UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA QUÍMICA

ÁREA DE CONCENTRAÇÃO:
SISTEMAS DE PROCESSOS QUÍMICOS E INFORMÁTICA

DETECÇÃO E DIAGNÓSTICO DE FALHAS EM SISTEMAS DE PROCESSOS QUÍMICOS. IMPORTÂNCIA DO CONHECIMENTO DE ESTADOS INTERMEDIÁRIOS DE PROCESSOS DINÂMICOS. DESENVOLVIMENTO DE UMA METODOLOGIA BASEADA EM REDES NEURAIS.

Autor: ANTONIO CÉSAR TEIXEIRA

Orientador.: JOÃO A. F. DA ROCHA PEREIRA

Tese de Doutorado apresentada à Faculdade de Engenharia Química como parte dos requisitos exigidos para a obtenção do título de Doutor em Engenharia Química.

Campinas - São Paulo
Agosto de 2000

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

T235d

Teixeira, Antonio César
Detecção e diagnóstico de falhas em sistemas de processos químicos. Importância do conhecimento de estados intermediários de processos dinâmicos. Desenvolvimento de uma metodologia baseada em redes neurais / Antonio César Teixeira.--Campinas, SP: [s.n.], 2000.

Orientador: João A. F. da Rocha Pereira
Tese (doutorado) - Universidade Estadual de Campinas, Faculdade de Engenharia Química.

1. Controle de processos químicos. 2. Redes neurais (Computação). 3. Sistemas dinâmicos diferenciais. 4. Destilação. 5 Falha de sistema (Engenharia). I. Pereira, João A. F. da Rocha. II. Universidade Estadual de Campinas. Faculdade de Engenharia Química. III. Título.

Tese de Doutorado defendida por ANTONIO CÉSAR TEIXEIRA e aprovada em 25 de Agosto de 2000 pela banca examinadora constituída pelos doutores:

Prof. Dr. João Alexandre F. da Rocha Pereira

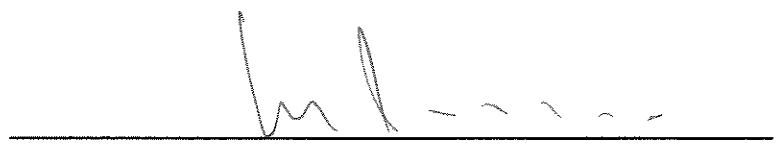
Prof. Dra. Letícia Soares Vasconcelos Sampaio

Profa. Dra. Sandra Lúcia da Cruz

Prof. Dr. Roger Josef Zemp

Prof. Dr. Enrique Ortega Rodrigues

Este exemplar corresponde à redação final da Tese de Doutorado em Engenharia Química defendida por Antonio César Teixeira e aprovada pela Comissão Julgadora em 25 de Agosto de 2000.



Prof. Dr. João Alexandre Ferreira da Rocha Pereira

"Põe quanto tu és no mínimo que fazes."

Fernando Pessoa

À Minha Família

À Cláudia

Ao Professor Pereira

AGRADECIMENTOS

A *Deus*, por estar vivo e poder ter completado este trabalho.

Ao meu orientador, Professor *Pereira*, pelo incentivo, e grande confiança dispensados durante todo o tempo em que trabalhamos juntos.

À FAPESP - Fundação de Amparo à Pesquisa do Estado de São Paulo, pelo apoio fincanceiro para manutenção pessoal e compra de equipamentos que possibilitaram este trabalho de pesquisa.

Aos professores desta Faculdade com quem tirei inúmeras dúvidas durante o desenvolvimento desta tese, em especial aos professores Roger Zemp e Ana Maria Fratini.

Aos meus amigos e colegas de departamento.

Ao pessoal de apoio técnico, Andréia e Joseane.

Resumo

Redes Neurais Artificiais (RNA) têm demonstrado um excelente desempenho em detecção e diagnóstico de falhas em muitas aplicações de engenharia. O grande problema quando se usa redes neurais está no fato de qua as redes somente funcionam bem para dados para os quais foi treinada.

Isto significa que é necessário usar tantos dados quantos possíveis a fim de se cobrir uma larga faixa de condições operacionais do processo. Muitos dos trabalhos publicados em detecção de falhas estãor relacionados ao comportamento no estado estacionário, e o presente trabalho leva em conta a dinâmica do processo na detecção e diagnóstico de falhas.

Um sistema computacional eficiente, baseado em redes neurais com Funções Base Radiais, foi desenvolvido para analisar a influência do comportamento dinâmico de plantas químicas complexas para detectar e diagnosticar falhas.

A retropropagação usando a regra delta generalizada (RDG) para o cálculo do gradiente para o treinamento de RNA's não é conveniente para sistemas com grande quantidade de dados.

Foi proposta também uma metodologia a qual permite agrupar dados para o estudo e análise de sistems complexos, o que permite a detecção e o diagnóstico em grandes sistemas.

Palavras-Chave: Controle de processos químicos; Redes neurais(Computação), Sistemas dinâmicos diferenciais; Destilação; Falha de sistema(engenharia).

Abstract

Artificial Neural Networks (ANN) have demonstrated excellent performance in automatic fault detection and diagnosis in many engineering applications. The great problem when using neural networks, is that neural networks perform only as robustly as the data from which they are trained.

This means that is necessary to use as much data as possible in order to cover the wider range of possible operational conditions of the process. Most of the published work on fault detection are related to steady state behavior, and the present work was undertaken to study the process dynamics effects on fault detection.

A computational system based on neural networks was developed to analyze the influence of the dynamic behavior of a complex chemical plants to detect and to diagnose faults. An efficient artificial neural network which may be trained through large data sets has also been developed.

Although backpropagation using a generalized delta rule (GDR) for gradient calculation has been popularized as a method of training ANN, it is clear that that this methodology is unsuitable for large data systems. For large data systems we found the great efficiency of neural networks using Radial Basis Functions.

Finally, is proposed a methodology which permits to group data from simple lumped systems to study and analyze complex systems which makes possible the detection and diagnosis of the large set of possible of faults.

Keywords: Control of Chemical Processes; Neural Networks (computation); Differential Dinamics Systems; Distillation; Fault of Systems (Engineering).

ÍNDICE

(Página)

CAPÍTULO 1

CAPÍTULO 2

CAPÍTULO 3

CAPÍTULO 7

CAPÍTULO 8

CAPÍTULO 9

APÊNDICES

LISTA DE SÍMBOLOS[4]

a_i –constante empírica para o cálculo da massa específica molar, mol/cm^3, eq.(4.58),

$a_{i,j}$ – parâmetro de interação binária de Wilson, cal/mol, eq.(4.29),

$a_{i,j}$ – parâmetro da equação UNIQUAC, eq.(4.51),

$a_{j,i}$ – parâmetro da equação UNIQUAC, eq.(4.51),

A – parâmetro da equação de Antoine, T em °C e P em mmHg, eq.(4.36)

A_p – área do prato, cm^2, eq.(4.59),

B – parâmetro da constante de Antoine, T em °C e P em mmHg, eq.(4.36),

B – fator de soma das iterações entre as moléculas, equação de TSONOPOULOS, eqs. (4.17), (4.18),

$B_{i,j}$ – fator de iteração entre as moléculas de um gás, eqs.(4.18), (4.49),

b_i – constante empírica para o cálculo da massa específica molar, eq.(4.58),

C – parâmetro da equação de Antoine, T em °C e P em mmHg, eq.(4.36),

Cp_i^0 – capacidade calorífica de um vapor ideal, J/mol.K, eqs.(4.44), (4.45), (4.65),

D – parâmetro da equação de Antoine, T em °C e P em mmHg, eq.(4.36),

D – vazão do destilado, mol/min, fig.(4.1), eqs.(4.5), (4.6), (4.61),

$D_{1,i}$ – parâmetro para cálculo da entalpia do componente 'ideal', eqs.(4.46), (4.47),

$D_{2,i}$ – parâmetro para cálculo da entalpia do componente 'ideal', eqs.(4.46), (4.47),

$D_{3,i}$ – parâmetro para cálculo da entalpia do componente 'ideal', eqs.(4.46), (4.47),

$D_{4,i}$ – parâmetro para cálculo da entalpia do componente 'ideal', eqs.(4.46), (4.47),

ΔH – fator de correção molar da entalpia à temperatura T e pressão P, relativa ao vapor ideal, cal/mol, eqs.(4.42), (4.48),

$\overline{\Delta h_i}$ – entalpia parcial molar do componente i, à temperatura T, pressão P e composição x, relativa ao vapor puro, ideal, e à mesma temperatura, cal/mol, eq.(4.50),

e – parâmetro constante da equação de Francis, e = 0.009345 min$^{2/3}$.cm$^{-1/3}$, eq.(4.59),

---

[4]Os símbolos descritos aqui só servem para o capítulo 4. Para os demais capítulos, os símbolos se referem apenas a parâmetros matemáticos admensionais, definidos no próprio corpo do texto por não terem definições especiais.

e(t) – erro entre o valor medido e o valor do 'set point', eq.(4.76),

E – parâmetro da equação de Antoine, T em °C e P em mmHg, eq.(4.36),

f_i^v – fugacidade do componente i na fase vapor, eqs.(4.15), (4.16),

f_i^l – fugacidade do componente i na fase líquida, eqs. (4.15), (4.19),

f_i^{ol} – fugacidade padrão ou de referência, cal/mol, eqs.(4.19), (4.41),

F_j – alimentação da coluna (fase líquida), mol/min, fig.(4.1), eqs.(4.8), (4.11), (4.13),

h – passo para resolução da equação de Runge-Kutta, eqs.(4.73), (4.74),

h – a soma das entalpias dos líquidos de cada componente como vapor ideal, cal/mol, eqs.(4.50), (4.51),

h_j . – entalpia da fase líquida no estágio j, cal/mol, eqs.(4.11), (4.13), (4.71),

h_{j-1} – entalpia da fase líquida no estágio j-1, cal/mol, eqs.(4.11), (4.13), (4.71),

h_v – altura do vertedouro, cm, eq.(4.59),

h_{NS} – entalpia do líquido no refervedor, eqs.(4.12), (4.14),

hf_j – entalpia de alimentação para o estágio j, cal/mol, eq.(4.71),

H^I – soma das entalpias dos vapores de cada componente como vapor ideal, cal/mol, eqs.(4.42), (4.43), (4.47), (4.63),

H_i^I – entalpia do vapor de cada componente como vapor ideal, cal/mol, eqs.(4.44), (4.44a), (4.46),

H_j – entalpia da fase vapor no estágio j, cal/mol, eqs.(4.11), (4.13), (4.71),

H_{j-1} – entalpia da fase vapor no estágio j-1, cal/mol, eqs.(4.13), (4.71),

i – índice de componente, fig.(4.1),

j – índice de prato, fig.(4.1),

K_c – ganho do controlador, eq.(4.76),

K_i – constante de equilíbrio entre o vapor e o líquido de um componente, eqs.(4.40), (4.41), (4.55),

$K_{i,j}$ – constante de equilíbrio do componente i no estágio j, eqs.(4.39), (4.40),

L_j – vazão do líquido que sai do estágio j, mol/min, eqs.(4.2), (4.8), (4.11), (4.13), 4.59),

L_{j-1} – vazão do líquido que entra no prato j, mol/min, eqs.(4.1), (4.2), (4.8), (4.11), (4.13), (4.71),

L_{NS-1} – vazão do líquido que sai do último prato, mol/min, eqs.(4.3), (4.4), (4.62),

L_1 – vazão de líquido que entra no primeiro prato, eqs.(4.5), (4.6), (4.61), (4.62),

L_w – comprimento do vertedouro, cm, eq.(4.59),

M_j – acúmulo de líquido no estágio j, eqs.(4.7), (4.8), (4.11), (4.13), (4.59),

$M_{i,j}$ – acúmulo do componente i (líquido) no estágio j, eq.(4.39),

M_{NS} – acúmulo de líquido no refervedor, eqs.(4.3), (4.4), (4.9), (4.12), (4.14), (4.57), (4.68),

M_1 – acúmulo de líquido no condensador, eqs.(4.5), (4.6), (4.10), (4.56),

NS – número de estágios,

NC – número de componentes,

p(t) – valor de saída da variável controla, eq.(4.76),

$\bar{p}$ – valor do bias da variável controlada, eq.(4.76),

P – pressão total do sistema, atm, eq.(4.16), (4.17), (4.41), (4.49),

P_i^S – pressão de vapor de um líquido saturado, atm, eq.(4.35), (4.36),

P_j – pressão no estágio j,

P_c – pressão crítica do componente i, atm, eq. (4.37),

Q_R – potência do refervedor, cal/min, eqs.(4.12), (4.14), (4.68), (4.69)

R – vazão de produto de fundo, mol/min, fig.(4.1), eq.(4.3), (4.4), (4.12),

R – constante universal dos gases, eqs.(4.29), (4.33), (4.37), (4.17),

RR – razão de refluxo, eq.(4.61),

T – temperatura, °C, eqs.(4.17), (4.38), (4.46), (4.48), (4.49), (4.58), (4.63), (4.66),

T_0 – temperatura de referência, °C, eqs.(4.44a), (4.46), (4.64),

t – tempo, min,

T_c – temperatura crítica, °C, eq.(4.37), (4.38),

v_i^L – volume parcial molar, mol/cm^3, eqs.(4.35), (4.37), (4.48),

Vb_i – volume molar do compenente i, cm^3/mol, eq.(4.31),

V_C – volume do líquido no condensador, cm^3, eq.(4.56),

V_j – vapor que entra no prato j, fig.(4.1), eqs.(4.1), (4.2), (4.8), (4.13), (4.71),

V_{j-1} – vapor que sai do prato j, fig.(4.1), eqs.(4.1), (4.2), (4.8), (4.11), (4.13),

V_{NS-1} – vapor que sai do refervedor, fig(4.1), eqs.(4.3), (4.4), (4.9), (4.12), (4.14), (4.63), (4.68),

V_R – volume do líquido no refervedor, cm^3, eqs.(4.57), (4.63),

x_i – composição de um componente numa mistura líquida, fig(4.1), eqs.(4.19), (4.26), (4.27), (4.28), (4.32), (4.40), (4.41), (4.50), (4.51), (4.65), (4.67),

$x_{i,j}$ – composição do líquido i que sai do estágio j, fig.(4.1), eqs.(4.1), (4.7), (4.8), (4.55), (4.70), (4.71),

$x_{i,j-1}$ – composição do líquido i que entra no estágio j, fig.(4.1), eqs.(4.1), (4.8), (4.71),

$x_{i,NS-1}$ – composição do líquido que entra no refervedor, fig(4.1), eqs.(4.3), (4.9), (4.69),

$x_{i,1}$ – composição do líquido que sai do condensador,fig(4.1), eqs.(4.5), (4.10), (4.62),

W_j – retirada lateral de líquido da coluna, mol/min, fig.(4.1), eqs.(4.2), (4.8), (4.11), (4.13),

y_i – composição do componente i numa mistura gasosa, eqs.(4.16), (4.41), (4.43), (4.47), (4.49),

y_i – valor anterior da variável dependente na equação de Runge-Kutta, eq.(4.74),

y_{i+1} – valor atual da variável dependente na equação de Runge-Kutta, eq.(4.74)

$y_{i,j}$ – composição do vapor i que entra no estágio j, fig.(4.1), eqs.(4.1), (4.10), (4.39), (4.54), (4.62), (4.71), (4.72),

$y^*_{i,j}$ – composição do vapor caso estivesse em equilíbrio dentro do prato, eq.(4.54),

Y_i – composição do componente i numa mistura gasosa, eq.(4.17),

$y_{i,j-1}$ – composição do vapor i que sai do estágio j, fig.(4.1), eqs.(4.1), (4.8), (4.54), (4.71),

$y_{i,NS-1}$ – composição do vapor que sai do condensador, fig.(4.1), eqs.(4.3), (4.9),

$z_{i,j}$ – composição de alimentação líquida do componente i na alimentação, fig.(4.1), eq.(4.8),

LETRAS GREGAS

$\alpha_{i,j}$ – **parâmetro definido na eq.(4.70),**

γ_i – **coeficiente de atividade do componente i, eqs.(4.19), (4.21), (4.23), (4.28), (4.41),**

δ_m – **parâmetro definido na eq.(4.32),**

ε – **eficiência de Murphree, eqs.(4.53), (4.54),**

ρ_j – **massa específica molar do componente i, no estágio j, mol/cm^3, eq.(4.59),**

ρ – **massa específica molar do líquido no estágio j, mol/cm^3, eq.(4.60),**

ψ_i – **coeficiente de fugacidade do componente i numa mistura gasosa, eqs.(4.16), (4.41),**

CAPÍTULO 1

1.0 INTRODUÇÃO

Um dos objetivos primários da operação em indústrias de processos é melhorar a eficiência de operação dos equipamentos de manufatura.

O termo "falha de processo" em seu sentido mais amplo indica aqueles sintomas que resultam de uma mudança física no processo, tais como desvios no fluxo, no nível do fluido, na temperatura e pressão, comparativamente à sua faixa normal de operação. Falhas também se referem às próprias mudanças físicas ocorridas nas plantas industriais tais como, vazamentos, entupimentos, incrustações, corrosão, desgastes, etc., HIMMELBLAU(1983), WATANABE(1994) e VORA et al. (1997).

O procedimento de detecção de falhas é muito importante dentro da indústria química de uma forma geral. A partir do bom controle de uma planta é possível se ter condições otimizadas, o que pode garantir, dentro de um certo nível de confiabilidade, a segurança de funcionamento da mesma. A detecção e o diagnóstico de falhas em unidades do processo, refletidas nos valores das variáveis do sistema, tem um grande papel no perfeito funcionamento da planta. Devido a fatores como factibilidades técnicas ou econômicas, nem todos os pontos sujeitos a uma falha podem ser monitorados dentro da planta e alguns nem mesmo podem ser visualizados diretamente, como por exemplo desativação de catalisadores. Defeitos podem ocorrer em toda a extensão da mesma trazendo prejuízos que no mínimo contribuem para baixar a eficiência do processo e em casos extremos podem causar a perda de vidas humanas.

Com o aumento da complexidade das unidades industriais, é cada vez mais necessário se ter meios seguros para se detectar e diagnosticar falhas ou indícios que possam levar a situações limites de segurança ou de perdas de matérias-primas ou produtos.

Os procedimentos de detecção e diagnóstico de falhas vêm assim se juntar, aos já existentes, controladores e alarmes presentes em sistemas complexos de processamento químico. Os controladores são imprescindíveis ao funcionamento de uma unidade industrial, e são particularmente úteis em situações onde as perturbações são "passageiras", ou seja, que podem ser corrigidas, sendo o ajuste de certos parâmetros do processo então

necessários, a fim de se trazer o funcionamento da unidade para seu estado original de operação (descontados possíveis *offsets*). Numa situação em que haja um vazamento de algum produto, ou se tenha o travamento de alguma válvula, por exemplo, a perda, daí decorrente, pode ser compensada, algumas vezes, pelos controladores, ajustando-se os demais parâmetros que estejam sendo controlados. Contudo, a ocorrência precisa ser identificada para que possam ser tomadas as medidas necessárias, pois o sistema não terá como, por si próprio, reestabelecer as condições originais de funcionamento. A utilização de alarmes se justifica quando se tem uma variável difícil de se controlar ou que cuja saída do ponto de operação normal pode gerar situações de risco.

Em casos onde os custos de sofisticados elementos de controle não se justificam ou onde a presença dos mesmos é impossibilitada por fatores de ordem técnica, é possível se chegar aos erros e às possíveis causas através de procedimentos computacionais, matemáticos ou não. O quão perto se pode chegar da detecção das falhas e dos diagnósticos corretos dependerá dos caminhos seguidos para se efetuar a tarefa.

