrada, 3 na camada escondida e 1 na camada de saída

```

1:  %{
2:    Plotagem da evolução dos pesos e bias durante a última época da
3:    fase de treinamento
4:
5:    Condições: No. de neurônios da camada de entrada: 1
6:              No. de neurônios da camada de entrada: 3
7:              No. de neurônios da camada de entrada: 1
8:  %}
9:
10: %%% Pesos e Bias da camada escondida
11:
12: WWL_b1 (1,:) = WWL(1,1,:);
13: WWL_w11(1,:) = WWL(1,2,:);
14: WWL_b2 (1,:) = WWL(2,1,:);
15: WWL_w21(1,:) = WWL(2,2,:);
16: WWL_b3 (1,:) = WWL(3,1,:);
17: WWL_w31(1,:) = WWL(3,2,:);
18:
19: %%% Pesos e Bias da camada de saída
20: MML_b1 (1,:) = MML(1,1,:);
21: MML_w11(1,:) = MML(1,2,:);
22: MML_w12(1,:) = MML(1,3,:);
23: MML_w13(1,:) = MML(1,4,:);
24:
25: % Plotagem dos Pesos e Bias da camada escondida
26:
27: figure(4)
28:
29: subplot(3,2,1), plot(WWL_w11,'r-');
30: title('\bfPESOS DA CAMADA ESCONDIDA');
31: legend('Peso (1,1)', 'Location', 'Best');
32: xlabel('Amostra');
33: ylabel('Amplitude');
34: grid on;
35:
36: subplot(3,2,3), plot(WWL_w21,'r-');
37: legend('Peso (2,1)', 'Location', 'Best');
38: xlabel('Amostra');
39: ylabel('Amplitude');
40: grid on;
41:
42: subplot(3,2,5), plot(WWL_w31,'r-');
43: legend('Peso (3,1)', 'Location', 'Best');
44: xlabel('Amostra');
45: ylabel('Amplitude');
46: grid on;
47:

```



```
48: subplot(3,2,2), plot(WWL_b1,'b-');
49: title('\bfBIAS DA CAMADA ESCONDIDA');
50: legend('Bias (1)', 'Location', 'Best');
51: xlabel('Amostra');
52: ylabel('Amplitude');
53: grid on;
54:
55: subplot(3,2,4), plot(WWL_b2,'b-');
56: legend('Bias (2)', 'Location', 'Best');
57: xlabel('Amostra');
58: ylabel('Amplitude');
59: grid on;
60:
61: subplot(3,2,6), plot(WWL_b3,'b-');
62: legend('Bias (3)', 'Location', 'Best');
63: xlabel('Amostra');
64: ylabel('Amplitude');
65: grid on;
66:
67: % Plotagem dos Pesos e Bias da camada escondida
68:
69: figure(5)
70:
71: subplot(3,2,1), plot(MML_w11,'r-');
72: title('\bfPESOS DA CAMADA DE SAÍDA');
73: legend('Peso (1,1)', 'Location', 'Best');
74: xlabel('Amostra');
75: ylabel('Amplitude');
76: grid on;
77:
78: subplot(3,2,3), plot(MML_w12,'r-');
79: legend('Peso (2,1)', 'Location', 'Best');
80: xlabel('Amostra');
81: ylabel('Amplitude');
82: grid on;
83:
84: subplot(3,2,5), plot(MML_w13,'r-');
85: legend('Peso (3,1)', 'Location', 'Best');
86: xlabel('Amostra');
87: ylabel('Amplitude');
88: grid on;
89:
90: subplot(3,2,4), plot(MML_b1,'b-');
91: title('\bfBIAS DA CAMADA DE SAÍDA');
92: legend('Bias (1)', 'Location', 'Best');
93: xlabel('Amostra');
94: ylabel('Amplitude');
95: grid on;
```