Analisando-se a literatura corrente na área de controle de processos, pode-se verificar que existem duas grandes divisões básicas no caso de detecção e diagnóstico de falhas em processos[1]. A primeira grande tendência faz análises qualitativas do funcionamento geral da planta, utilizando basicamente sistemas especialistas[2]. A outra grande tendência é utilizar o conhecimento numérico representado basicamente por balanços de massa e de energia na planta inteira ou em partes da mesma que se quer ter sob monitoramento. Ambas as formas de cálculo podem aparecer isoladas, em procedimentos auto-suficientes, ou seja, nos quais é perfeitamente possível se detectar e/ou diagnosticar falhas em variáveis de processo. Alguns pesquisadores no entanto, VENKATASUBRAMANIAN et al.(1990), por exemplo, argumentam que a utilização de procedimentos híbridos talvez seja a forma mais precisa de se detectar ou diagnosticar falhas.

---

[1] Não se pode dizer que isto seja regra geral pois há autores como KASSIDAS et al.(1998) que fazem a divisão entre dispositivos que usam modelos "mecanicistas" construídos a partir de princípios básicos (conservação de massa, energia, etc.) ou métodos não "mecanicistas" baseados em reconhecimento de padrões. Özyurti et al.(1998) restringem a apenas duas grandes tendências: Sistemas especialistas e redes neurais artificiais.

[2] Ver análise da literatura no capítulo 2.

Dentre as formas que utilizam basicamente conhecimento matemático, está a utilização de redes neurais treinadas para detectar e/ou diagnosticar falhas. O próprio VENKATASUBRAMANIAN e seus colaboradores têm alguns trabalhos publicados com esta característica (1989, 1990) e também HOSKINS E HIMMELBLAU(1988) WATANABE et al.(1989), WATANABE et al.(1994) entre outros. São estudadas nestes trabalhos comportamentos das redes neurais frente a diferentes situações de falhas bem como quantidades e graus de ocorrências das mesmas. Um outro aspecto que é abordado nos trabalhos citados, é sobre a ocorrência de falhas múltiplas, situação razoavelmente comum na prática, segundo os autores. A arquitetura das redes também é objeto de estudo em vários trabalhos.

A vantagem da rede neural em relação aos procedimentos heurísticos, parece estar no tempo de processamento de um e de outro. No caso de utilização de procedimentos matemáticos, os quais não são apenas representados por redes neurais, mas também por sistemas de equações de balanços de massa e energia, resolvidos por variadas ferramentas matemáticas, seguidos de restrições escritas a partir do fluxograma do processo, o tempo de detecção de falhas está ligado ao desempenho do conjunto *software/hardware* utilizados para monitorar o processo. No caso em que o "detector" de falhas escolhido for por balanços, tanto de massa quanto energia, a cada novo estado "fotografado" a partir do funcionamento da planta, é preciso alimentar o programa de controle, de modo automático, *on-line*, ou através de arquivos de dados especiais. A rapidez da resposta neste caso vai depender do nível de automatização do sistema. Redes neurais uma vez treinadas, podem ser mais rápidas no fornecimento de resultados já que a maioria dos cálculos envolvida já foi processada na fase de treinamento ou de determinação dos pesos, que é a fase anterior à utilização propriamente dita, e é a mais demorada.

Um dos grandes problemas do reconhecimento de padrões é a quantidade de dados disponíveis para o treinamento da rede, que muitas vezes pode não ser suficiente, e a construção de um modelo fenomenológico não é uma tarefa fácil, de forma que quase sempre este fator dificulta tomar-se este caminho a fim de se chegar a dados para a utilização de reconhecimento de padrões, ISERMAN(1984) e FRANK(1990).

Em sistemas onde o tempo de detecção e/ou diagnóstico de falhas não for o fator principal, pode-se utilizar procedimentos heurísticos como por exemplo, sistemas especialistas. O tempo de processamento tem de ser avaliado, porque neste caso, uma rotina de interação operador-máquina é executada e o tempo de alimentação das questões feitas pelo programa com as respectivas respostas extraídas do funcionamento do processo dependerá de mais este parâmetro. Para se resolver este problema pode-se utilizar um procedimento que faça a ligação automática entre os dados do processo e o procedimento que contenha o programa analisador especialista.

Em princípio, não é possível se emitir um parecer a favor de um ou de outro procedimento de "cálculo", tanto no aspecto do tempo quanto no de precisão nos resultados, pois para tanto seria necessário que se fizesse um estudo comparativo de ambos utilizando-se a mesma planta ou a mesma parte da planta e isto não foi apresentado na literatura pesquisada por nenhum dos autores. A utilização do procedimento híbrido também foi comentada por VENKATASUBRAMANIAN et al.(1990) assim como a utilização de dados de operação da planta em estado dinâmico e não somente estacionário como se tem apresentado na literatura, segundo as referências levantadas. Este trabalho está referenciado como descrito num artigo de YAMAMOTO e VENKATASUBRAMANIAN(1990), o qual não foi localizado para consulta em nenhuma das publicações pesquisadas. WATANABE et al.(1989) também apresentam um estudo de diagnóstico de falhas utilizando redes neurais, contudo, também utilizam como dados para treinamento da rede, apenas valores relativos ao estado estacionário.

Seria interessante ter uma rede treinada com valores representando fases intermediárias entre dois estados estacionários a fim de se estudar o comportamento da mesma quando solicitada a detectar falhas. Neste caso, ou se obtém dados reais da planta instalada ou, de forma mais prática, através de um simulador dinâmico do processo, ou de parte do mesmo.

Um outro problema que também pode ser apresentado quando se elabora um procedimento de detecção e/ou diagnóstico de falhas, diz respeito à flexibilidade do mesmo, ou seja, até que ponto um dado procedimento, quer matemático quer heurístico, pode ser utilizado quando se tem mudanças na constituição da planta e até que ponto a

confiabilidade das respostas do procedimento continuam válidas. O desejável é se ter um procedimento o mais geral possível e segundo a literatura, os sistemas especialistas são os que mais sofrem com este tipo de restrição. A atualização do algoritmo é muito mais trabalhosa e tediosa no caso de sistemas especialistas que no caso de redes neurais ou mesmo de balanços de massa e energia, mas quanto maior o número de informações que se tiver a respeito do sistema, mais precisa será a detecção do erro ocorrido. Em termos de rede treinada para detecção de erros com origem simples (apenas um foco), é possível se alcançar 100% de precisão no diagnóstico da falha, conforme apresentado por VENKATASUBRAMANIAN et al.(1990), outros trabalhos (descritos na revisão bibliográfica), também apresentam resultados com alta precisão de detecção. À medida que se vai generalizando o procedimento para detecção e diagnósticos de múltiplas falhas com a rede treinada ainda para erros simples, o sistema vai perdendo em precisão, sugerindo que novas formas de treinar a rede devem ser testadas.

1.1 OBJETIVOS E RESUMO DA TESE

O objetivo principal deste trabalho é o de mostrar que o conhecimento dos estados intermediários da resposta dinâmica de um processo sob uma condição de falha é importante e que permite melhorar as condições de utilização das redes neurais contornando deficiências próprias dos algoritmos de treinamento conhecidos. Para tanto, foi preciso construir um programa computacional que fosse o mais eficiente possível no treinamento de redes neurais de forma que permitisse o treinamento de grandes conjuntos de dados conforme os que aparecem em sistemas complexos. Paralelamente à construção do programa computacional pretendia-se desenvolver uma metodologia que conseguisse tratar dados de sistemas complexos a fim de detectar e diagnosticar falhas, sejam falhas simples ou múltiplas, e determinar o nível de uma falha qualquer, utilizando para tanto, dados dinâmicos do funcionamento do sistema.

A fim de se chegar aos objetivos desejados, foram desenvolvidos dois programas principais sendo um de simulação de uma coluna de destilação multicomponente e outro para o treinamento de redes neurais. Este último está dividido em várias partes, sendo possível treinar redes usando o algoritmo em *backpropagation* simples; usando o *backpropagation* modificado pelos algoritmos de Marquardt-Levenberg e também por um algoritmo quase-Newton. O algoritmo mais eficiente é o que usa funções base radiais. O detalhamento de cada um destes algoritmos está no corpo deste trabalho, o qual contém capítulos referentes à teoria de redes neurais e aos aspectos relacionados à detecção e ao diagnóstico de falhas em processos químicos. O capítulo dois faz um apanhado geral da literatura citada apresentando um resumo de trabalhos publicados e apresenta uma justificativa para a escolha da utilização de dados do estado dinâmico para o treinamento das redes neurais para o procedimento de detecção e diagnóstico de falhas. No capítulo 3 está detalhado o que é o procedimento de detecção e diagnóstico de falhas e os modelos existentes para sua execução, mostrando a importância dos mecanismos de detecção e diagnóstico de falhas em sistemas de processos. Detalhes da tarefa do tratamento de dados e os tipos de falhas que podem ocorrer durante o funcionamento do processo também são apresentados. Neste capítulo, são mostrados problemas para se diagnosticar uma falha e os problemas que podem estar relacionados aos instrumentos de medida. São detalhadas as formas de diagnóstico e apresentados os modelos para algumas destas formas. Finalmente, uma abordagem das características desejáveis num mecanismo de detecção e diagnóstico de falhas e uma comparação empírica das diversas abordagens. O capítulo 4 descreve um simulador dinâmico de uma coluna de destilação multicomponente utilizada para gerar dados para o treinamento das redes neurais. O método de cálculo é rigoroso e foram utilizadas duas formas de resolução das equações diferenciais. Primeiro resolveu-se utilizando o DASSL que é uma rotina de cálculo baseada no método de Gear e também uma rotina empregando o método de Runge-Kutta de quarta ordem. Foi levada em conta também a hidrodinâmica da coluna. Devido às características do sistema utilizado, metanol e água, a fase vapor foi considerada ideal. O controlador utilizado na coluna foi um clássico PID digital. São apresentados gráficos para acompanhamento dos resultados de simulação. O capítulo 5 é destinado à análise e descrição das redes neurais e seus aspectos constitutivos. Aqui procurou-se mostrar os diversos aspectos das redes neurais discorrendo-

se através dos melhoramentos surgidos e das implicações no desempenho das mesmas. No capítulo 6 é detalhado o modelo dos algoritmos utilizados na confecção do programa computacional para o treinamento das redes neurais. São apresentadas as equações bem como as modificações nos algoritmos básicos que foram implementados. Finalmente no capítulo 7, estão apresentados os resultados referentes à detecção e ao diagnóstico de falhas que foram introduzidas na simulação da coluna de destilação. Cada um dos exemplos estudados traz um conjunto completo de resultados mostrando a viabilidade de utilização de dados no estado dinâmico do processo para se detectar/diagnosticar falhas em processos de forma eficiente. A metodologia desenvolvida bem como as formas de tratamento dos dados está também descrita neste capítulo.

CAPÍTULO 2

2.0 ANÁLISE DA LITERATURA

Foram encontrados na literatura pesquisada, inúmeros trabalhos que descrevem a detecção de falhas em processos. Nem todo o material existente referenciado pode ser aqui descrito. Contudo, fez-se um apanhado geral dos métodos existentes e dos vários tipos de processos estudados e o resultado desta pesquisa está descrita. Indicações de alguns trabalhos não resumidos, também são descritos apenas como fonte de referências complementares ao trabalho aqui proposto.

2.1 ASPECTOS FUNDAMENTAIS

A apresentação dos trabalhos segue a ordem cronológica e dentro de cada ano, estão agrupados por ordem alfabética.

WATANABE e HIMMELBLAU (1984) descreveram como detectar e diagnosticar falhas para processos estocásticos. A proposta básica para detecção/diagnóstico de falhas era construir uma estratégia em dois estágios na qual o primeiro estágio envolvesse a estimação do estado do processo e o segundo identificação dos valores dos parâmetros do processo. Comparando os valores identificados aos valores em condições normais de operação, pôde-se chegar às causas das falhas. Utilizaram uma estratégia baseada no filtro de Kalman estendido, tentando eliminar as deficiências associadas ao método, como por exemplo os "bias". Na ilustração utilizada para demonstrar a estratégia desenvolvida, o método se mostrou como um ferramenta útil a ser aplicada ao problema real de detecção/diagnóstico de falhas.

VENKATASUBRAMANIAN(1987) descreveram o protótipo do MODEX (Model-Oriented Diagnostic EXpert), o qual é um pacote para localização de causa de

anormalidades em processos químicos. É baseado em raciocínio qualitativo, sendo capaz de diagnosticar diversas falhas. A estratégia de inferência usa o raciocínio baseado no modelo. Os autores afirmaram que o sistema especialista é um perfeito diagnosticador, examinado todas as potenciais origens a partir das quais as falhas podem aparecer. Fizeram um apanhado geral dos processos anteriores a este, de detecção de falhas analisando a utilização de árvores de falha. Mostraram o papel dos dígrafos representados por arcos e nós na propagação das falhas através do processo. Compararam os sistemas especialistas até então utilizados apontando suas deficiências, como por exemplo a falta da desejável generalidade, para que possa ser utilizado em qualquer sistema e não em sistemas específicos, comportando assim mudanças quando o processo também mudar. Falta de diversidade nos possíveis indícios de falhas, não abrangendo apenas indícios com origem única. "Transparência de raciocínio", confiabilidade. Quando frente a falhas não previstas, o sistema deveria ser capaz de identificá-la corretamente. (Para sistemas com o conhecimento baseado apenas na experiência, esta parte só pode resultar em falha). Mostraram as formas de construção do conhecimento necessário à resolução dos problemas indicando a forma como construíram para o caso do MODEX. Neste procedimento, foram utilizadas estruturas e regras produzidas. As estruturas quantitativas foram definidas em termos de apenas três situações, alta, baixa e normal, sendo que as variáveis fora do estado normal estão em falha. Apresentaram dois exemplos demonstrando o funcionamento do MODEX chegando à conclusão de que o protótipo apresentado era viável e útil na ajuda a operadores. Afirmaram que o conhecimento baseado em raciocínio qualitativo baseado em modelo captou as conexões físicas entre as unidades e as variáveis do processo. Disseram que por causa da generalidade do conhecimento base, diferentes porções do processo puderam ser analisadas sem modificar as regras produzidas. A generalidade do processo possibilitou ao sistema diagnosticar diferentes configurações das mesmas unidades do processo. O sistema mostrou grande flexibilidade no sentido de que pode buscar as causas de falhas não descritas e também manusear falhas com múltiplas origens, conseguindo assim a tão desejada diversidade.

O projeto FALCON, VENKATASUBRMANIAN e DHURJATI (1987), talvez seja o mais completo analisador de processo que tenha sido desenvolvido.

Trata-se de um instrumento real que utiliza dois tipos de procedimentos na sua metodologia de detecção/diagnóstico de falhas em equipamentos e instrumentos de medida. O FALCON surgiu de uma necessidade real de controlar processos e foi desenvolvido pelo esforço conjunto de pesquisadores ligados à universidade de Delaware e à Du Pont entre outros. Tem como diretrizes as seguintes características:

1. O processo alvo é a planta de ácido adípico da Du Pont,
2. Possibilidade de detecção de 39 diferentes falhas a que está sujeito o processo instalado,
3. Adequação aos instrumentos de medida existentes,
4. Desenvolvimento de um simulador dinâmico para verificação de dados da planta,
5. Introdução de falhas nas variáveis simuladas para se conseguir estudar o processo.

O FALCON utiliza princípios fundamentais de engenharia representados por modelos quantitativos e também por heurísticas qualitativas as quais são tão poderosas quanto as ferramentas matemáticas. Utiliza 30 variáveis diferentes da planta, amostradas a cada 15 segundos. Estas variáveis incluem os *set-points* e saídas dos controladores associados a um número de outros parâmetros como por exemplo temperaturas, pressões e medidas de fluxos. O modelo final no qual foi baseado, inclui 250 equações diferenciais simultâneas (não-lineares) e mais de 1000 parâmetros. A resolução deste problema consumiu 12 anos-homem. A simulação dinâmica é necessária para testar o conhecimento-base pois é impossível testá-lo apenas com dados da planta. A filosofia inicial utilizada foi a de usar representações matemáticas quantitativas como um primeiro passo e então aumentar o conhecimento com informações qualitativas ou heurísticas. Todas as variáveis disponíveis no controle foram graficadas em função do tempo.

A parte de integração do projeto foi escrita em FORTRAN e a parte das heurísticas foi definido em linguagem LISP. O instrumento de inferência processa os dados alimentados em forma de vetor e quando uma falha ocorre, informa a falha e o vetor de dados corrente. Quando a planta muda de estado padrão ou quando sofre uma parada, uma mensagem identificando a mudança é emitida.

Como conclusão do trabalho, tem-se que heurísticas têm menos domínio de aplicabilidade do que regras quantitativas e isto é uma razão primária para se evitar heurísticas onde seja possível se colocar metodologia de engenharia quantitativa que é mais racional e menos subjetiva. Ainda referente ao modelo utilizado, pode-se dizer que é processo-específico e não comporta mudanças no processo sem um extenso trabalho de atualização de heurísticas e de equações de balanço. Não são fornecidos resultados numéricos para análise de comportamento do analisador de processos.

HOSKINS e HIMMELBLAU(1988) descreveram que o estudo de problemas de processos químicos através de sistemas especialistas requereram a utilização de processamentos simbólicos através de regras explícitas. O modelo proposto, utilizando redes neurais, não contém qualquer representação de regras, explicitamente, apesar de ser baseado numa seqüência de regras dos sistemas especialistas baseados em conhecimentos. O método imita a arquitetura do cérebro humano. As vantagens iniciais destacadas pelos autores ficaram por conta do alto grau de robustez do sistema, ou seja, o sistema não entra em pane devido a presença de falhas no sistema, e a alta habilidade do sistema em "aprender" novas situações melhorando o desempenho. Este artigo ilustra a utilização de uma estrutura de rede neural na representação do conhecimento a respeito do processo. A topologia utilizada foi a de multi-camadas com alimentação antecipada. As redes neurais artificiais aprendem modelos de ativação. A heurística de aprendizado aqui utilizada foi a do GDR. A arquitetura utilizada foi a ZNL (*zone node link*), a qual possui um ambiente de projeto e simulação, implementada em linguagem C. A utilização deste ambiente permitiu centrar esforços apenas na solução dos problemas sem precisar estar implementando mudanças em detalhes de construção da rede. O artigo traz como exemplo de aplicação da técnica, um conjunto composto de três reatores-tanques em série. As variáveis utilizadas para controle de detecção de falhas são, a taxa de fluxo, a temperatura e a concentração dos dois componentes. As leituras são feitas em estado-estacionário. A arquitetura da rede é composta de 6 entradas correspondentes às variáveis de estado do sistema, três camadas ocultas e 6 nós de saída, correspondendo às falhas listadas. Foram utilizados 12 conjuntos de medidas para treinar a rede, sendo 11 conjuntos com algum tipo de falha e um referente ao estado normal de operação (sem falhas). Estes estados foram obtidos utilizando CMS

(*Combined Model Simulator*), um programa de computador projetado para simulação digital de uma combinação de reatores e/ou vasos.