## Apêndice 14

Código-fonte em MATLAB utilizado plotagem dos pesos e bias, referentes à última época de treinamento, para redes com 2 neurônios na camada de entrada, 5 na camada escondida e 1 na camada de saída

```

1:  %{
2:    Plotagem da evolução dos pesos e bias durante a última época da
3:    fase de treinamento
4:
5:    Condições: No. de neurônios da camada de entrada: 2
6:              No. de neurônios da camada de entrada: 5
7:              No. de neurônios da camada de entrada: 1
8:  %}
9:
10: %%% Pesos e Bias da camada escondida
11:
12: WWL_b1 (1,:) = WWL(1,1,:);
13: WWL_b2 (1,:) = WWL(2,1,:);
14: WWL_b3 (1,:) = WWL(3,1,:);
15: WWL_b4 (1,:) = WWL(4,1,:);
16: WWL_b5 (1,:) = WWL(5,1,:);
17:
18: WWL_w11(1,:) = WWL(1,2,:);
19: WWL_w21(1,:) = WWL(2,2,:);
20: WWL_w31(1,:) = WWL(3,2,:);
21: WWL_w41(1,:) = WWL(4,2,:);
22: WWL_w51(1,:) = WWL(5,2,:);
23:
24: WWL_w12(1,:) = WWL(1,3,:);
25: WWL_w22(1,:) = WWL(2,3,:);
26: WWL_w32(1,:) = WWL(3,3,:);
27: WWL_w42(1,:) = WWL(4,3,:);
28: WWL_w52(1,:) = WWL(5,3,:);
29:
30: %%% Pesos e Bias da camada de saída
31: MML_b1 (1,:) = MML(1,1,:);
32: MML_w11(1,:) = MML(1,2,:);
33: MML_w12(1,:) = MML(1,3,:);
34: MML_w13(1,:) = MML(1,4,:);
35: MML_w14(1,:) = MML(1,5,:);
36: MML_w15(1,:) = MML(1,6,:);
37:
38: % Plotagem dos Pesos e Bias da camada escondida
39:
40: figure(4)
41:
42: subplot(5,3,1), plot(WWL_w11,'r-');
43: title('\bfPESOS DA CAMADA ESCONDIDA - N1');
44: legend('Peso (1,1)', 'Location', 'Best');
45: xlabel('Amostra');
46: ylabel('Amplitude');
47: grid on;

```

```
48:
49: subplot(5,3,4), plot(WWL_w21,'r-');
50: legend('Peso (2,1)', 'Location', 'Best');
51: xlabel('Amostra');
52: ylabel('Amplitude');
53: grid on;
54:
55: subplot(5,3,7), plot(WWL_w31,'r-');
56: legend('Peso (3,1)', 'Location', 'Best');
57: xlabel('Amostra');
58: ylabel('Amplitude');
59: grid on;
60:
61: subplot(5,3,10), plot(WWL_w41,'r-');
62: legend('Peso (4,1)', 'Location', 'Best');
63: xlabel('Amostra');
64: ylabel('Amplitude');
65: grid on;
66:
67: subplot(5,3,13), plot(WWL_w51,'r-');
68: legend('Peso (5,1)', 'Location', 'Best');
69: xlabel('Amostra');
70: ylabel('Amplitude');
71: grid on;
72:
73: subplot(5,3,2), plot(WWL_w12,'g-');
74: title('\bfPESOS DA CAMADA ESCONDIDA - N2');
75: legend('Peso (1,2)', 'Location', 'Best');
76: xlabel('Amostra');
77: ylabel('Amplitude');
78: grid on;
79:
80: subplot(5,3,5), plot(WWL_w22,'g-');
81: legend('Peso (2,2)', 'Location', 'Best');
82: xlabel('Amostra');
83: ylabel('Amplitude');
84: grid on;
85:
86: subplot(5,3,8), plot(WWL_w32,'g-');
87: legend('Peso (3,2)', 'Location', 'Best');
88: xlabel('Amostra');
89: ylabel('Amplitude');
90: grid on;
91:
92: subplot(5,3,11), plot(WWL_w42,'g-');
93: legend('Peso (4,2)', 'Location', 'Best');
94: xlabel('Amostra');
95: ylabel('Amplitude');
96: grid on;
97:
98: subplot(5,3,14), plot(WWL_w52,'g-');
99: legend('Peso (5,2)', 'Location', 'Best');
```

```
100: xlabel('Amostra');
101: ylabel('Amplitude');
102: grid on;
103:
104: subplot(5,3,3), plot(WWL_b1,'b-');
105: title('\bfBIAS DA CAMADA ESCONDIDA');
106: legend('Bias (1)', 'Location', 'Best');
107: xlabel('Amostra');
108: ylabel('Amplitude');
109: grid on;
110:
111: subplot(5,3,6), plot(WWL_b2,'b-');
112: legend('Bias (2)', 'Location', 'Best');
113: xlabel('Amostra');
114: ylabel('Amplitude');
115: grid on;
116:
117: subplot(5,3,9), plot(WWL_b3,'b-');
118: legend('Bias (3)', 'Location', 'Best');
119: xlabel('Amostra');
120: ylabel('Amplitude');
121: grid on;
122:
123: subplot(5,3,12), plot(WWL_b4,'b-');
124: legend('Bias (4)', 'Location', 'Best');
125: xlabel('Amostra');
126: ylabel('Amplitude');
127: grid on;
128:
129: subplot(5,3,15), plot(WWL_b5,'b-');
130: legend('Bias (5)', 'Location', 'Best');
131: xlabel('Amostra');
132: ylabel('Amplitude');
133: grid on;
134:
135: % Plotagem dos Pesos e Bias da camada escondida
136:
137: figure(5)
138:
139: subplot(5,2,1), plot(MML_w11,'r-');
140: title('\bfPESOS DA CAMADA DE SAÍDA');
141: legend('Peso (1,1)', 'Location', 'Best');
142: xlabel('Amostra');
143: ylabel('Amplitude');
144: grid on;
145:
146: subplot(5,2,3), plot(MML_w12,'r-');
147: legend('Peso (2,1)', 'Location', 'Best');
148: xlabel('Amostra');
149: ylabel('Amplitude');
150: grid on;
151:
```

```
152: subplot(5,2,5), plot(MML_w13,'r-');
153: legend('Peso (3,1)', 'Location', 'Best');
154: xlabel('Amostra');
155: ylabel('Amplitude');
156: grid on;
157:
158: subplot(5,2,7), plot(MML_w14,'r-');
159: legend('Peso (4,1)', 'Location', 'Best');
160: xlabel('Amostra');
161: ylabel('Amplitude');
162: grid on;
163:
164: subplot(5,2,9), plot(MML_w15,'r-');
165: legend('Peso (5,1)', 'Location', 'Best');
166: xlabel('Amostra');
167: ylabel('Amplitude');
168: grid on;
169:
170: subplot(5,2,6), plot(MML_b1,'b-');
171: title('\bfBIAS DA CAMADA DE SAIDA');
172: legend('Bias (1)', 'Location', 'Best');
173: xlabel('Amostra');
174: ylabel('Amplitude');
175: grid on;
```

CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA  
CELSO SUCKOW DA FONSECA – CEFET/RJ

DIRETORIA DE PESQUISA E PÓS-GRADUAÇÃO  
COORDENADORIA DO PROGRAMA DE PÓS-GRADUAÇÃO EM TECNOLOGIA

DISSERTAÇÃO

APLICAÇÃO DE REDES NEURAIS ARTIFICIAIS EM  
IDENTIFICAÇÃO DE NÃO-LINEARIDADES RÍGIDAS

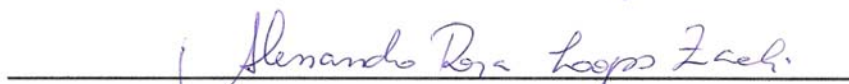
Wendel Furtado da Silva

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO PROGRAMA DE PÓS-GRADUAÇÃO EM TECNOLOGIA COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM TECNOLOGIA.


Data da Defesa: 06/06/2008

Aprovação:

  
Paulo Lúcio Silva de Aquino, D.Sc.

  
Alessandro Rosa Lopes Zachi, D.Sc.

  
Luciana Faletti Almeida, D. Sc.

  
Maria Luiza Fernandes Velloso, D. Sc.

# Livros Grátis

( <http://www.livrosgratis.com.br> )

Milhares de Livros para Download:

[Baixar livros de Administração](#)

[Baixar livros de Agronomia](#)

[Baixar livros de Arquitetura](#)

[Baixar livros de Artes](#)

[Baixar livros de Astronomia](#)

[Baixar livros de Biologia Geral](#)

[Baixar livros de Ciência da Computação](#)

[Baixar livros de Ciência da Informação](#)

[Baixar livros de Ciência Política](#)

[Baixar livros de Ciências da Saúde](#)

[Baixar livros de Comunicação](#)

[Baixar livros do Conselho Nacional de Educação - CNE](#)

[Baixar livros de Defesa civil](#)

[Baixar livros de Direito](#)

[Baixar livros de Direitos humanos](#)

[Baixar livros de Economia](#)

[Baixar livros de Economia Doméstica](#)

[Baixar livros de Educação](#)

[Baixar livros de Educação - Trânsito](#)

[Baixar livros de Educação Física](#)

[Baixar livros de Engenharia Aeroespacial](#)

[Baixar livros de Farmácia](#)

[Baixar livros de Filosofia](#)

[Baixar livros de Física](#)

[Baixar livros de Geociências](#)

[Baixar livros de Geografia](#)

[Baixar livros de História](#)

[Baixar livros de Línguas](#)

[Baixar livros de Literatura](#)  
[Baixar livros de Literatura de Cordel](#)  
[Baixar livros de Literatura Infantil](#)  
[Baixar livros de Matemática](#)  
[Baixar livros de Medicina](#)  
[Baixar livros de Medicina Veterinária](#)  
[Baixar livros de Meio Ambiente](#)  
[Baixar livros de Meteorologia](#)  
[Baixar Monografias e TCC](#)  
[Baixar livros Multidisciplinar](#)  
[Baixar livros de Música](#)  
[Baixar livros de Psicologia](#)  
[Baixar livros de Química](#)  
[Baixar livros de Saúde Coletiva](#)  
[Baixar livros de Serviço Social](#)  
[Baixar livros de Sociologia](#)  
[Baixar livros de Teologia](#)  
[Baixar livros de Trabalho](#)  
[Baixar livros de Turismo](#)