RAMESH et al.(1988) mostraram que a característica comum entre os vários tipos de procedimentos apresentados por trabalhos anteriores é o fato de utilizarem arquiteturas as quais envolvem separação do conhecimento base e do mecanismo de inferência. A estruturação eficiente do sistema especialista para diagnóstico é muito importante para plantas complexas. Focalizar e controlar o problema é importante para se ter um guia para o diagnóstico. A utilização de indícios é muito importante para se conseguir esta diminuição no universo de falhas a ser pesquisado. O problema é que os indícios nem sempre se apresentam "perto" da disfunção ou então o número de indícios simultâneos é muito grande. Focalizar demais o indício pode resultar em desvantagem, em casos como este. O caminho para contornar tais problemas é passar de um sistema grande de disfunções, para um subsistema menor. Esta estratégia resulta na organização de conjuntos de disfunções em termos de conceitos simples. Assim, diagnósticos em tempo real requerem a consideração de somente uma parte do espaço de soluções. A estratégia discutida neste trabalho é geral e pode ser aplicada aos diversos tipos de processos químicos. Se a estrutura é feita explicitamente, é possível desenvolver um ambiente de programação o qual capta a estrutura. O resultado é um sistema especialista no qual as técnicas de IA (inteligência artificial) podem ser feitas de modo mais claro para o usuário. O mapeamento da planta é importante desde os pontos iniciais pois é útil para descrever situações que podem se indícios de anormalidades. Para o diagnóstico em uma planta, onde diversas informações de indícios estão disponíveis, os autores consideram como espinha dorsal do diagnóstico, o raciocínio baseado nas informações dos sensores da planta. Os dados de indícios são dispostos de maneira hierárquica partindo sempre do mais geral para o mais específico. Os nós mais altos na estrutura são associados à hipótese de falhas gerais e como resultado têm mais conhecimento geral associado, enquanto os nós mais baixos têm conhecimentos mais específicos.

VENKATASUBRAMANIAN e RICH (1988) devido à falta de generalidade do raciocínio baseado no modelo, fizeram neste trabalho uma tentativa de melhora e aperfeiçoamento do

método, integrando o conhecimento compilado ao conhecimento em nível rigoroso. Além de mais preciso, o novo método é mais eficiente. Esta integração é formalizada em uma estrutura na forma de arquitetura de duas camadas de um objeto orientado. O trabalho introduziu diversos temas na área de diagnóstico de processos baseado em conhecimento. Foi também apresentado um procedimento para gerar automaticamente as hipóteses de mau funcionamento da planta raciocinando através da estrutura do processo. O procedimento por ser uma melhora de outro (MODEX) já existente, foi chamado de MODEX2 o qual usa a combinação de decomposição estrutural e funcional. O uso explícito da estrutura e do comportamento do processo leva à generalidade do conhecimento base, possibilitando diagnosticar diferentes configurações das mesmas unidades de processo. A metodologia mostrou flexibilidade visto que pode descobrir causas de falhas não relatadas e ainda pode manipular falhas que apresentam origens múltiplas, alcançando com isto a tão desejada flexibilidade do método.

RICH e VENKATASUBRAMANIAN (1989) apontaram que um sistema de diagnóstico baseado em sistemas especialistas poderia ser avaliado de acordo com a velocidade com que alcança suas conclusões e o nível de confiabilidade destas conclusões quando se estudasse um sistema. A confiabilidade de um sistema especialista o qual encontra soluções apenas com a utilização de heurísticas é restrito pois o sistema não tem como verificar as relações de causa intermediárias que foram omitidas pela heurística. Falharam portanto quando foram utilizados para detectar falhas não previstas em princípio. Uma estratégia baseada no aprendizado da máquina, no qual o conhecimento base compilado é desenvolvida incrementalmente gerando novas heurísticas e refinando as antigas, parece ser a alternativa. Novas heurísticas são geradas quando nenhuma está presente no conhecimento base. Os autores propõem portanto, um esquema de aprendizado para aquisição de novas heurísticas e especialização das já existentes, afirmando que a aprendizagem dos sistemas especialistas a partir de experiências passadas é uma característica útil e necessária, pois para se fazer um diagnóstico satisfatório, o procedimento tem que ser rápido e confiável. A necessidade destas duas características simultaneamente pode levar a conflitos potenciais. O procedimento apresentado neste trabalho, visava a corrigir este tipo de limitação dos procedimentos até agora existentes.

VENKATASUBRAMANIAN e CHAN (1989) utilizaram uma metodologia baseada em redes neurais para diagnosticar falhas de um processo de craqueamento catalítico fluidizado. O sistema foi capaz de diagnosticar situações de falhas para as quais não foi treinado. Mostraram as limitações do KBES (sigla em inglês para sistema especialista baseado em conhecimento), que são a trabalhosa natureza da aquisição de conhecimento do processo, incapacidade do sistema de aprender ou melhorar dinamicamente seu desempenho e a imprevisibilidade do sistema de externar seu domínio de especialidade. Este estudo propôs um caminho baseado em redes neurais para contornar os problemas acima expostos na área de diagnóstico de falhas. O estudo comparou o procedimento utilizado com o desenvolvido por BROWNSTON et al.(1985) para o caso do processo de craqueamento e dá uma visão global do CATDEX, um sistema especialista de diagnóstico baseado em regras. Como vantagem do uso da rede neural, os autores citaram a facilidade de se ter resultados após a rede estar treinada e a possibilidade de se ajustar dinamicamente a mudanças no ambiente, a fim de realizar generalizações úteis a partir de exemplos específicos e reconhecer invariâncias a partir de dados complexos de alta dimensão. Redes neurais requerem um modelo de reconhecimento e um modelo de classificação. Utilizaram no algoritmo de aprendizagem, *Backpropagation*, *feedforward* e duas camadas ocultas. O CATDEX tem como desvantagem a impossibilidade de acompanhar possíveis mudanças na planta e incapacidade de diagnóstico em situações novas além de grande dificuldade em ser atualizado. Foram utilizadas redes treinadas com falhas simples na investigação de múltiplas falhas simultâneas. Constatou-se que quanto maior o número de falhas, pior o resultado conseguido para a detecção de falhas.

WATANABE et al.(1989) propõem um sistema baseado em rede neural artificial para diagnosticar falhas utilizando uma rede multi-camada em dois estágios. O primeiro estágio será usado para descobrir as causas das falhas e o segundo o grau da falha, assim podendo identificar de modo incipiente uma possível falha. Os autores mostram que uma falha no estágio inicial de ocorrência pode ser "leve" podendo ser bem compensada por um bom controlador em "*feedback*". O trabalho trata do problema do diagnóstico on-line de falhas

incipientes usando poucas variáveis do processo. São apresentadas comparações entre os métodos de detecção de falhas utilizando sistemas especialistas com críticas às suas limitações. No esquema aqui apresentado, cada nó de saída é associado a uma falha e a condição normal é associada ao último nó de saída livre. O ajuste dos pesos é feito aqui em *backpropagation*. O exemplo utilizado para demonstrar a utilização das redes neurais é o de um conversor catalítico de heptano a tolueno o qual opera em condições próximas ao estado estacionário. O controlador foi assumido não ter falhas, ou seja, ser auto diagnosticável. Foram indicadas 5 possíveis falhas de ocorrer na ilustração apresentada. As variáveis de entrada eram as concentrações de C7H8, a temperatura de saída do aquecedor e o sinal de saída do controlador. As situações de falhas foram simuladas em quatro níveis diferentes (desde o nível 1, ou de incipiência, até o nível 4 ou de alta deterioração) medindo-se as conseqüências nas demais variáveis, para efeitos de treinamento da rede. As falhas foram sempre geradas, fazendo-se a variável em falha ser uma fração do valor verdadeiro. A arquitetura da rede contava com as três variáveis controladas, à entrada, uma camada oculta com quatro nós, escolhida por ter minimizado a quantidade de iterações necessárias, e a camada de saída com as cinco possíveis falhas. A rede foi treinada para fornecer resultados binários (0's e 1's). São apresentadas tabelas comparativas das porcentagens de acerto no diagnóstico de falhas de acordo com o grau da falha em que a rede foi treinada e os resultados se mostram animadores sendo que em nenhum caso o diagnóstico foi feito com menos de 90% de certeza, ficando a média de certeza acima de 95%. A rede foi treinada ao nível mais baixo de deterioração dos valores e mesmo assim conseguiu discriminar entre as causas das falhas a qualquer nível de deterioração. Quando a rede foi treinada com outros níveis de deterioração dos valores das variáveis, notou-se que para o nível 3, também se tem um ótimo nível de detecção. Para a rede treinada ao nível 4, os diagnósticos das falhas 1 e 2 não foram muito satisfatórios. O objetivo aqui é treinar a rede para determinar-se além da falha ocorrida, o nível da deterioração do valor apresentado, simultaneamente. A rede deve ter assim, os três valores de entrada, correspondentes às três variáveis de entrada e 20 saídas correspondentes ao número de falhas vezes o número de níveis de deterioração. O treinamento de tal rede requer extensivos cálculos. Por causa de problemas relacionados à estocagem de valores, é apresentada uma nova arquitetura de construção de redes que permite treinar a rede em dois estágios. No primeiro se determina uma causa entre as causa

possíveis das falhas e no segundo se estima o nível de deterioração da falha identificada. Depois de se comprovar que a rede treinada ao nível de deterioração1 pode detectar falhas em qualquer nível, foi então utilizada a rede já descrita treinada com variáveis corrompidas ao nível 1, sendo este o primeiro estágio do procedimento completo. A mesma configuração de rede usada no primeiro estágio, foi também utilizada no segundo estágio. Os resultados se mostraram satisfatórios podendo ser distinguidas boas parcelas de detecção dos níveis das respectivas falhas. A superfície de decisão foi treinada com seis dimensões. A vantagem da rede em dois estágios é que, se o conhecimento sobre a causa de uma nova falha estiver disponível, somente a rede no primeiro estágio precisa ser retreinada para conseguir diagnosticar o nível da falha. E quando se conhecer um novo nível para as mesmas falhas, somente o segundo estágio precisa ser retreinado.

VENKATASUBRAMANIAN et al. (1990) depois de trabalhos desenvolvidos na área de detecção de falhas utilizando sistemas especialistas, onde apresentam como limitações inerentes ao procedimento, a trabalhosa natureza de aquisição de conhecimento, a inabilidade do sistema de aprender ou de melhorar dinamicamente seu desempenho e a imprevisibilidade do sistema fora do domínio especializado, apresentaram como solução a estas limitações, a utilização de redes neurais. Apresentaram uma análise detalhada de aprendizagem e de generalização das características da rede neural para a detecção e diagnóstico de falhas em processos em estado estacionário. Este trabalho apresenta a possibilidade de se detectar múltiplas falhas num sistema, utilizando uma generalização da rede treinada com erros nas faixas de 5 e 15%. Medidas com erros nas faixas de 10 e 25% apresentaram resultados satisfatórios. A possibilidade de a rede conseguir detectar falhas nas leituras dos sensores também foi investigada. Aqui foi considerado que um sensor está em falha sempre que lê um valor normal quando o valor está errado. Nem todos os casos de falhas nos sensores foram detectados pela rede treinada e isto indica que alguns sensores são críticos para a perfeita detecção de falhas no processo. Foram também analisados termos constitutivos da rede neural, a saber, a taxa de aprendizagem e o *momentum*, além dos pesos e das camadas ocultas. Uma generalização com uma rede com nove camadas ocultas, treinadas para a detecção de falhas simples também foi escolhida para a generalização de detecção de múltiplas falhas. Na maioria dos casos, as redes funcionaram

a contento. Os autores concluem que a utilização de redes neurais é mais vantajosa que a utilização de sistemas especialistas porque mesmo que se precise retreinar a rede a uma mudança da planta, é ainda mais fácil que revisar toda a base de conhecimento de um sistema especialista.

GRANTHAM e UNGAR (1990) mostraram que os sistemas de diagnósticos baseados em modelos apareceram para cobrir as lacunas deixadas pelos sistemas baseados em heurísticas. Trata-se de um procedimento versátil pois pode ser utilizado em um grande número de sistemas o que necessitaria grandes modificações às regras particulares caso fosse feito por heurísticas. Isto é possível desde que se conheça as unidades de processo e suas interconexões. Um procedimento de primeiros princípios (FP) requer robustez e generalidade para dar ao sistema habilidade para criar seus próprios modelos a partir da descrição das substâncias e dos objetos presentes e das condições de processo prevalecentes. A criação automática de modelos é possível porque o sistema tem um entendimento dos fenômenos físicos e químicos básicos. O sistema baseado em FP é melhor que os sistemas baseados em modelo porque se referem aos princípios físicos e químicos fundamentais e têm a habilidade de modificar automaticamente os modelos de suas unidades, o que permite identificar falhas as quais mudam significativamente o modo de operação da unidade. O preço pago pela maior flexibilidade vem cobrado em termos da eficiência. O FP utiliza matemática qualitativa e física qualitativa as quais são usadas, respectivamente, para resolver modelos matemáticos qualitativos e modelos físicos e químicos regentes do processo. O algoritmo utilizado foi o QPT (teoria dos processos qualitativos) de FORBUS (1984). Os autores mostram que a habilidade do sistema em criar suas próprias unidades de modelo permite identificar todas as considerações nas quais um modelo errado tenha sido construído e determinar quais mudanças para as considerações produzem um modelo que explica o comportamento anormal da unidade. Isto possibilitou ao sistema diagnosticar irregularidades no processo sem a necessidade de modelos explícitos do processo. Foi desenvolvido para a análise de unidades individuais ou pequenas plantas. Para grandes plantas, consideram que o FP torna-se computacionalmente não atrativo. Propuseram para contornar este problema, um procedimento híbrido sendo composto por um sistema baseado em modelo e o FP. A utilização de equações

quantitativas também foi sugerida, a fim de diferenciar entre falhas com o mesmo efeito qualitativo mas com diferentes efeitos quantitativos.

HOSKINS et al. (1991) mostraram que dados conseguidos através de sensores ou de outros meios, na planta, podiam servir, se transformados de modo conveniente, para serem utilizados no processo de tomada de decisões sobre o sistema. Propuseram-se a ilustrar um trabalho de detecção e de diagnóstico de falhas em um grande e complexo processo químico. Mostraram como a rede neural pode exibir o comportamento dos sistemas especialistas baseados em conhecimento sem conter qualquer representação explícita das regras. O sistema avaliado é composto de centenas de variáveis de estado, sendo que nem todas podem ser medidas. Os autores descrevem uma situação de falha e mostram os pontos que devem ser entendidos pelo operador a fim de sanar o problema antes que se degenere em uma situação catastrófica. A detecção prematura da falha pode ser de grande importância a fim de se evitar situações fora de controle, e a localização das falhas é de suma importância a fim de se tomar as providências corretivas e finalmente identificar as causas físicas da falha. A detecção on-line é desejável e quanto mais cedo se identifica, melhor. Mostraram que a utilização de redes neurais é melhor que algumas outras formas de detecção por causa da maior rapidez de processamento. Este trabalho utilizou uma rede com duas camadas ocultas e *backpropagation*. A rede contava com 418 nós de entrada e 40 nós na primeira camada oculta e 20 nós na camada de saída, representando vinte estados possíveis sendo 19 erros e o estado normal a serem detectados. Para simulação do processo, utilizaram um pacote computacional definido em DOIG(1986), chamado *Syschem Plant*. O tempo de convergência em uma estação de trabalho Sun 4/10 levou 24h com uma tolerância de 0,05

KELLER et al. (1994) afirmaram que a forma mais fácil de se medir a consistência de dados é ter vários sensores medindo a mesma variável. Apresentaram um algoritmo para o tratamento de diversos erros em medidas ou para vazamentos em sistemas em estado estacionário. O algoritmo recursivo proposto melhora o teste GLR, ou teste de máxima verossimilhança proposto por NARASINHAM E MAH (1987), para o caso em que diversos erros grosseiros aparecem simultaneamente.

SUNGGU e SHIN(1994) listaram métodos probabilísticos automáticos para o diagnóstico de sistemas de multi-processamento. Os autores mostraram a teoria presente nos métodos de diagnósticos probabilísticos e descreveram vários algoritmos em forma de termos simples, com a ajuda de exemplos. Os métodos de diagnósticos foram confrontados e analisados, mostrando as vantagens comparativas dos vários algoritmos frente a diversos fatores importantes para o diagnóstico.

WATANABE et al.(1994) discutiram uma nova arquitetura de redes neurais chamada HANN, sigla em inglês, para rede neural artificial hierárquica. A característica principal de uma HANN é a sua possibilidade de dividir um grande conjunto de padrões em pequenos conjuntos os quais podem ser mais facilmente classificados. O trabalho consiste do uso das redes neurais para diagnosticar falhas incipientes em processos fora do estado estacionário nos quais mais de uma falha ocorre simultaneamente. Os autores analisaram a vantagem em se usar redes para diagnósticos de processos complexos uma vez que esta pode "mapear" de forma satisfatória sistemas complexos sobre os quais não se conhece o modelo, traçando um paralelo de funcionamento entre esta e outros métodos de detecção/diagnóstico de falhas, apresentando suas vantagens e desvantagens. Falaram da dificuldade de se determinar múltiplas falhas utilizando outras formas de raciocínio como por exemplo o baseado em primeiros princípios por causa da necessidade de se ter modelagens bastante fiéis à realidade do sistema de processos o que requer cálculos extensivos. O modelo não podia conter erros o que levaria a falsas interpretações de falhas. Os autores também avaliaram a dificuldade em se construir um sistema de detecção baseado em sistemas especialistas o que requeria um consumo de tempo bastante considerável e ainda a utilização de sistemas especialistas no sistema. A intenção, aqui, é desenvolver uma rede neural específica que seja eficiente no diagnóstico de múltiplas falhas e que pudesse ser facilmente treinada (com presença de falhas simples). A HANN é formada em dois estágios sendo que cada um contém uma função diferente. Mostraram como a classificação de múltiplas falhas simultâneas poderia ser gerada a partir do treinamento com dados envolvendo falhas simples. Duas restrições foram utilizadas para o treinamento da rede, e

isto pode ser restritivo demais se se considerar que erros em uma planta real não ocorrem do modo em que estão descritos no trabalho. As restrições seguidas para o treinamento da rede são: os parâmetros não podem se desviar muito dos seus valores reais (2%) e todas as falhas devem ocorrer dentro de um mesmo grau de desvio. O procedimento foi utilizado na detecção de falhas em um reator químico com um controlador PI, convencional, considerado sem possibilidades de falhas. Os autores apresentaram um estudo de extrapolação da rede neural treinada com baixos desvios, no caso extremo estudado (20% de desvio máximo), a rede cometeu erros na classificação de falhas em 21% dos casos, o que não pode ser chamado exatamente de uma ótima marca. Em casos onde a rede apresentava falhas dentro do padrão treinado, o índice de acerto no diagnóstico foi de no máximo 93%, para erros simples. Segundo o trabalho, isto ocorre porque uma rede neural nem sempre apresenta resultados de extrapolação satisfatórios, uma vez que sua melhor performance ocorre em interpolações. Além do mais nenhum estudo foi feito a respeito da importância das variáveis em falha não detectadas pelo sistema. Redes com faixas de desvio maiores poderiam ter sido testadas. A HANN é mais eficiente que ANN na detecção de múltiplas falhas porque o número de padrões requeridos para treinamento da rede é muito menor no primeiro caso.

CHING et al.(1995) mostraram um método para reduzir a complexidade do equalizador RBF usando somente centros dentro da vizinhança local dos dados de entrada a serem apresentados à rede. Os autores afirmaram que para um problema de equalização, tal técnica permitiu adaptação prática quando o subconjunto de equalização podia ser menor que o conjunto completo de dados.

TSAI e CHANG (1995) propuseram-se a estudar um método de diagnóstico de falhas em sistemas dinâmicos utilizando um esquema de redes neurais integrado. Os autores disseram que neste tipo de monitoramento de falhas de sistemas, o componente mais crítico do procedimento é o gerador de resíduos, o qual é basicamente uma medida da diferença entre o comportamento observado do sistema e aquele que deveria ser observado em condições normais. A proposição em utilizar um sistema de redes integrado partiu do fato de que em

alguns sistemas é difícil se desenvolver modelos matemáticos reais a partir de princípios de engenharia química. Os resíduos são processados em dois outros componentes do sistema. Primeiro, testes simples de *tresholds* são feitos no detector de falhas para identificar situações anormais, depois a tarefa de diagnosticar a falha é conduzida com a rede em alimentação antecipada com "mapas" dos modelos das origens das falhas residuais. A construção da rede em dois estágios se dá de modo que no primeiro estágio, se estimam mapas dos valores de saída, e na segunda rede, faz-se a representação funcional das relações matemáticas. O sistema utilizado para estudo do modelo proposto é constituído de três tanques-reservatórios de mesmo tamanho e de um tanque-reservatório maior, além de tubos para se fazer a interligação entre eles e válvulas para se controlar as vazões entre os mesmos e finalmente, bombas. Foram estudados cinco sistemas no total. As variáveis controladas são pressão de saída da bomba, e a altura do nível de líquido em dois dos tanques. Os autores não apresentam resultados numéricos suficientes para uma análise mais conclusiva, contudo afirmam que o procedimento é capaz de identificar corretamente a origem de falhas simples no sistema.

AL-GHONEIM e KUMAR(1996) mostraram a vantagem que é, em termos de desempenho de funcionamento, a combinação de múltiplas redes neurais. As redes individuais devem ser treinadas independentemente. Os autores mostraram como três redes podem ser usadas para o treinamento, usando o que eles chamaram de função objetivo de mérito, apresentando um novo método de combinar as redes neurais. Pela proposta, a função objetivo é calculada para cada ANN e ponderada para chegar a uma função objetivo aglomerada.

BLUE e HALL(1996) mostraram que a relativa simplicidade de redes com apenas uma camada intermediária poderia permitir o uso de um número mais amplo de algoritmos que poderia ser usado para redes com múltiplas camadas ocultas. Apresentaram um algoritmo que transforma uma rede complexa em uma rede de camada simples. O algoritmo refez as ligações entre as camadas ocultas. Estas novas unidades tiveram os pesos calculados de tal forma que aproximam as contribuições das ligações suprimidas da rede. Alguns exemplos foram apresentados para validar as proposições.

RAMAMURTI e GOSH(1996) apresentaram um método de adaptação dinâmica da arquitetura de uma rede neural em decorrência de mudanças nas condições do problema analisado. O problema de se ter o mapeamento variando com o tempo foi chamado de problema de aprendizagem seqüencial. Os autores também discutiram o fato de que uma rede neural utilizando os conceitos da RBF podesse produzir respostas idênticas àquelas para as quais foram treinadas mas que isto pode não ser conveniente em situações onde mudanças nos elementos de saída do sistema mudam com o tempo. A arquitetura por eles apresentada foi baseada em estruturas especialistas as quais poderiam deter o problema de crescimento incontrolado das redes, uma vez que o tamanho ótimo das mesma não é conhecido a priori. O modelo empregou uma mistura de redes especialistas as quais são modelos probabilísticos que dividem o espaço de entrada em regiões as quais se sobrepõem.

SARKAR(1996) apresentou um método que usa uma quantidade de parâmetros de treinamento de uma rede neural para definir a capacidade de generalização da mesma. O autor indica como as diversas escolhas para ajuste dos parâmetros podem influenciar no desempenho de uma rede. Mostrou que dado um conjunto de dados para treinamento de uma rede, e parâmetros de arquitetura da rede como por exemplo o número de camadas, o número de neurônios em cada camada, o método de inicialização e o algoritmo de aprendizagem, é possível determinar a contribuição de cada um destes fatores no desempenho da rede para o treinamento do conjunto de dados. Apresentou além disto, um método para verificar a validade dos valores estimados dos parâmetros.

MYLARASWAMY e VENKATASUBRAMANIAN(1997) descreveram a contribuição prestada ao consórcio ASM, o qual foi um empreendimento desenvolvido por dez empresas petrolíferas norte americanas a fim de se estimar o impacto econômico de situações de anormalidade no funcionamento de seus sistemas de processos bem como para desenvolver um sistema de controle de processos avançado. O sistema de controle apresentado é uma estrutura fechada de forma híbrida baseada em múltiplas especialidades. Segundo os autores, esta forma híbrida foi necessária devido ao fato de que métodos de diagnóstico

simples não são suficientes para detecção de problemas em escala industrial. Foi proposto, desta forma, um método no qual diferentes formas de diagnóstico são integradas. O modelo proposto foi implementado em G2 e combina diagnóstico baseado em modelo causal, classificadores estatísticos e reconhecimento de padrões sintáticos. Foi apresentada uma comparação entre algumas formas de detecção de falhas, e, particularmente, para a metodologia que emprega redes neurais, os autores argumentam a impossibilidade de extrapolação dos resultados. O modelo desenvolvido foi testado numa unidade industrial real onde apenas os estados de falha que ocorreram na unidade foram testados.

VORA et al.(1997) apresentaram o resultado do estudo de redes neurais em contra propagação. Segundo os autores, as redes com contra propagação são simples e rápidas para o treinamento e têm ainda a vantagem de a melhor arquitetura poder ser determinada de ante mão. Os autores disseram ainda que este tipo de rede não tem o problema de mínimos locais, podendo desta forma sempre encontrar a resposta correta. Foi demonstrado o funcionamento das redes e apresentado um algoritmo de funcionamento das mesmas. Foram consideradas apenas falhas simples (só uma falha de cada vez), que poderiam ocorrer em um reator CSTR. Apesar de usarem o modelo dinâmico do reator, só foram utilizados para o treinamento da rede, os valores do estado estacionário, fixando magnitudes das falhas bem como o seu tipo. Cada uma das sete diferentes situações de falhas foram simuladas separadamente com desvios de 0,1 a 15% a intervalos de 0,1% a partir do valor de operação normal. Cada falha tem, desta forma, 150 padrões para treinamento das redes. Todos os valores de treinamento foram agrupados numa rede com 1050 valores de entrada. Com este procedimento, os autores conseguiram determinar numa única rede, a falha e a sua magnitude. Por causa da dificuldade de treinamento, os autores escolheram 105 padrões dentre os 1050 para o treinamento da rede e diminuíram de 0,1 para 1% a precisão conseguida para a detecção da falha.Os autores disseram que a CPNN (rede neural probabilística) não classifica os padrões de maneira correta quando se inicializa os pesos com valores randômicos. Foi testada também a robustez de CPNN à presença de ruídos. A cada uma das situações de falha foi adicionada e substituída uma porcentagem de sua própria magnitude. Disseram ainda que a CPNN é mais rápida para o treinamento em

comparação com a *backpropagation* devido à presença de um menor número de passos computacionais.

ZHAO et al.(1997) tentando contornar as deficiências do método de detecção de falhas com redes neurais e com a utilização de sistemas especialistas, propuseram um sistema híbrido que conjuga as vantagens de cada metodologia. Os autores disseram que existem cerca de 10 tipos de sistemas neurais integrados a sistemas especialistas. Falaram ainda da dificuldade em se usar redes neurais em *feedforward* com funções sigmodais e as redes com funções base radiais (*RBF*). Desta forma asseguraram ser melhor utilizar a *wavelet-sigmoid basis neural network* (*WSBN*). A teoria *wavelet* é um importante desenvolvimento na análise de Fourier. A estratégia de diagnóstico de falha atua em dois níveis. No primeiro, a rede detecta as falhas processando os dados do funcionamento dinâmico do sistema, mostrando o grau das falhas. A forma de utilização destes dados não foi explicada no trabalho consultado. No segundo nível, o sistema especialista interpreta os resultados do diagnóstico da *WSBN* e prediz a qualidade dos produtos e o estado do reator, e só então faz propostas para remover as falhas, usando a capacidade de 'raciocínio' contida nas regras de construção. A metodologia foi testada apenas para falhas simples.

AFONSO et al. (1998) propuseram um sistema robusto para detecção de falhas nos principais sensores de um sistema de controle. O trabalho expôs a experiência de implementação de um esquema automático para detecção e identificação de falhas em uma planta de escala industrial controlada por um *multiloop* SISO PI (*Single Input Single Output Proporcional Integral*). Segundo os autores, a meta era avaliar o comportamento do sistema de controle por retroalimentação quando uma falha ocorresse em qualquer um dos sensores de medida das variáveis controladas. Neste trabalho apenas foram consideradas situações de falhas unitárias e não simultâneas. A identificação dos parâmetros do processo foi realizada por meio de uma variante do algoritmo de mínimos quadrados recursivos (RLS). A detecção da falha foi alcançada através de um algoritmo de decisão estatística que analisa o comportamento dos parâmetros do processo recuperados pelo RLS. Simultaneamente, um algoritmo de estimativa do estado do processo baseado no filtro de Kalman estendido

(EKF) possibilitou a identificação da falha através da comparação dos valores estimados das variáveis e os valores das respectivas medidas dos sensores.

BAGAJEWICZ e JIANG(1998) propuseram um método para identificação de erros grosseiros em sistemas de processos dinâmicos. Através de reconciliação linear usando o teste e medida da integral dinâmica. Utilizaram um sistema com múltiplos erros grosseiros, como vazamentos e ruídos nos valores dos *holdups*. A inconveniência do modelo utilizado estava no fato de que não conseguia estimar erros grosseiros para determinadas combinações dos mesmos pois isto levava as matrizes a se tornarem singulares. Isto ocorreu devido ao fato que se dois erros introduzidos no sistema eram equivalentes, ou seja, tinham o mesmo efeito para a reconciliação, existiam combinações lineares entre os erros. Desta forma, a função objetivo tinha o mesmo valor, mesmo que os valores das variáveis reconciliadas tenham valores diferentes.

BURDSALL e GIRAUD-CARRIER(1998) propuseram uma solução baseada no uso de uma estratégia evolucionária para auto-otimização. Diferentemente de NERUDA(1995), onde o algoritmo genético (GA) envolvia parametrizações canônicas funcionalmente equivalentes às redes com RBF, os GA's por eles desenvolvidos envolviam cenários *fuzzy* os quais transformavam as funções base da camada oculta da rede. Os autores disseram que a robustez do método era aumentada eliminando-se a escolha não determinística das posições de partida ou de iniciação dos centróides. O GA utilizado foi real e não binário. Uma função "punitiva" foi usada para evitar o sobre-ajustamento causado pela saturação de super-indivíduos ou genes (aqueles que têm um ajuste que na média é melhor que os demais). O maior problema segundo os autores foi o sobre-ajustamento principalmente em pequenos conjuntos de dados. Comparativamente ao RBF "puro", os autores encontraram melhores resultados.

DUNIA et al.(1998) mostraram que pelo fato de as falhas de processo afetarem as medidas de processo de forma diferente, pode-se caracterizar as falhas monitorando as medidas afetadas durante as operações anormais. Contudo, nem todas as falhas podem ser detectadas

pela violação da faixa de operação normal de sensores individuais. Os autores disseram que em processos químicos, falhas quase sempre violam a correlação de dados, fazendo o SPE (Predictor Quadrático de Erros) um excelente indicador para detecção de falhas. Entretanto, diversas condições devem ser verificadas para se assegurar que uma falha possa ser detectada usando SPE. Seguindo estas restrições, propuseram o uso da variância não reconstruída para determinar o número ótimo de componentes principais e o conjunto de sensores para fazer o monitoramento do processo.

LEGER et. al.(1998) utilizaram cartas de controle estatístico e redes neurais artificiais para fazer a detecção e o diagnóstico de falhas em processos pois segundo eles, em primeiro lugar pode ser difícil desenvolver um modelo de processo preciso o suficiente para ser usado em detecção de falhas. Em segundo lugar o uso de cartas estatísticas para sinais medidos favorece o trabalho do operador que vai lidar com variáveis as quais ele reconhece. Estas cartas acumulam desvios das médias das amostras a partir do alvo real ou do alvo desejado, de tal forma que as acumulações alcançam um valor limite alto ou baixo, quando um sinal fora de controle é dado.

ÖZYURTI et al.(1998), descreveram que o problema de diagnóstico de falhas em indústrias de processos químicos pode ser atacado através de uma grande quantidade de métodos, cada um tendo, obviamente suas vantagens e desvantagens. Mostraram que os caminhos a serem seguidos podem ser divididos em seis categorias as quais não são mutuamente excludentes. Desenvolveram um algoritmo simbólico híbrido para detecção e diagnóstico de falhas, o *SC-net*, o qual é baseado em *connecticionst systems* que combinam as características desejáveis dos caminhos "coneccionistas" com as redes neurais e sistemas especialistas, a saber, representação paralela e distribuída do conhecimento, tolerância de falhas, resistência a ruídos, possibilidade de manipular saídas e entradas contínuas e a capacidade simbólica dos sistemas especialistas. A fim de contornar incertezas em suas respostas, lançam mão da lógica *fuzzy*.

REDDY e MAVROVOUNIOTIS(1998) apresentaram um novo método para reduzir a dimensionalidade dos dados usando redes neurais *Input-Training* (IT-*nets*) como modelos não-lineares entre variáveis observadas e variáveis latentes[3]. Os autores mostraram que com este modelo, somente uma camada oculta é necessária por causa da diminuição da dimensionalidade do conjunto de dados e conseqüentemente do problema. O uso das redes IT é demonstrado para a redução da dimensionalidade dos dados, para o realocamento dos dados que estão faltando, para a detecção de erros grosseiros e para a coaptação de dados. Segundo os autores, uma característica importante do *IT-net* é sua capacidade para treinar com dados incompletos. Uma comparação foi feita entre o PCA (Análise do Componente Principal) e as *IT-nets* para o acompanhamento do monitoramento da planta em tarefas como a detecção de erros grosseiros e detecção de falhas. Segundo os autores, *IT-nets* funcionaram melhor que PCA no monitoramento quando as medidas envolviam correlações lineares. Um exemplo simulado (redução à metade dos dados a uma parábola) foi apresentado demonstrando a diferença fundamental entre projeções de redes auto-associativas e uma *IT-net*: uma *IT-net* é descontínua, enquanto a rede auto-associativa é contínua e leva a um comportamento indesejado. Como trabalho posterior, os autores se propuseram a investigar se a variáveis latentes, obtidas como variáveis ótimas de entrada para o treinamento da *IT-net*, são realmente uma importante fonte de informações sobre o estado interno do processo.

WON e MODARRES(1998) descreveram um método para determinação de falhas parciais em sistemas de processos baseado em probabilidades, onde o método *Bayesiano* é modificado pelo modelo da curva-F, chamando-o de *Improved Bayesian Method* (IB). Segundo os autores, o método apresentado, quando comparado a outras metodologias existentes, é particularmente útil no trato de incertezas associadas a falhas parciais, as quais implicam uma maior dificuldade no diagnóstico de falhas. Segundo os autores, a análise probabilística de múltiplos sintomas quando falhas parciais existem pode ser importante em análise de riscos de plantas de processo.

---

[3] Ver Capítulo 3.

ZHAO et al.(1998) por causa da dificuldade de se representar estados dinâmicos durante o treinamento de redes neurais para a detecção de falhas, propuseram a utilização de transformações integrando *wavelets* com redes neurais. Os autores apresentaram uma lista de trabalhos que fazem este tipo de integração e disseram que devido a sua alta precisão, simplicidade de estrutura e rápida velocidade de treinamento, estão se tornando cada vez mais populares como método de estimação de parâmetros, em lugar das redes neurais tradicionais. Disseram que apesar de poder serem usadas para casos multi-dimensionais, os trabalhos até então apresentados somente aplicam para casos de baixa dimensão, devido à grande dificuldade de serem aplicadas para casos mais complexos. A fim de contornar este problema da dimensionalidade, que impede a utilização das *wavelets* em sistemas múlti dimensionais, propõem um novo algoritmo múlti dimensional que evita a necessidade de se fazer o produto de várias *wavelets* e uma dimensão, diminuindo desta forma a dificuldade de chegar à resposta desejada. Desta maneira, tomando-se a *multidimensional wavelet* como função de ativação para a camada oculta e a função sigmóide como função de ativação para a camada oculta e para a camada de saída, uma nova rede neural é proposta, chamada, *multidimensional non-orthogonal non-product wavelet-sigmoid basis function neural network*(WSBFN).

CHAN e LAU(1999) apresentaram uma melhora de um algoritmo baseado em lógica *fuzzy* o qual enriqueceu a expressividade das regras aumenta o poder de inferenciamento do sistema. Segundo os autores, com a introdução de uma nova constante no algoritmo o mesmo se tornou capaz de avaliar a decisão baseado em um efeito combinado de fatores individuais. Mostraram que a importância dos fatores individuais ou dos pesos atribuídos aos fatores puderam também ser moldados de acordo com preferência particular do usuário. A incorporação deste mecanismo aumentou a certeza de que a decisão foi válida, melhorando o poder representacional e a flexibilidade do sistema especialista.

CHEN et al.(1999a) utilizaram *wavelets* para o problema do tratamento de dados em sistemas estacionários. Coisa que as transformadas de Fourier não são capazes de fazer. Isto envolve representar uma função de tempo em relação a blocos simples de construção,

chamados *wavelets*, cuja estrutura mais completa para a transformada foi desenvolvida por CVETKOVIC e VETTERLI(1995), quando projetaram um filtro de análise de múlti resolução não sub-amostrado para determinar as singularidades do sinal.

CHEN et al.(1999b) utilizaram dados de uma refinaria que utiliza processo catalítico para o desenvolvimento de um sistema de diagnósticos. Uma característica do processo é que ele é muito complexo e possui interações dinâmicas entre os balanços de calor, massa e pressão entre os dois vasos reatores e as condições de fluidização. No topo dos *loops* de controle, sistemas de segurança têm um papel importante de prevenir situações desastrosas. Segundo os autores, a utilização da *ARTnet* foi capaz de reduzir a dimensão do transiente dinâmico dos sinais através da extração de características. Outra vantagem da rede com *wavelet,* ao invés de apenas a rede neural pura, é que aquela evita a influência do ruído, facilita a seleção do *treshold* e ainda melhora a velocidade de computação.

HUSSAIN(1999) procurou mostrar uma revisão de métodos de aplicação das redes neurais em controle de processos químicos desenvolvida nos últimos anos e apresentada em literatura aberta. A pesquisa implementada envolve as três maiores categorias do controle. Controle baseado no modelo inverso, controle preditivo e controle adaptativo. As redes neurais utilizadas são basicamente as redes com RBF. Os resultados da pesquisa apresentada mostraram que as redes neurais são versáteis e que são capazes de serem incorporadas em várias estratégias e métodos de controle não-linear. Mostrou que a redes *feedforward* de multicamada com funções de ativação sigmoidal ou hiperbólica é ainda muito utilizada em muitas destas aplicações, o que segundo o autor, demonstra a suficiência e capacidade desta metodologia em identificar e controlar problemas em sistemas de processos. O autor chamou ainda a atenção para o fato de que muitas das aplicações reportadas pela literatura se refererirem quase sempre a equipamentos de escala laboratorial ou senão simuladores o que abre uma brecha para testes em equipamentos reais em escala industrial a fim de se testar a robustez dos métodos.

WEN e CHANG(1999) estenderam a teoria para resolução do problema de diagnóstico baseado no modelo de inferência abductiva. Mostraram que um novo modelo de inferência

fuzzy foi capaz de manusear estes problemas, juntamente com um novo critério de descrever a plausibilidade relativa de diferentes hipóteses de diagnósticos. Baseado neste critério, o problema foi formulado como um problema de programação inteira. Apresentaram os resultados quando testado para três casos de estudo, mostrando que o modelo é eficiente.

2.2 ASPECTOS COMPLEMENTARES[4]

Outros modelos e resultados publicados na área de detecção e diagnóstico de falhas podem ser encontrados nos seguintes trabalhos (por ordem de ano e dentro do ano por ordem alfabética).

BECRAFT e LEE(1991) utilizaram redes neurais hierarquizadas na detecção de falhas em sistemas complexos.

JOKINEN(1991) fizeram a comparação do desempenho de duas redes neurais diferentes no diagnóstico de falhas, utilizando dados de processo em estado estacionário.

ADAMS e STERLING(1992) utilizaram sistemas especialistas com diversas funções no monitoramento de um evaporador de um processo químico. Fizeram um estudo da integração de sistemas baratos e eficientes de monitoramento.

CILLIERS(1992) utilizou redes neurais na detecção de falhas em processos em estado-estacionário e discutiu a possibilidade de utilização de estimativa de parâmetros.

---

[4]Se referem a aspectos ligados ao tema principal do trabalho mas utilizando outras metodologias, menos próximas do tema principal deste trabalho.

HIMMLELBLAU(1992) utilizou redes neurais para o estudo de detecção de falhas em estado-estacionário para trocadores de calor e comparação dos resultados com métodos estatísticos.

VAIDYANATHAN e VENKATASUBRAMANIAN(1992a/b) utilizaram dois métodos de detecção de falhas utilizando redes neurais. Mostraram um método de discretização das saídas desejadas.

VENKATESWARLU et al.(1992) propuseram um método de detecção de falhas em dois estágios utilizando filtros de Kalman, testando os modelos em um reator CSTR.

WATANABE e HOU(1992) fizeram a detecção de falhas incipientes causadas por degradação gradual em variáveis de processo, utilizando redes neurais.

BECRAFT e LEE(1993) utilizaram um procedimento integrado de redes neurais e de sistemas especialistas para detecção de falhas em sistemas complexos.

FAN et al.(1993) apresentaram um procedimento de detecção de falhas em sistemas operando em estado-estacionário para o caso de um sistema de conversão de heptano a tolueno.

ROELE e WARWICK(1993) utilizaram um esquema simples para controle de refluxo, do *reboiler* e do condensador em um processo de destilação em batelada, utilizando processos paralelos e redes neurais no controle inteligente.

ROJAS-GUZMAN e KRAMER(1993) utilizaram análises baseadas em conhecimentos probabilísticos para comparação de análises baseadas em sistemas especialistas, na detecção/diagnóstico de falhas em sistemas químicos.

SORSA, T. e KOIVO, H. N., (1993) utilizaram de redes neurais no estudo de funcionamento de um reator (CSTR), com o arranjo das diferentes categorias de falhas visualizado pela análise do componente principal.

KAVURI e VENKATASUBRAMANIAN(1993) desenvolveram um classificador em dois estágios o qual usa técnicas *fuzzy* com unidades elipsoidais. Neste algoritmo, primeiro se determina o número de camadas ocultas e também uma primeira estimativa para os pesos das redes. O segundo estágio usa a metodologia do *backpropagation* para ajustar os pesos,

KAVURI e VENKATASUBRAMANIAN(1994) propuseram um método para contornar as limitações acerca do tratamento do sistema, quanto a ser uma caixa-preta, e utilizaram redes neurais.

KENDRA et al.(1994) fizeram a integração de um sistema baseado em conhecimento e um sistema multivariável a fim de desenvolver um controle robusto o qual pudesse monitorar o processo químico.

AFONSO et al.(1995) descreveram a implementação de um esquema automático de detecção e de identificação de falhas baseado em análises estatísticas dos desvios dos parâmetros do processo.

CHANG e CHEN(1995) utilizaram filtro de Kalman estendido na resolução de problemas que apresentam falhas com mais de uma origem.

CHII-SHANG TSAI e CHUEI-TIN CHANG(1995) simularam a operação de um sistema de estocagem em semi-batelada para mostrar o funcionamento de um esquema de redes neurais integrados, para a detecção de falhas em sistemas dinâmicos.

DUNIA et al.(1995) fizeram a identificação de falha no sensor via reconstrução.

OZYURT et al.(1995) desenvolveram um caminho híbrido para diagnóstico de falhas utilizando conexões simbólicas.

SAVKOVIC-STEFANIVIC(1995) apresentaram um modelo de detecção de falhas em plantas químicas, utilizando um simulador qualitativo .

TZAFESTAS e DALIANIS(1995) utilizaram redes neurais e mecanismos especialistas, num sistema híbrido para supervisionar uma planta de celulose. Neste trabalho, mostraram dois modelos diferentes de redes.

WENNERSTN et al.(1995) apresetaram uma visão global das diferentes técnicas de detecção de falhas.

GOMM(1994) mostraram um método para detecção de falhas na presença de comportamentos dinâmicos não-modelados e de perturbações randômicas.

SRINIVASAN et al.(1997) propuseram uma estrutura integrada para verificação da segurança de processos. Nesta estrutura, a análise de riscos foi feita em duas fases. Na primeira, utilizaram o *HAZOPExpert*, o qual é um sistema especialista baseado no modelo, para identificar todos os prováveis riscos no pior caso. A partir destes riscos, cenários ambíguos e de riscos sérios, os quais passam a ser analisados detidamente, são

identificados. Na segunda fase os riscos foram avaliados em detalhes, usando o modelo do processo quantitativo.

2.3 ANÁLISE FINAL E JUSTIFICATIVA DA TESE

Da análise da literatura, pode-se fazer um resumo de alguns métodos utilizados na detecção/diagnóstico de falhas. Duas correntes básicas existentes, conforme apresentado anteriormente, levam em conta a utilização de sistemas especialistas ou a utilização de métodos matemáticos. Ambas as tendências, têm representantes que diferem em detalhes de um modelo para outro, contudo é possível extrair características gerais que identificam um método ou outro. Suas respectivas vantagens e desvantagens podem ser descritas e relacionadas como abaixo.

Dentre os modelos baseados em conhecimentos qualitativos do processo, estão o conhecimento baseado no modelo, a análise de árvore de falhas, o estudo de grafos dirigidos, a análise baseada em primeiros princípios. São modelos qualitativos heurísticos de análise. Têm como características:

* requerem um modelo descritivo rigoroso para cada unidade ou planta investigada;

* requerem a descrição da topologia da planta e uma bibliografia básica de falhas e a maneira que elas modificam o comportamento da unidade;

* os procedimentos são usados para buscar, a partir de parâmetros observados, similaridades no modelo aos quais estejam associados o processo, até que a falha seja identificada;

* permitem grande flexibilidade de análise a grandes sistemas mas pecam pelo nível de complexidade quando se deseja analisar pequenos sistemas;

* a maior limitação está em requerer a construção do modelo do processo antecipando-se as possíveis falhas e efeitos. Em situações que ocorram falhas novas, o sistema não consegue diagnosticar corretamente o acontecido e se o fizer, incorre em erros abruptos;

* dependem demais do conhecimento específico de especialistas no processo;

* não têm, em sua maioria, um procedimento simples de raciocínio, representando o sistema muitas vezes como simples caixas-pretas;

* utilização demais de análises subjetivas,

* dificuldades em identificar falhas que ocorrem devido a várias fontes.

Dentre os modelos baseados em conhecimentos matemáticos, estão aqueles que utilizam redes neurais e aqueles que se utilizam apenas de balanços de massa e energia rigorosos do processo (*Deep Knowledge*). Têm como características:

*requerem a utilização de modelos de equações diferenciais e algébricas que é tanto mais complexo quanto maior for o entrelaçamento das variáveis do processo. No caso de redes neurais, estas equações são necessárias apenas na fase de treinamento da rede;

*requerem a utilização de uma biblioteca com modelos de falhas que possam ocorrer durante a operação do sistema;

*maior possibilidade de generalização quando comparados aos procedimentos qualitativos, podendo diagnosticar situações as quais não haviam sido previstas em princípio;

*rapidez de processamento, no caso das redes neurais, depois que o sistema tenha sido treinado;

*possibilidades de generalização para detecção de múltiplas falhas simultâneas;

*possibilidade de manipulação de extensos sistemas sem que seja necessário conhecimento explícito das relações de muitas variáveis do processo;

* necessidade de construção de modelos do processo por causa da grande falta de valores das variáveis com falha em comparação com variáveis em estado normal para treinamento das redes;

*possibilidade de determinar o grau de extensão da falha, seja baseado em equações, seja baseado em padrões pré-especificados;

*possibilidade de detecção/diagnóstico com precisão absoluta, ou seja, 100% de certeza, (pelo menos para a detecção de falhas para as quais foram treinadas).

Por causa das vantagens apresentadas acima e por julgar que os procedimentos baseados em conhecimento matemático são mais convenientes, em relação aos sistemas especialistas, para se estudar um sistema de processos quando o mesmo não tem existência física, mas apenas teórica, por meio de estudo fenomenológico, escolheu-se fazer o estudo do diagnóstico e detecção de falhas utilizando-se um procedimento baseado em redes neurais. Através da leitura de trabalhos apresentados, de forma resumida, nos dois itens anteriores, nota-se que a quase totalidade dos autores utiliza dados para treinamento das redes apenas no estado estacionário. Os autores que utilizam sistemas dinâmicos para o treinamento das redes não o fizeram de forma contínua, ou seja, não mapearam uma trajetória de resposta dinâmica do sistema às perturbações ocorridas.

Conforme dito no capítulo de introdução deste trabalho, um dos pontos fracos das redes neurais é justamente a extrapolação de reconhecimento de padrões para os quais não tenham sido treinadas. Desta forma, pode-se concluir que esta utilização de padrões "isolados", representados apenas por "fotografias" de estados estacionários, tende a amplificar esta deficiência das redes neurais artificiais, reduzindo a sua aplicabilidade na prática.

Com base nesta característica e percebendo que a simulação da coluna de destilação, depois que se introduzia uma perturbação, a qual pode representar uma situação de falha, produzia uma resposta dinâmica, nos momentos iniciais logo após a ocorrência da falha, suficientemente próxima, numericamente, para diversos níveis de falhas nos valores das variáveis monitoradas, propôs-se utilizar os diversos perfis destas variáveis para treinar as redes neurais. Com isto queria-se mostrar que se se utilizasse os perfis dos vários parâmetros de acompanhamento, a diversos níveis de perturbação, seria possível

diagnosticar corretamente situações de falhas mesmo sem conhecer de antemão exatamente o perfil da variável para um determinado nível de falha. Seria possível, desta forma, contornar a dificuldade de extrapolação[5] das redes neurais artificiais.

É nesta premissa que está a razão deste trabalho. Queria-se mostrar que o conhecimento dos estados intermediários no funcionamento dinâmico de um sistema de processos é útil para se contornar uma deficiência de um dispositivo que, fora este aspecto, mostra-se promissor no estudo do comportamento de sistemas de processos perante situações de falhas.

---

[5] É de entendimento mais ou menos comum que as redes neurais sejam aproximadoras de funções, e como tal deveriam, no "caminho traçado", pelo ajuste, serem capazes de interpolar valores de saída ao serem apresentadas a entradas desconhecidas. Na prática, o funcionamento de uma rede neural não permite concluir isto. Preferimos pensar nas redes neurais como ajustadores pontuais e não contínuos, mas que logicamente quanto mais bem comportada for a distribuição dos padrões mais o ajuste se aproximará de um caminho ou trajetória, podendo-se, no limite, pensar nos pesos das redes como parâmetros funcionais. Ver também o Capítulo 5, sobre a generalização das redes.

CAPÍTULO 3

3.0 DETECÇÃO E DIAGNÓSTICO DE FALHAS[6]

Uma planta de processos moderna é caracteristicamente grande e segue uma estratégia de operacionalidade cada vez mais complexa, com longas cadeias de processos seqüenciais e com elaborados instrumentos de controle. Como conseqüência, o diagnóstico de falhas em processos se torna cada vez mais complexo. Uma falha é um estado anormal de uma máquina ou um sistema; seja de uma parte, de uma linha ou de todo o sistema. A ocorrência de uma falha está associada a um número de fatores, os quais por sua vez estão relacionados a um outro número de sintomas.

Uma série de sintomas não antecipados, causados por falhas de baixa probabilidade, ou a pressão por decisões mais rápidas podem levar operadores a chegar a conclusões errôneas, WON e MODARRES(1998). Tecnologias de controle e monitoramento são aplicados a processos químicos a fim de se conseguir uma operação eficiente e segura. A complexidade das medidas em tempo real requerem a ajuda de computadores para que se possa monitorar o processo e interpretar os dados transferindo de uma forma relevante e confiável ao operador humano, REDDY e MAVROVOUNIOTIS(1998).

Um esquema, com a utilização de computador, responsável pela aquisição e tratamento de dados e pela tomada de decisões pode ser visto na Figura (3.1).

Detecção e diagnóstico de falhas dependem muito da natureza da mesma e tornam-se mais difíceis de serem alcançados quando causados por falhas parciais. Desde que a detecção precisa e a identificação em estágios não avançados é muito importante para se minimizar a perda de produtos ou para prevenir conseqüências mais sérias, é necessário se ter um método automático de detecção de falhas.

Uma grande variedade de caminhos para se descrever e para se alcançar a detecção de falhas existe detalhada na literatura consultada. Entre estes estão a teoria de detecção de sinais, de Green e Swets, a teoria de erros versus a teoria dos modelos de taxas de erros,

---

[6] A principal referência para este capítulo é o trabalho de KAVURI e VENKATASUBRAMANIAN(1993)

descrita no trabalho de NIEMELA e KRENDEL(1975), análise de série de tempo[7], e o reconhecimento de padrões, GREENSTEIN e ROUSE(1982), onde diversas técnicas foram desenvolvidas a fim de se conseguir fazer a detecção destas falhas parciais. Métodos empregando lógica *fuzzy*, método bayesiano, redes neurais, método dos dígrafos, árvores de falhas já foram desenvolvidos por diversos deste autores citados, dentre tantos outros. O maior problema é que, na maioria dos casos, a metodologia só funciona bem se o sintoma da variação parcial da falha não é muito severa. Segundo MORRISON(1994), o principal problema é a falta de detalhamento para sistemas mais complexos e também as considerações que cada método faz a fim de alcançar o melhor desempenho.

O fato de as formas para detecção de falhas serem muito variadas e envolverem inúmeras metodologias, não impede que estas metodologias possam, de uma forma global, ser separadas basicamente em duas formas genéricas de abordagem. Segundo esta primeira classificação, pode-se dizer que o diagnóstico/detecção de falhas pode ser de natureza qualitativa, de natureza quantitativa ou contar com ambas as abordagens simultaneamente.

Este trabalho segue como linha de desenvolvimento a segunda maneira, especificamente utilizando redes neurais, que apesar de não poder ser considerada, exatamente, como um método quantitativo 'puro[8]', por envolver, na verdade, o reconhecimento de padrões, os quais poderiam ser feitos apenas com a utilização de dados do histórico do processo, sem a necessidade de utilização de equações do modelo do processo, será aqui considerado como tal por causa da forma de representação dos padrões que seguiu uma representação numérica baseada em dados de simulação dinâmica de um sistema de processo químico.

Aqui se está preocupado em informar as diversas formas de abordagem do procedimento de diagnóstico de falhas. Um detalhamento mais profundo dos modelos qualitativos não faz parte deste trabalho. Como as redes neurais são o objeto principal de análise aqui utilizados, reservou-se os dois próximos capítulos para um detalhamento, a altura, das necessidades deste trabalho.

---

[7] Onde se usa o histórico do sistema para se prever o seu futuro.

[8] No sentido de utilização de equações diferenciais, representando balanços de massa e energia.

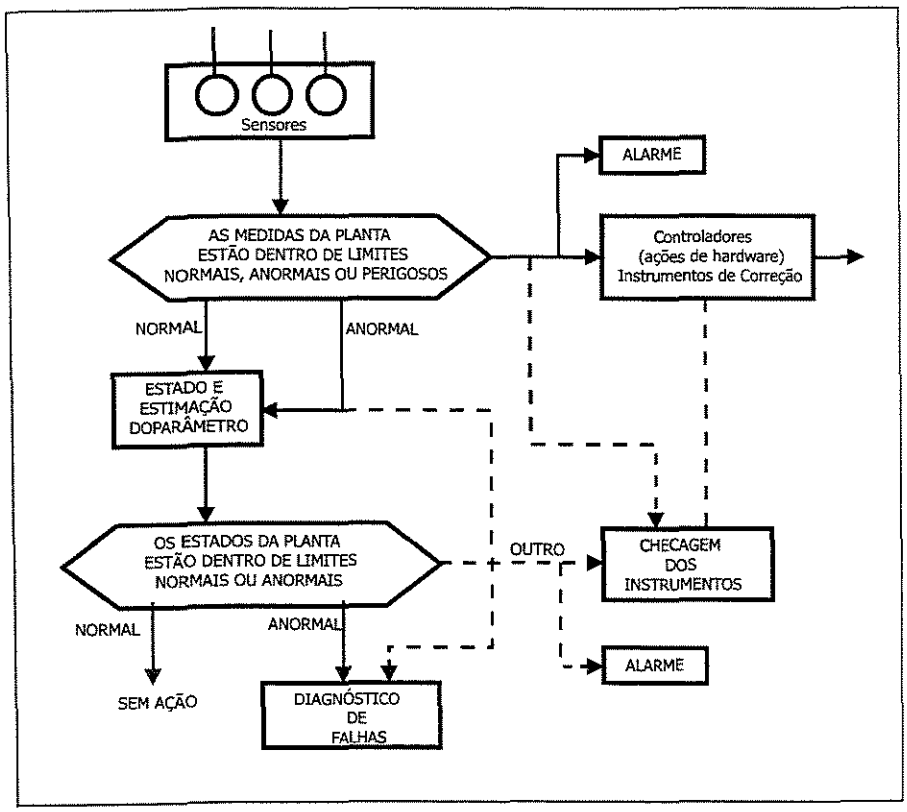


Figura (3.1): Sistema de detecção para diagnóstico automático.

3.1 A NATUREZA DAS TAREFAS DE DETECÇÃO DE FALHAS

Segundo MORRISON(1994), é surpreendente o quão pouco foi escrito a respeito do diagnóstico de falhas em sistemas de diagnósticos avançados dado o tempo que este assunto tem ocupado o trabalho de operadores e especialistas. O autor mostra que o processo de diagnóstico de falhas parece compreender um conjunto básico ou genérico de componentes em uma diversidade de indústrias, e desta forma, a literatura geral a respeito de diagnóstico de falhas, provavelmente, pode ser generalizada para novos ambientes de manufaturas.

O passo inicial no trabalho de diagnóstico de falhas é a coleta de dados do histórico do processo, e especificamente quando se trabalha com métodos de análise que utilizam séries de tempo e reconhecimento de padrões, estes dados têm dupla função; mostrar o comportamento sistemático do processo e servir de fonte de dados para detecção de falhas futuras. Em seguida, deve-se se ater à escolha dos mecanismos de análise, baseados muitas vezes na qualidade dos dados históricos de que se dispõe. O estabelecimento das relações entre fatores e sintomas é o próximo passo de forma a produzir subsídios para a construção da matriz de diagnósticos.

Dentre os estágios comuns a vários trabalhos já publicados, na área de detecção/diagnóstico de falhas, como os de RASMUSSEN e JENSEN(1974), compreendendo indústrias eletrônicas, BAINBRIDGE(1981), para uma grande quantidade de controle de processos, e BEIREITER e MILLER(1988), descrevendo fontes de dificuldades em sistemas de manufaturas em geral, pode-se destacar os quatro seguintes passos:

Detecção das Falhas;

Coleta de Informações;

Eliminação ;

Diagnóstico.

3.1.1 DETECÇÃO DE FALHAS

Apesar de detecção e diagnóstico de falhas estarem fortemente ligadas, muitas vezes sua abordagem é feita de forma separada para fins de detalhamento. Detecção de falhas em um processo industrial é representada por um grande vetor de sensores aos quais estão associados alarmes nas salas de controle. A detecção de falhas pode ser conduzida por

uma combinação de sensores na forma de softwares ou hardwares bem como pela vigilância de operadores a manifestações físicas de falhas.

Logicamente não é possível se listar todos os tipos de falhas e suas probabilidades de ocorrência em uma planta química. Alguns dos possíveis desvios são:

- desvios de pressão,

- desvios de temperatura,

- desvios de fluxos,

- desvios de níveis,

- e assim por diante

Causas de falhas podem ser associadas a distribuição de produtos ou matérias-primas de forma incorreta, mistura incorreta, superaquecimento, ressonâncias, lubrificação incorreta, geração de vórtex, entupimentos, sedimentação, adesão, vazamentos, defeitos de construção, contaminantes, erros humanos, efeitos climáticos, falha de instrumentos e outras. Apesar de cada planta ter condições de operação específicas, o histórico de funcionamento da mesma se apresenta como uma fonte muito valiosa de informações.

3.1.2 COLETA DE INFORMAÇÕES

Informações sobre as falhas do sistema devem ser coletadas antes que qualquer tentativa de se chegar a um diagnóstico possa ser tentada. Normalmente, informações precisam ser selecionadas ou filtradas a partir de um conjunto de dados, onde muitos dentre eles podem ser irrelevantes. Em geral, a primeira informação que o operador do controle recebe pode ser útil apenas para localizar a área geral ou a parte do equipamento no qual a falha está localizada. Em controle de processos de indústrias, tal informação normalmente

é apresentada num painel de instrumentos na sala de controle. Em sistemas menos integrados, tais informações precisam ser inferidas a partir do comportamento real do equipamento.

3.1.3 ELIMINAÇÃO

O próximo estágio do diagnóstico envolve a redução do conjunto de futuras falhas através de uma busca de informações adicionais. Isto pode ser feito de forma vertical ou horizontal. Em uma primeira busca, o operador eliminará ou reterá todos os subsistemas que são candidatos a apresentarem falhas antes de partir para uma análise mais pormenorizada no componente. Por exemplo, isto pode ser feito classificando-se a falha no subsistema como sendo elétrica ou mecânica. Na forma vertical, todos os componentes do subsistema são verificados antes de se mudar de subsistema. Neste nível, são checados todos os sensores, um a um.

3.1.4 DIAGNÓSTICO

O estágio final envolve confirmar a hipótese sobre a qual o componente ou o subsistema realmente falhou. A fim de assegurar que o diagnóstico está correto, é necessário o conhecimento, por parte do operador, do caminho pelo qual deve seguir na retificação da falha.

Se mais que uma causa ocorre, o diagnóstico de falhas se refere à determinação (após a detecção da ocorrência da falha) do equipamento ou porção do mesmo que está causando a falha. Ou seja, diagnóstico é a determinação de quais dos subsistemas, ou do meio ambiente, está violando as condições suficientes para satisfazer as especificações de desempenho do processo, é, portanto, o estudo das relações de falhas, fatores e sintomas, e

é usado para predizer e controlar o desempenho de um sistema, seja ele de que tipo for. Quanto mais complexo o sistema, logicamente mais complexo será isolar a causa de uma determinada falha. Além do mais, um esquema de diagnóstico de falhas deve ser pensado de modo a contemplar a presença de pequenos ruídos nos valores das medidas. Se a informação obtida do sistema é ambígua, outro tipo de informação deve ser procurado de forma a se chegar a um diagnóstico mais conclusivo.

Para se chegar a um diagnóstico correto, é necessário se ter certeza que existe informação lógica e suficiente a fim de reduzir a uma única possibilidade. Se um diagnóstico errado é feito, a correção poderá levar o sistema para condições que possam apresentar até mais perigo que a que realmente esteja acontecendo.

3.1.5 TIPOS DE PROBLEMAS DE DIAGNÓSTICOS

SHEPHERD(1975) dividiu os tipos de falhas em processos em três categorias.

- **falhas esperadas ou familiares,**
- **falhas intermitentes,**
- **falhas desconhecidas.**

Falhas familiares ou esperadas são, logicamente, os tipos mais fáceis de serem identificadas. Em muitos casos, pode-se, de modo direto, escrever um manual ou um algoritmo descrevendo tais problemas. A quebra de uma ferramenta em uma máquina é um exemplo disto.

Com respeito a falhas intermitentes, o problema é um pouco mais grave e a própria quantidade de descrições empíricas de como se comportar numa situação destas é muito menor. Falhas intermitentes são muito difíceis de se detectar.

Segundo BAINBRIDGE(1981), falhas novas são relativamente raras em plantas onde o controle de processos já é feito há bastante tempo. Isto cria o problema de falta de oportunidade de treinamento das habilidades dos operadores, o que paradoxalmente, acaba complicando ainda mais a resolução do problema quando surge uma falha deste tipo e que o operador tem que resolver um problema para o qual ele não foi treinado e possivelmente não tem conhecimento.

O conjunto de requerimentos os quais um bom sistema de diagnóstico deveria ter para ser capaz de funcionar com sucesso na indústria pode ser listado como segue: (a) ser capaz de fazer a detecção de falha no início de ocorrência da mesma; (b) apresentar qualidade na detecção/diagnóstico, não acionando falsos alarmes; (c) resolução para diferenciar entre um tipo e outro de falha; (d) robustez contra ruídos, não tomando como falha uma situação fora do desejado, contudo, prevista; (e) modelagem eficiente a fim de se ter o funcionamento do modelo o mais próximo possível da realidade; (f) facilidade de implementação e manutenção; (g) funcionamento em tempo real, a fim de detectar a falha tão logo a mesma ocorra; (h) capacidade de explicação, de modo a indicar a real causa da falha (diagnóstico); (i) identificações de falhas desconhecidas, mesmo que não seja possível diagnosticá-la, é importante detectá-la; (j) adaptabilidade, de modo a não se precisar mudar todo o sistema a cada mudança de especificações ou mesmo devido ao desgaste natural da planta.

Detecção de falhas, ou monitoramento de processos, é uma forma, conforme já visto antes, de determinar rapidamente se um dado processo está operando de forma aceitável ou não. A detecção rápida é essencial de tal forma que se possa identificar mudanças inesperadas tão logo as mesmas ocorram de modo a se poder tomar as providências o mais cedo possível. Desta forma, o gerenciamento das operações do processo exige que a situação seja corretamente conhecida. Isto requer a habilidade de identificar o estado corrente da planta. Contudo, esta questão pode não ser facilmente respondível porque, segundo KAVURI e VENKATASUBRAMANIAN(1993):

- as medidas podem ser insuficientes

- as medidas podem não ser confiáveis - devido a grandes ruídos ou falhas

mal funcionamento das unidades de processo – por exemplo vazamentos ou entupimentos

- degradação de unidades de processo – por exemplo desativação de catalisadores, incrustações em trocadores de calor

- grandes perturbações externas desconhecidas – por exemplo aumento na temperatura do fluido refrigerante.

Exatidão na detecção é fundamental para se evitar alarmes falsos ou sinalizações incorretas de presença de falhas no sistema. Alarmes falsos são indicativos de mal desempenho num esquema de detecção. Mesmo um pequeno alarme falso durante uma operação normal de um sistema monitorado é inaceitável porque leva a uma perda de confiança no sistema de detecção, podendo levar o operador a ignorar futuros sinais de aparecimento de falhas. Entretanto, mesmo um sistema de detecção que tem uma baixa taxa de alarme falso durante uma operação normal pode registrar um alarme falso quando a planta monitorada vai para um estado não usual, e isto pode ser aceitável em algumas aplicações. O compromisso de projetar um sistema de detecção ponderando fatores como taxa de alarmes falsos, sensibilidade a faltas incipientes, e prontidão de detecção são difíceis de fazer porque requer um conhecimento extensivo, e um entendimento explícito do critério de desempenho do sistema monitorado.

Em geral o que se deseja é uma detecção rápida e precisa, mas nem sempre isto pode ser conseguido visto que são condições quase antagônicas. Uma vez que a falha tenha sido identificada, a raiz do problema deve ser rapidamente diagnosticada. O método mais comum é simplesmente adicionar um alarme e ter-se um operador para determinar o que está errado com a planta e como corrigir o problema.

No contexto geral do diagnóstico de falhas, a definição de falhas inclui:

- Mudanças grosseiras nos parâmetros do processo: falhas nos processos surgem quando há um distúrbio atuando diretamente no processo a partir das vizinhanças através de uma ou mais variáveis exógenas. Um exemplo deste tipo de falha é uma mudança na concentração

do reagente na alimentação do reator. Neste caso, a concentração é uma variável exógena, a variável cuja dinâmica não está relacionada ao processo;

- Mudanças estruturais: mudanças estruturais se referem a mudanças nos caminhos de fluxo da informação. Isto ocorre devido a falhas severas nos equipamentos. Falhas estruturais resultam em uma mudança no fluxo de informações entre várias variáveis.

-Mal funcionamento dos sensores e dos atuadores: Erros grosseiros podem ocorrer com atuadores e sensores também - podendo ser uma falha mecânica, a presença de um bias (positivo ou negativo) ou uma falha de saturação do material. Alguns instrumentos registram os sinais essenciais para o controle da planta e uma falha em um destes instrumentos pode fazer com que as variáveis se desviem além do limite aceitável, a menos que a mesma seja prontamente detectada e corrigida. Esta é uma das razões que obriga a detecção, rapidamente, de qualquer falha nos instrumentos.

Fora do escopo do diagnóstico de falhas estão incertezas não estruturadas tais como ruídos no processo e nas medidas. Estes são tratados como distúrbios desconhecidos e o diagnóstico do sistema deve ser robusto a eles. O problema geral do diagnóstico de falhas é mostrado na Figura (3.2). A figura mostra um sistema de processo controlado e indica as diferentes buscas de falhas. Um sistema de diagnóstico recebe, diretamente, saídas dos sensores, entradas dos atuadores, e às vezes, as saídas dos atuadores. Estas são variáveis observáveis (pela medida) do processo. Falhas e distúrbios não são observados e necessitam ser identificados.

O problema do diagnóstico de falhas, no caso geral, deve ser capaz de identificar falhas em qualquer uma das classes acima, bem como combinações entre elas. O problema se torna assim muito complexo à medida que se aumenta o tamanho do sistema, uma vez que as diferentes classes não podem ser consideradas como sendo isoladas.

Para detecção de falhas, a opção mais importante é o tipo de teste a ser usado para determinar se o processo está operando ou não conforme o esperado. É preciso responder à questão se simples sensores ou múltiplos sensores precisarão ser empregados. Os testes podem ser baseados em dados do histórico do processo ou em predições a partir de simulações do processo. Desde os primeiros trabalhos, o ato de detecção de falhas sempre contou com o auxílio de simuladores. Simulação é conveniente no desenvolvimento de

esquemas específicos de detecção, contudo, não podem prever todos os desafios os quais estão, na prática presentes.

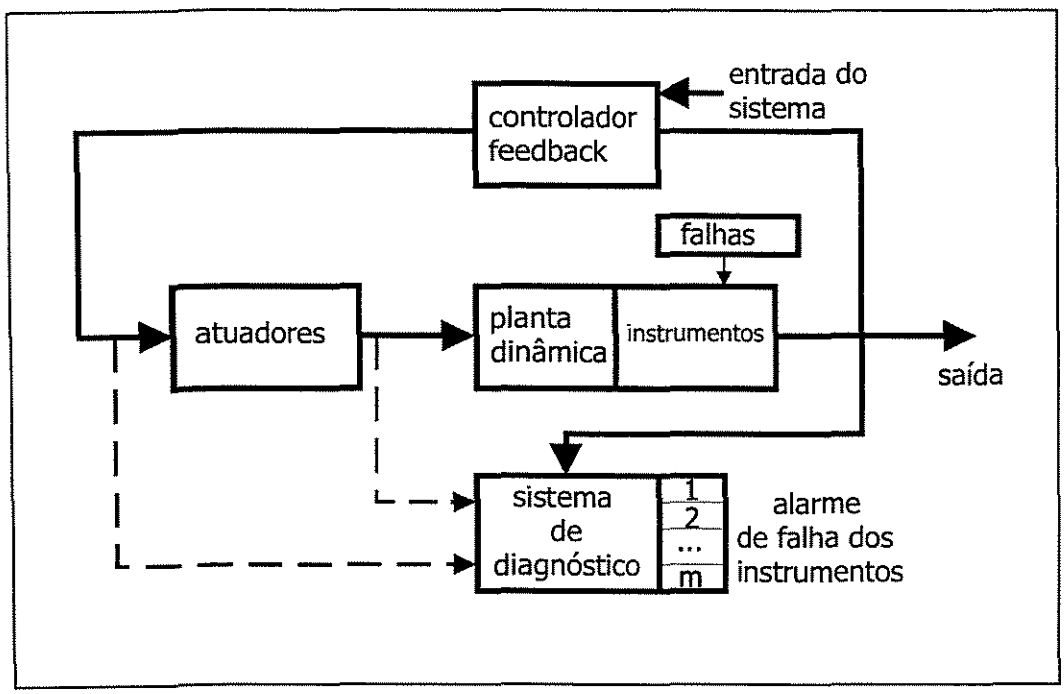


Figura (3.2): Estrutura Geral de Diagnóstico.

O tipo mais simples e mais comum de monitoramento de sistemas usa alarmes baseados em sensores simples, onde fixam-se limites ou *thresholds* de tal forma que o mecanismo emite algum tipo de sinal visual ou sonoro quando a variável monitorada apresenta algum desvio do valor esperado. Deve-se no entanto, se levar em conta ruídos e variações normais do processo, estatisticamente quantificáveis.

Existe uma grande variedade de testes estatísticos os quais usam múltiplos sensores para determinar se uma perturbação ocorreu ou não no sistema. Dentre estes métodos estão o método da Soma Cumulativa dos Erros, Qui-Quadrado e outras extensões dos métodos de sensores simples, conforme ALT(1982) e JACKSON(1985). Há também técnicas de redução de dados, tais como PCA, Análise do Componente Principal e PLS, Projeção de Estruturas Latentes, as quais são combinações de sensores para redução do número de variáveis a serem analisadas. O problema de alguns destes métodos é a grande quantidade de dados em condições de falha que são necessários, a partir do histórico do sistema.

O diagnóstico pode ser conduzido através de sistemas especialistas baseados em conhecimento do sistema ou através da seleção de um modelo que melhor ajuste os dados observados.

Sistemas baseados em conhecimento usam regras heurísticas, "se-então" para encontrar a causa da falha, dadas informações a respeito do estado da planta, seja através de informações de operadores, seja através de dados do próprio processo. Estes sistemas são construídos, juntando-se e codificando-se as experiências de operadores de processo e engenheiros. De uma forma similar, árvores de análise e sistemas de reconhecimento de padrões podem ser usados para encontrar a causa da falha, dadas descrições características no histórico de dados. A desvantagem deste método é a grande especificidade que acompanha cada planta de forma que nem sempre é possível generalizar o conhecimento adquirido para outras plantas ainda que sejam similares quanto às estruturas de processamento.

Como uma alternativa aos sistemas especialistas baseados em conhecimento, os quais examinam regras, sistemas baseados em modelo examinam a planta química para descobrir as mudanças no modelo que produzem um comportamento similar ao observado, chegando desta forma ao diagnóstico. Apesar de modelos quantitativos padrões como as EDO's serem mais fáceis de descrever estes sistemas, eles normalmente requerem detalhes quantitativos que nem sempre estão disponíveis para situações de falha, não possibilitando desta forma as descrições causais requeridas pelo diagnóstico baseado em modelos. RICH e VENKATASUBRAMANIAN(1985) dizem que um sistema especialista ideal deveria usar ambos os procedimentos, baseados em conhecimento e baseado no modelo, a fim de se produzir diagnósticos rápidos e confiáveis em situações comuns e situações onde o diagnóstico não é tão óbvio.

Ao invés de separar os procedimentos de detecção e diagnóstico de falhas, diversos algoritmos quantitativos combinados estão apresentados na literatura. Sistemas de seleção de modelos compara as predições de diversos modelos de sensores de plantas selecionando o modelo que produz o melhor ajuste para a descrição da planta. As técnicas de reconciliação de dados e detecção de erros grosseiros, TEIXEIRA(1997), usam o modelo da planta de modo a estimar os valores 'verdadeiros' a partir de medidas corrompidas com

pequenas flutuações ou ruídos, determinando se existem ou não erros nas medidas. Por natureza, tais sistemas têm múlti-sensores. Diagnóstico de falhas é um procedimento complementar à retificação de dados de processo. O problema destes métodos é o fato de terem sido aplicados somente para pequenas plantas, ou para grandes plantas com apenas um tipo de falhas, LOPARO et al.(1991).

Durante situações como partida, encerramento e outros procedimentos transientes, as técnicas usuais de monitoramento não podem ser usadas, pois de forma similar, as técnicas de seleção do modelo e de reconciliação de dados requerem modelos que são linearizados em torno de um ponto de operação. Sistemas especialistas baseados em conhecimento podem ser usados para diagnosticar falhas em situações transientes desde que tenham sido incluídas regras para operações em condições de operação fora do estado estacionário. Sistemas baseados em modelo também podem diagnosticar durante estados transientes se este comportamento for levado em conta na construção do modelo.

Dois sistemas que comparam simulações de modelos qualitativos a observações da planta real foram recentemente desenvolvidos. O primeiro, **DATMI**, *Dynamic Across-Time Measurement*, tenta explicar as observações selecionando uma descrição qualitativa do comportamento físico a partir de observações qualitativas do sistema, DECOSTE(1990); o segundo, **MIMIC**, diagnostica mudanças a partir do comportamento esperado, comparando com dados de simulação qualitativa, utilizando apenas conhecimento não matemático, DVORAK e KUIPERS(1991).

3.1.6 FALHAS EM INSTRUMENTOS DE MEDIDA

É importante que todos os equipamentos de uma planta química funcionem conforme o projetado a fim de se ter os produtos em nível ótimo, KUMAR1982. Uma vez que plantas químicas são dinâmicas por natureza, são necessários controladores para forçá-la a operar em condições próximas do estado estacionário. Entretanto, mesmo os melhores controladores podem apresentar falhas, e é em vista disto, que a planta, como um todo, deve ser monitorada. Além disto, não existe, na prática, um estado estacionário verdadeiro. As

equações de balanço do sistema nunca podem ser satisfeitas de forma exata quando se utiliza dados nas condições dinâmicas de funcionamento, sempre haverá algum desbalanço, pois podem ocorrer variações de entrada e saída em cada equipamento da planta que não estão exatamente balanceadas com as demais condições de operação; vazamentos ou erros de medição devem também ser levados em conta para se analisar esta questão.

Por mais sofisticado que seja um instrumento de medida, a ele pode sempre se atribuir um erro, a Tabela (3.1) mostra algumas causas de erros de medida.

Há diversos caminhos para o projeto de sistemas de instrumentação de forma a assegurar sua confiabilidade mesmo na presença de falhas. Alguns destes requer a presença de redundância no uso de sensores tendo um dispositivo agindo apenas como um monitor o qual detecta uma falha e faz com que um sinal redundante se desvie dos outros onde não esteja ocorrendo uma falha. Isto é chamado de redundância de *hardware*. Em aplicações onde o custo de *hardware* é muito significativo ou o espaço físico de operação é limitado, procedimentos baseados em redundância funcional (ou redundância analítica) oferecem proteção contra falha em sensores sem a utilização de sensores redundantes.

ERRO	NATUREZA DO ERRO
I	**Erro Sistemático ou fixo**
	A. Erro de Calibração
	B. Erro devido a bias
	C. Erro devido a falha no instrumento
	D. Erro de natureza técnica
	E. Erro devido a vazamentos, deposições,etc
II	**Erros Rândomicos**
	A Erros de julgamento
	B. Erros devido às condições de flutuação
	C. Erros devido à vibração, picos de tensão elétrica, etc.
III	**Erros Ilegítimos**
	A. Erros por falta de habilidade
	B. Erros computacionais
	C. Erros por falta de cuidado ou imperícia

Tabela (3.1): Classificação dos erros de medida.

3.2 TIPOLOGIA DO CONHECIMENTO PRÉVIO

O conhecimento prévio necessário para diagnóstico de falhas é o conjunto de falhas e relações entre estas observações e as falhas. Um sistema de diagnóstico pode tê-lo diretamente, (como uma tabela de busca), ou pode ser inferenciado a partir de alguma fonte de conhecimento. Uma forma de domínio do conhecimento é desenvolvida a partir do entendimento fundamental e é referenciado como conhecimento profundo ou conhecimento causal, segundo MILNE(1987). A outra forma é adquirida a partir da experiência e é referenciada como conhecimento superficial ou conhecimento compilado. Desta perspectiva, pode-se classificar o conhecimento prévio do sistema em duas categorias:

1.conhecimento baseado no modelo do processo

2.conhecimento baseado no histórico do processo

Para o primeiro tipo de conhecimento pode-se considerar

* modelos qualitativos físicos e causais

* abstração hierárquica do conhecimento do processo

* modelos quantitativos

Para o segundo tipo pode-se considerar

desenvolvimento de funções estatísticas representativas

partição do espaço de medidas

tendência das informações

3.2.1.1 MÉTODOS BASEADOS NO MODELO DO PROCESSO

Conhecimento causal qualitativo de processos químicos é largamente usado em muitas aplicações tais como diagnósticos, análises de condições de operacionalidade e análises de segurança. Modelos causais são usados nestas aplicações para mostrar o comportamento dos processos sob condição normal e várias condições anormais. Modelagem qualitativa é atrativa porque pode ser desenvolvida a partir do conhecimento do processo, é auto explicativo, e é um meio adequado e eficiente de representação para muitas situações de diagnóstico. Os sistemas de processos são freqüentemente descritos com a ajuda de modelos quantitativos para os processos, entretanto, estes modelos matemáticos representam os sistemas de processos sem uma referência explícita das relações causa e efeito. O conhecimento causal é desenvolvido, freqüentemente, a partir do entendimento global do processo – a partir de um conhecimento funcional e estrutural possível através de experiência. No entanto, há uma grande quantidade de conhecimento causal oculto nas equações matemáticas constituintes do modelo e que podem ser extraídas para o uso em diagnóstico de falhas, UMEDA et al. (1980).

Diagnóstico é o inverso da simulação. Simulação está preocupada com a derivação do comportamento do processo dados seus aspectos estruturais e funcionais. Diagnóstico, por outro lado, está preocupado com a dedução da estrutura a partir do comportamento. Este tipo de dedução precisa raciocinar sobre as relações de causa e efeito. Relações de causa e efeito são freqüentemente representadas em forma de dígrafos, com arcos diretos levando da causa ao efeito. Cada nó no dígrafo causal corresponde a uma variável. Dígrafos têm nós que representam eventos ou variáveis e bordas que representam relações entre os nós. São muito mais compactos que tabelas. Em uma situação geral, as bordas podem ser evento dependentes, isto é, as relações entre dois eventos ou duas variáveis podem ser dependentes de outros eventos ou de outras variáveis do sistema. Se as bordas são feitas dependentes, é também possível mostrar o comportamento seqüencial do sistema, SHAELWITZ et al.(1977). A noção de causalidade não assume qualquer escolha de espaço-quantidade para descrever o sistema. Na verdade, são fatos separados. A tarefa de

diagnóstico do processo é decomponível em três fatos separados: escolha do modelo causal (esquema para informação do fluxo causa-efeito), escolha do espaço-quantidade para descrever o sistema e a metodologia para diagnóstico.

3.2.1.2 ABSTRAÇÃO HIERÁRQUICA DO CONHECIMENTO DO PROCESSO

A idéia de decomposição é utilizada para que se possa inferenciar o comportamento do sistema global a partir de leis que governem o comportamento dos subsistemas. Na decomposição, o princípio *no-function-in-structure*[9] é central: as leis dos subsistemas podem não representar o funcionamento do sistema como um todo. Em uma descrição hierárquica, pode-se representar uma descrição genérica de uma classe de unidades de processo, onde as equações que governam uma classe de unidades do processo podem ter considerações que sirvam para a classe toda, mas podem ter considerações as quais não representam bem todas as unidades daquela classe. Um outro princípio importante para decomposição de sistemas é o princípio da localidade, pois as leis para uma parte específica podem não se referir a uma outra parte. O princípio *no-function-in-structure* permite considerar o comportamento como consistente entre várias unidades. O princípio da localidade permite que o comportamento seja predito baseado somente em informações locais. Dentre as possíveis formas de decomposições de um sistema de processos estão:

Estrutural: especifica a conectividade de informações de uma unidade,

Funcionalidade: especifica a saída de uma unidade como função de suas entradas (e possivelmente informações de estado).

---

[9] Sem função na estrutura.

A decomposição em unidades de processo permite uma representação geral da funcionalidade de um sistema em termos das relações entrada-saída das unidades. Não é importante para o sistema de diagnóstico ou para o módulo de raciocínio se a funcionalidade seja expressa em termos qualitativos ou quantitativos, tudo depende das relações posteriores de recuperação de resultados ou de diagnósticos.

Técnicas de detecção de falhas podem ser efetivamente usadas para, rapidamente, reduzir o foco de um sistema de diagnóstico usando técnicas de decomposição e abstrações hierárquicas para a representação do conhecimento. A decomposição de um sistema de processos pode ser feita em vários níveis de abstração. Se o nível de abstração é o sistema de controle, então estes subsistemas representam várias malhas de controle individual, SHAFAGHI et al.(1984). No nível das unidades, subsistemas representam unidades individuais.

Conforme já citado, há duas dimensões ao longo das quais abstrações a diferentes níveis é possível – estrutural e funcional, RASMUSSEN(1986). A função abstração hierárquica representa as relações meios-fim entre um sistema e seus subsistemas. Esta hierarquia descreve de alto a baixo quais as várias unidades com certas funções sejam usadas e como elas servem a propósitos mais elevados. Comparando-se a função do subsistema com a função desejada, a hipótese de falha pode ser testada. FINCH e KRAMER(1987) descrevem os subsistemas em um baixo nível, como unidades de processo, sensores, controladores, atuadores e elementos de controle. A idéia é que a falha do subsistema de alto nível é devido a falhas da função de uma ou mais das unidades e usa esta descrição para, rapidamente, identificar o subsistema o qual é fonte de mal funcionamento. A busca funcional é uma direção natural no diagnóstico de sistemas complexos tendo subsistemas com funções específicas que são individualmente reconhecíveis na resposta do sistema global. Este método tem a vantagem de se evitar desnecessários detalhamentos em estágios iniciais do diagnóstico e rapidamente focar nas áreas-problema.

3.2.1.3 MODELOS QUANTITATIVOS

Modelos desenvolvidos a partir dos primeiros princípios, usando balanços conservativos (massa, energia e momentum) e relações constitutivas (tais como equações de estado) são raras vezes usadas no controle e em diagnóstico de falhas na literatura. Modelos de entrada-saída ajustados com dados de processos *on-line* são preferidos, considerando-se que é mais fácil na prática e mais conveniente do que ter uma representação do sistema em forma de algoritmos. Problemas de diagnóstico de falhas surgem por causa da existência de falhas e distúrbios os quais não produzem efeitos observáveis na planta ou nos instrumentos, (sensores e atuadores).

O comportamento do processo é observado a partir das saídas dos sensores e das entradas dos atuadores e quando uma falha ocorre, afeta as relações que existem entre as variáveis observáveis. Matematicamente, GERTLER(1992), mostra que esta relações podem ser representadas da seguinte forma:

Considere que,

$\mathbf{u}(t) = [\mathbf{u}_1(t) \Lambda\ \mathbf{u}_m(t)]^T$ sejam as entradas (observáveis) dos atuadores

$\mathbf{y}(t) = [\mathbf{y}_1(t) \Lambda\ \mathbf{y}_k(t)]^T$ sejam as saídas dos sensores(observáveis)

colocando o modelo em forma de entrada-saída, relacionando $\mathbf{u}(t)$ e $\mathbf{y}(t)$ somente, dado em forma de média móvel:

$$\mathbf{H}(z)\mathbf{y}(t) = \mathbf{G}(z)\mathbf{u}(t) \qquad (3.1)$$

ou

$$\mathbf{y(t)} = \frac{\mathbf{G(z)}}{\mathbf{H(z)}}\mathbf{u(t)}$$

onde $\mathbf{H(z)}$ e $\mathbf{G(z)}$ são matrizes polinomiais em $\mathbf{z^{-1}}$, onde $\mathbf{z^{-1}}$ é o operador reverso[10]. As matrizes $\mathbf{H(z)}$ e $\mathbf{G(z)}$ são da forma:

$$\mathbf{H(z)} = \mathbf{I} + \mathbf{H_1}.\mathbf{z^{-1}} + \mathbf{H_2}.\mathbf{z^{-2}} + \Lambda + \mathbf{H_n}.\mathbf{z^{-n}}$$

$$\mathbf{G(z)} = \mathbf{I} + \mathbf{G_1}.\mathbf{z^{-1}} + \mathbf{G_2}.\mathbf{z^{-2}} + \Lambda + \mathbf{G_n}.\mathbf{z^{-n}}$$

O modelo de processo (3.1) é um modelo ideal quando não existem falhas, ou qualquer outro distúrbio, corrompendo os dados, inclusive ruídos. Quando se desenvolve um modelo com falhas a partir desta representação, é preciso se distinguir entre duas formas diferentes de falhas. As falhas aditivas e as falhas multiplicativas. Ambos os tipos resultam em termos adicionais ao modelo do processo. Falhas multiplicativas resultam em mudanças nos parâmetros, isto é, nas matrizes $\mathbf{H(z)}$ e $\mathbf{G(z)}$, multiplicando variáveis observadas e assim dependendo dos valores reais das variáveis. Falhas aditivas, por outro lado, aparecem como adições ao modelo do processo e são independentes dos valores das variáveis observadas.

Falhas nos sensores e nos atuadores podem ser incluídas na equação (3.1), de forma que:

$$\mathbf{H(z)}(\mathbf{y(t)} + \mathbf{q(t)}) = \mathbf{G(z)}(\mathbf{u(t)} + \mathbf{p(t)}) + \mathbf{K(z)}\mathbf{w(t)}$$

ou

---

[10] Do original, backward-shift operator

$$\mathbf{H(z)(y(t)) = G(z)(u(t)) + H(z)q(t) + G(z)p(t) + K(z)w(t)} \quad (3.2)$$

$\mathbf{q(t)}$ são as falhas dos sensores, $\mathbf{p(t)}$ são as falhas nos atuadores e $\mathbf{w(t)}$ são distúrbios desconhecidos. Distúrbios desconhecidos incluem "incertezas não estruturadas" , tais como medidas com ruídos e falhas desconhecidas. Tudo isto é incorporado ao modelo do processo como falhas aditivas. Falhas nos parâmetros não podem ser modeladas como falhas aditivas quando elas ocorrem nas matrizes $\mathbf{H(z)}$ e $\mathbf{G(z)}$. Devem ser representadas como falhas multiplicativas e são incluídas no processo da seguinte forma:

$$\mathbf{(H(z) + \Delta H(z))y(t) = (G(z) + \Delta G(z))u(t)} \quad (3.3)$$

Comparando-se as equações (3.2) e (3.3), outra distinção pode ser vista entre falhas multiplicativas e aditivas: falhas aditivas ocorrem no modelo como funções desconhecidas do tempo, multiplicando matrizes conhecidas, enquanto as falhas multiplicativas ocorrem no modelo como funções conhecidas do tempo (observáveis) multiplicando matrizes desconhecidas. Estas diferenças tem relação com a forma como são tratados os tipos de falhas no mecanismo de diagnóstico, GERTLER(1992).

3.2.2.1 MÉTODOS BASEADOS NO HISTÓRICO DO PROCESSO

A Figura (3.3) mostra métodos baseados unicamente no histórico do processo. Métodos estatísticos são usados para identificar propriedades compartilhadas por dois sinais diferentes. Através destes procedimentos, usam-se filtros os quais são sensíveis à uma determinada freqüência, separando-oS, de forma a se poder analisar a composição dos sinais. Os métodos não estatísticos são de dois tipos: aqueles que analisam a tendência das características e aqueles os quais simplesmente dividem o espaço de medidas em diferentes

classes de falhas. Métodos de análise de tendência variam de descrições qualitativas, as quais examinam as informações do histórico do processo para certos tipos de eventos primitivos a métodos os quais examinam a freqüência para identificar eventos.

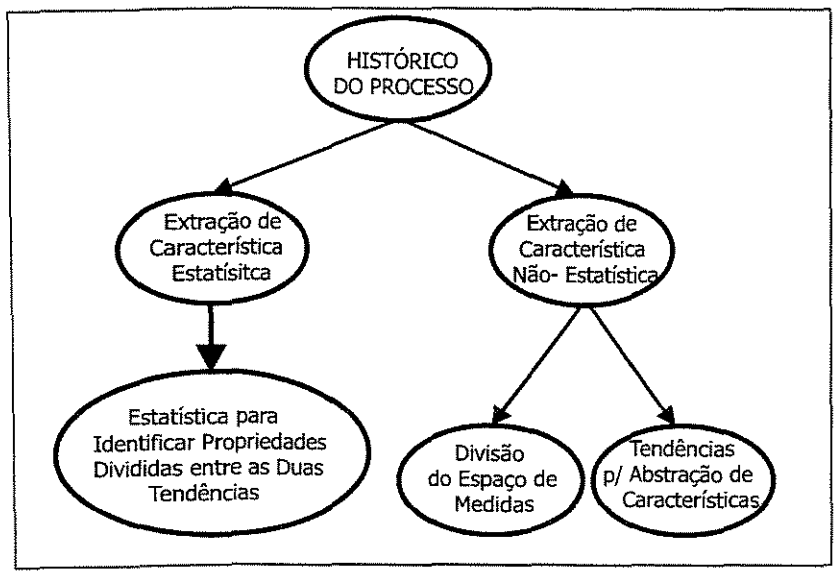


Figura (3.3): Representação do histórico temporal.

3.2.2.2 PROPRIEDADES ESTATÍSTICAS A PARTIR DO HISTÓRICO TEMPORAL

Quatro tipos principais de funções estatísticas são usadas para descrever as propriedades básicas dos dados randômicos: (a) valores quadráticos médios; (b) funções de densidade de amplitude; (c) funções de autocorrelação; (d) funções densidade de potência espectral. Os valores quadrados médios possibilitam uma descrição muito básica dos dados. As funções de densidade de amplitude possibilitam informações ligadas às propriedades dos dados no domínio da amplitude. As funções de autocorrelação e de potência espectral fornecem informações de probabilidade no domínio de tempo e domínio de freqüência, respectivamente. Para dados em estado estacionário, funções de autocorrelação e de

potência espectral possibilitam a mesma informação mas em domínios diferentes – na verdade são pares de transformadas de Fourier.

3.2.3 EXTRAÇÃO DE CARACTERÍSTICAS NÃO ESTATÍSTICAS A PARTIR DO HISTÓRICO DO PROCESSO

Há outros caminhos pelos quais se pode extrair novas características a partir do histórico do processo. Um caminho comum para representação do histórico do processo é dividindo-se o espaço das medidas em regiões de diferentes comportamentos do processo. Pode-se desenvolver detectores de características os quais são usados para identificar de quais partições de um dado processo o comportamento está mais próximo. Dependendo de detalhes da informação usada para estes detectores de características, há informações que vão de qualitativas passando pelas semi-quantitativas a quantitativas. Um exemplo de uma representação qualitativa do histórico do processo são os sistemas baseados em regras desenvolvidos a partir de conhecimento compilado ou experimental. Um exemplo de representação semi-quantitativa é o uso da análise de intervalo e outras análises de ordem de magnitude. Representações quantitativas, por outro lado, tentam capturar a distribuição da informação dos padrões em um espaço de medidas. Este é o caso de um caminho recentemente desenvolvido o qual utiliza redes neurais no diagnóstico de falhas.

Há muitos caminhos para o monitoramento de processos, interpretação de dados e extração de característica. Métodos de extração de característica manipulam dados para produzir variáveis úteis, características, relacionadas ao estado do processo. Métodos de interpretação de dados relacionam estas características ao estado do processo ou outros descriptores os quais são usados pelo operador para a detecção da falha. Deste modo, o monitoramento de processo é uma combinação de passos de extração de características e interpretação de dados.

A extração de característica é uma forma de redução da dimensionalidade do conjunto de dados de um sistema e é a transformação de informações alojadas em dados

medidos em um conjunto manuseável e útil de descriptores do processo. Alguns dos métodos utilizados para redução de dimensionalidade são baseados em **projeção matricial** e são aplicados para dados multivariáveis. Reduzem a dimensão de variáveis projetando valores observados em uma hipersuperfície[11] de tamanho reduzido. A projeção entre os dois espaços é possível através de modelos de mapeamento e desmapeamento que captura a relação entre as variáveis observadas e as variáveis latentes (na dimensão menor). São exemplos o SPC, PCA (componente principal) e regressão de mínimos quadrados parciais (PLS), redes neurais e outros métodos multivariáveis não lineares tais como *Principal Curves/surfaces* e versões não lineares de PCA e PLS.

Um outro caminho para se representar o histórico do processo é a abstração da tendência da informação. Análise de tendência e predição são componentes importantes de monitoramento de processos e supervisionamento de controle. Conhecer-se as tendências, as causas que podem influenciá-las e as tendências futuras é essencial para a decisão de supervisionamento. A partir de uma perspectiva do procedimento, a fim de se obter uma tendência do sinal não suscetíveis a variações momentâneas devido a ruídos, algum tipo de filtragem precisa ser feita.

A abstração qualitativa permite uma representação compacta da tendência pela representação somente dos eventos significativos. Para tarefas como diagnóstico, uma representação de tendência qualitativa freqüentemente disponibiliza valiosas informações que facilita o raciocínio sobre o comportamento do processo.

3.3 PAPEL DO CONHECIMENTO REDUNDANTE EM DIAGNÓSTICO

Todos os métodos de diagnóstico requerem alguma forma de redundância as quais são basicamente divididas em dois grupos: redundância de *hardware* e redundância

[11] Porque os dados não estão distribuídos de forma planar.

funcional ou analítica. Redundância de *hardware* requer o uso de sensores redundantes o que pode se mostrar caro ou não factível[12]. Redundância funcional ou analítica é conseguida pelo uso de estimadores[13].

3.3.1 REDUNDÂNCA ANALÍTICA

Redundância analítica é conseguida através do conjunto de todas as relações temporais existentes entre as entradas dos atuadores e as saídas dos sistemas os quais apresentam diferenças iguais a zero quando não ocorrem mudanças no sistema. Um esquema geral para o uso de redundância analítica em diagnóstico é mostrada na Figura (3.4). Há dois tipos de redundâncias analíticas em diagnósticos, CHOW e WILLSKY(1994), BASSEVILLE(1988) e FRANK(1990):

1.Redundância Direta: são relações algébricas entre diferentes medidas de sensores. Na ausência de uma falha nos sensores que aparece na relação, o resíduo deve ser igual a zero. Esta relação pode ser usada para computar o valor de uma medida de um sensor usando a medida de um outro sensor. Pode-se também usar este valor computado para comparar com o valor medido por aquele sensor. Um desvio nestes valores mostra que um dos dois sensores está com falha.

2. Redundância Temporal: estas são relações diferenciais entre diferentes saídas dos sensores e entradas dos atuadores. Entretanto, redundância direta não pode identificar falhas nos atuadores quando entradas a um tempo k não se relaciona com saída ao mesmo tempo

---

[12] Problemas de localização por exemplo, podem inviabilizar a utilização de um determinado sensor que seria necessário.

[13] Por exemplo filtro de Kalman, ou observadores de Luenberger.

k. Para ver como redundância temporal é usada para detecção de falha em sensor e atuador, considere o sistema SISO da forma:

$$\mathbf{x(k+1) = ax(k) + bu(k)} \tag{3.4}$$

$$\mathbf{y(k) = cx(k)} \tag{3.5}$$

onde **u** é a entrada do atuador. Uma falha no atuador corresponderia a uma mudança no parâmetro **b**. Falha no sensor corresponde a mudanças no parâmetro **c**. Multiplicando-se (3.4) por **c** e substituindo-se na equação (3.5), tem-se:

$$\mathbf{y(k+1) = ay(k) + cbu(k)} \tag{3.6}$$

O resíduo da equação (3.6) é, agora, sensível tanto a falha no sensor quanto no atuador.

Métodos baseados em redundância analítica derivam resíduos os quais são insensíveis às incertezas mas são sensíveis às falhas. Um dos caminhos mais diretos de fazer isto é o método da perturbação desacoplada. Por este método, todas as incertezas são tratadas como distúrbios e filtros (também conhecidos como observador de entrada) são projetados para desacoplar o efeito das falhas e das entradas desconhecidas tal que eles possam ser diferenciados.

3.3.2 REDUNDÂNCIA DE HARDWARE

Técnicas de escolha são freqüentemente usadas em sistemas que possuem alto grau de redundância de hardware, WILLSKY(1976). Por exemplo, considere três sensores

idênticos medindo a mesma variável. Se um dos três sinais difere de forma marcante dos outros dois, o sinal diferente é identificado como em estado de falha. A diferença entre dois sinais em todo par de sensores em um grupo indica uma falha. Esquemas de escolha são fáceis de implementar e são rápidos para identificar falhas mecânicas em instrumentos. Um caminho alternativo é buscar, dados os limites de erros em cada um dos sensores, por subconjuntos de medidas de sensores com diferentes graus de consistência. O grupo mais consistente é usado para se estimar a quantidade medida. O subgrupo menos consistente é usado para isolar os sensores com falhas.

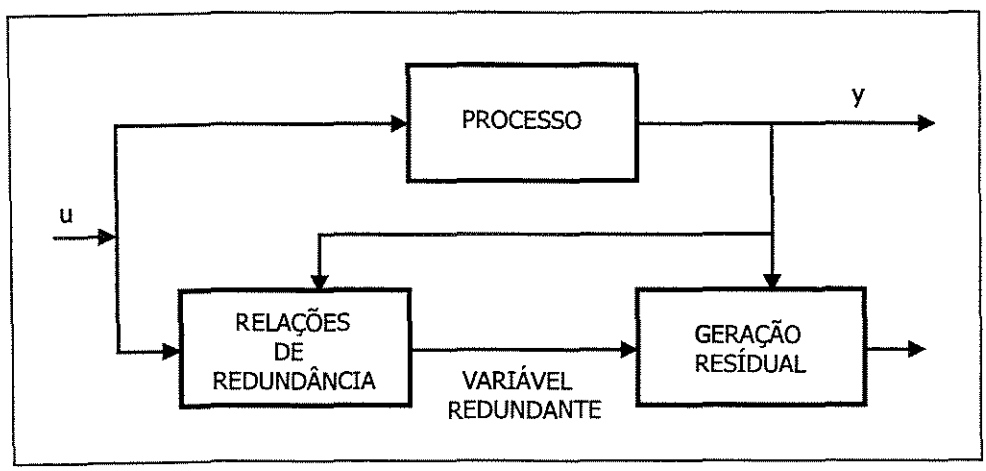


Figura (3.4): Esquema geral para o uso de redundância analítica.

3.4 TIPOS DE ESTRATÉGIAS DE DIAGNÓSTICOS

RASMUSSEN(1986) considera que existem basicamente dois tipos diferentes de estratégias para diagnóstico de falhas, busca tipográfica e busca sintomática. Neste item estão detalhadas estas duas estratégias, nos itens 3.5 e 3.6 e 3.7 outras formas de abordagens são descritas.

3.4.1 BUSCA TOPOGRÁFICA

A busca pode ser efetuada em um sistema que esteja apresentando alguma anormalidade com referência a algum modelo representando a situação normal ou uma operação planejada. A falha será encontrada como uma contraposição ao estado normal e será identificada pela sua localização no sistema. Este tipo de busca é chamado de busca topográfica, conforme ilustração na Figura (3.5). A topografia de busca depende somente do modelo de operação normal da planta. A consistência e a exatidão da estratégia não dependem, no modelo, da extensão do mau funcionamento, e consequentemente é pouco influenciada pelos distúrbios múltiplos e desconhecidos. As estratégias topográficas dependem da busca com referência a um modelo de funcionamento normal de forma que satisfazem bem à identificação de distúrbios que não sejam empiricamente conhecidos ou que não tenham sido previstos. Uma vez que as falhas não sejam conhecidas, em princípio, a busca topológica ajuda somente nas proximidades do foco[14] de diagnóstico de falha para o subsistema de tal forma que este método de detecção de falhas mostra somente a presença ou não da mesma. Entretanto, com a aplicação repetida do procedimento, ao subsistema, a localização pode ser identificada. A importante característica da localização da falha usando este método é que o mesmo não assume que o mau funcionamento tenha uma origem conhecida.

3.4.1.1 INOVAÇÕES DE ABORDAGEM NA DETECÇÃO DE FALHAS

O sinal do erro ou a inovação do processo é definida como a diferença entre a saída real do sistema e a saída esperada baseada no modelo normal. A saída esperada é

---

[14] Esta conseqüência é mais sentida em sistemas de grande escala.

gerada a partir do modelo usando a saída prévia do sistema real. É chamada inovação do processo porque representa a nova informação trazida pela última observação. É usado um filtro de *kalman* para gerar as inovações do processo quando se sabe que a média é zero, ou seja, resíduos independentes. Quando o processo não tem falha, a seqüência de inovações padronizadas satisfaz a hipótese do *white noise*. Quando uma falha ocorre, as inovações padrões saem da média zero e perdem a característica de 'brancura'. Com a ajuda destas estatísticas, o problema de detecção de falha pode ser formulado como um problema de teste de hipóteses considerando o comportamento normal do processo como a hipótese nula. A inovação do processo é testada contra esta hipótese a um desejado nível de significância.

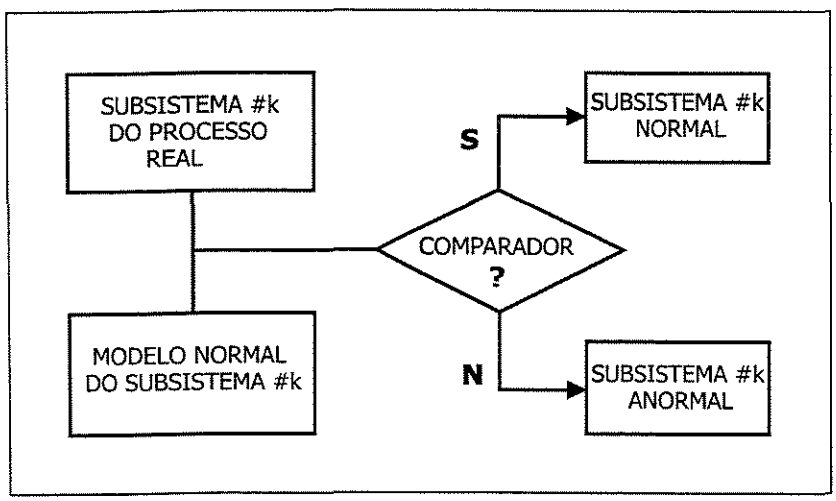


Figura (3.5): Topografia de Busca.

3.4.2 BUSCA SINTOMÁTICA

Um conjunto de observações representando o estado anormal do sistema pode ser usado como modelo para selecionar um subconjunto apropriado de situações, relacionados a diferentes condições anormais do sistema dentro de uma biblioteca de sintomas conhecidos. Este tipo de busca é chamada busca sintomática. A principal característica

deste método é que suas decisões são derivadas a partir das estruturas dos conjuntos de dados, suas relações anormais, e não a partir da estrutura topológica das propriedades do sistema. Busca sintomática é vantajosa do ponto de vista de economia de informações (um modelo não é essencial). Sua limitação é que um padrão de referência a um estado real de operação deve estar disponível, e falhas múltiplas ou distúrbios novos podem ser difíceis de serem identificados.

Em abordagem sintomática, as entradas dos processo também alimentam o modelo de referência. As saídas do processo são então comparadas às do modelo de referência. Quando os parâmetros no sistema não são diretamente medidos, há duas formas diferentes de identificar variações nos parâmetros entre o processo e o modelo:

Abordagem de malha fechada: a diferença entre a saída do modelo de referência e a saída do processo (erro residual) realimenta o modelo de referência e atualiza seus parâmetros tal que minimize o erro. Figura (3.6). Notar que estas abordagens não são limitadas para casos onde os modelos de referência são quantitativos.

Abordagem de malha aberta: os parâmetros do sistema são assumidos como sendo constantes no modelo de referência. Discrepâncias significativas entre o processo e o modelo são refletidas no erro residual. Muitos modelos de referência são usados com diferentes valores dos parâmetros. Figura (3.6). Comparando-se os erros residuais, o mais próximo do modelo de referência é identificado como o mais representativo do processo.

Em uma situação de estado finito, cada estado correspondendo a uma partição distinta do espaço de parâmetros, o processo pertence a um dos possíveis estados finitos. Neste caso, se todos os estados são explicitamente enumerados, tem-se uma abordagem de malha aberta. Se os estados são gerados e testados baseados na realimentação das discrepâncias, tem-se uma abordagem de malha fechada.

3.4.2.1 BUSCA ATRAVÉS DE HIPÓTESE E TESTE

Se uma busca é baseada em padrões de referência gerados *online* pela modificação de um modelo funcional, em correspondência com uma perturbação/falha hipotética, a

estratégia de busca é feita através de 'hipótese e teste' na forma de malha fechada. Figura (3.8). Nesta abordagem, hipóteses são geradas e testadas seqüencialmente. A eficiência desta busca depende da eficiência em gerar hipóteses. As hipóteses são geradas a partir de busca topográfica e de busca sintomática. Nesta abordagem, o diagnóstico se processa em três etapas: (a) formulação da hipótese, (b) determinação dos efeitos da falha tomada por hipótese (simulação da falha), e (c) comparação dos resultados com os da planta (teste das hipóteses). Se os sintomas preditos estão completa ou parcialmente na planta, a hipótese pode ser conservada, e o procedimento repetido até que nenhuma hipótese melhor possa ser encontrada. Quando a quantidade de falhas estiver muito grande, o conjunto pode precisar ser reduzido antes que as simulações das falhas sejam feitas. Conhecimento compilado ou heurístico é comumente usado para reduzir o conjunto de falhas a fim de se chegar ao conjunto de hipóteses definitivo.

3.4.2.2 GERAÇÃO DE HIPÓTESE EM UM ESPAÇO DE BUSCA FINITO

Um gráfico de hipóteses G(P) para um problema de diagnóstico fechado é desenvolvido através de um processo de geração de hipóteses e testes contínuos. A geração do G(P) é essencialmente fazer explícita uma porção de G(P) pelo procedimento de busca. O grafo começa com um nó de partida mais um conjunto de regras para gerar sucessivos nós a partir de qualquer nó não terminal até que uma dada condição de finalização seja alcançada. Se o conjunto de regras para gerar os nós sucessivos são heurísticos, então a geração de hipóteses é também heurística. A expansão geral do espaço de um grafo é normalmente feita com base em algum tipo de custo, onde uma função de avaliação decide, baseada nestes custos, qual dos nós deve ser aberto.

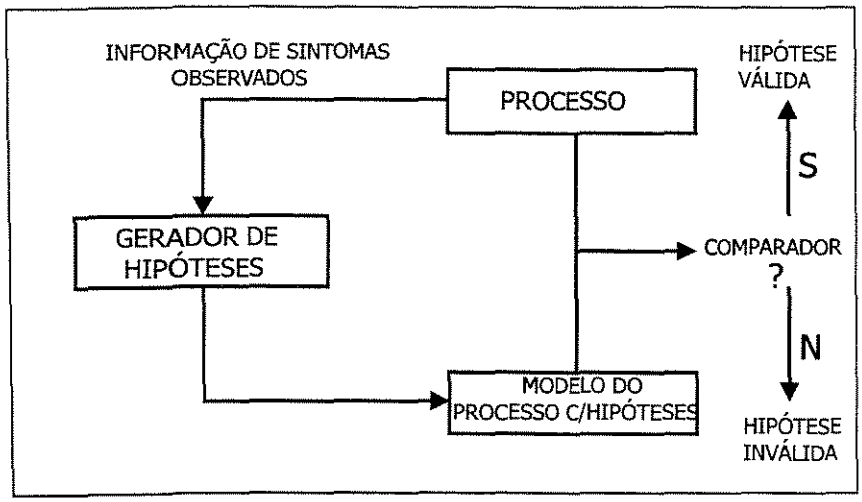


Figura (3.8): Dispositivo de geração e teste seqüencial.

3.4.2.3 ABORDAGEM HEURÍSTICA PARA A GERAÇÃO DE HIPÓTESES

PENG e REGGIA(1987a,b), desenvolveram um método probabilístico para superar a complexidade combinatorial da abordagem de hipóteses e testes quando múltiplas falhas são consideradas. A abordagem, enquanto limita o número de hipóteses geradas, ainda garante a identificação das hipóteses. Esta base faz uso da "teoria de cobertura parcimoniosa" ou a "Lâmina de Occum" como critério para a plausabilidade da hipótese, o qual diz que uma cobertura simples é preferível a uma complexa. Em outras palavras, a explicação mais simples é a explicação mais provável para a causa de uma falha do sistema. Os autores descrevem a probabilidade baseados numa abordagem para guiar a formação de hipóteses tal que as hipóteses mais prováveis são as primeiras. A limitação desta abordagem em senso prático é que ela requer extensivo conhecimento na forma de probabilidades numéricas.

3.5 ESTRATÉGIAS DE RACIOCÍNIO SIMBÓLICO

Raciocínio simbólico freqüentemente envia a abordagem para três tipos diferentes de raciocínio. Raciocínio abductivo, indutivo e raciocínio automático. Abdução é a geração de uma explicação hipotética (ou causa) para o que foi observado. Ao contrário da dedução lógica, pode-se conseguir mais que uma resposta no raciocínio abductivo. Uma vez que não existe um caminho geral para decidir entre alternativas, o melhor que se pode fazer é encontrar uma hipótese que seja a mais provável. Assim, abducção pode ser encarada como um raciocínio onde se pesa as evidências na presença de incerteza. A busca de uma causa para a anormalidade em um sistema de processo é um raciocínio abductivo. No MODEX2, que é um sistema especialista, baseado no modelo, para diagnóstico de falhas, VENKATASUBRAMANIAN e RICH(1998), usam o raciocínio abductivo para gerar hipóteses para as causas das falhas.

Aprendizagem indutiva é a classificação de um conjunto de experiências em categorias ou conceitos. Este tipo de raciocínio é desenvolvido quando se generaliza ou especializa uma definição conceitual aprendida tal que ela inclua todas as experiências que pertençam a um conceito e exclua aqueles que não. A definição clara de um conceito ou categoria é raramente simples por causa da grande variedade de experiências e incertezas (ruídos nos dados). Por esta razão, um esquema de aprendizagem adaptativa é preferida.

Freqüentemente se faz considerações automáticas dos valores de várias quantidades que são manipuladas, com a intenção de permitir que outros valores específicos prevaleçam sobre os valores correntes (por exemplo desde que a saída é bloqueada, o fluxo é zero), ou para se rejeitar a consideração automática se ela leva a uma inconsistência (por exemplo se a saída de um tanque é bloqueada, não pode haver decaimento do nível do mesmo). Uma característica fundamental do raciocínio automático é que ele é não monotônico. Em lógica tradicional, uma vez que o fato seja deduzido, ele é considerado verdadeiro para o resto do raciocínio. É a isto que se chama de monotônico. Entretanto, quando uma nova evidência surge, freqüentemente se necessita revisar os fatos deduzidos

para manter a consistência lógica. Considere o argumento prévio segundo o qual o nível do tanque não poderia decrescer (desde que a saída do tanque está bloqueada). Após esta dedução, se se conseguir mais alguma evidência que o tanque tem um grande vazamento, por exemplo, será preciso reformular a conclusão de que o nível do tanque não pode baixar. Tal raciocínio onde a retratação das deduções é permitida é também chamado de monotônico. Raciocínio automático ou monotônico é uma ferramenta sem valor para se lidar com situações onde as informações não estejam disponíveis a tempo ou que se tenha que lidar com muitos raciocínios simultaneamente.

3.5.1 ESTRATÉGIAS DE HIPÓTESE E TESTE COM MODELOS CAUSAIS

Há muitas abordagens para o projeto de sistemas especialistas, baseados em modelo, para diagnóstico de falhas em processos químicos. Uma destas abordagens é a estratégia de hipótese e teste, conforme o que está detalhado no item 3.4.2.1.

Um método rigoroso de formulação de hipótese é o método de O'SHIMA e seus colaboradores, descrito em SHIOZAKI et al.(1985). A premissa básica desta abordagem é que, para uma hipótese de falha ser considerada viável, as trajetórias causais devem ligar a origem da falha proposta a todas as medidas anormais observadas. Um dígrafo sinalizado é usado para a representação das influências entre as variáveis do processo. O algoritmo localiza todas as possíveis fontes de perturbações levando em conta o padrão da medida observada. Este algoritmo, entretanto, não funciona para respostas inversas e para falhas múltiplas.

O segundo aspecto deste método é a simulação da falha. Por causa das considerações de tempo real, simulações numéricas são impraticáveis. Dentro do esforço para se desenvolver métodos de simulação qualitativa, tem-se usado dígrafos sinalizados. Consideráveis trabalhos foram desenvolvidos neste campo, a fim de se modelar qualitativamente os sistemas e de se fazer a representação do conhecimento causal.

SIMON(1977), sugeriu o método de ordenamento causal que é usado para se chegar às relações causais através de uma redução adequada.

3.5.2 BUSCA EM ÁRVORE DE FALHA

Árvore de falha provê um meio computacional para combinar lógicas para analisar sistemas de falhas. Para se conseguir diagnósticos consistentes a partir de árvores de falhas, as árvores devem, compreensivelmente, representar as relações causais do processo. Para assegurar esta consistência, são desenvolvidos modelos causais em forma de dígrafos sinalizados. As árvores de causa de falha são desenvolvidas a partir destes dígrafos, LAPP e POWERS(1977). Árvores de falhas determinam caminhos de causa através dos quais eventos primários (falhas) podem se propagar através do sistema até causar um evento significativo (algum mal funcionamento significante). Dado um evento significante, árvores de falha desenvolvem, usando as informações do processo em forma de dígrafos, erros primários ou combinações dos eventos primários os quais podem resultar num evento significativo. A informação do processo é fornecida na forma de modelos das unidades e a topologia do processo mostra a conectividade da informação entre as unidades. Os modelos das unidades representam como as variáveis são relacionadas tanto quando as unidades estão funcionando normalmente e também quando falham. O problema da árvore de falha pode ser formulado como uma busca num espaço finito. Dado um estado inicial, o algoritmo aplica operadores que transformam o estado inicial em um estado desejado. O estado inicial é considerado um estado significativo. O estado-objeto é ter a árvore de falha conectando o estado significativo a todos os possíveis estados primários. Os dígrafos definem os vários estados possíveis. Operadores usam o dígrafo para responderem a questão "o que poderia causar isto?" em cada nó na árvore de falha.

Uma vez que a árvore de falha tenha sido montada, a informação proveniente dela é estocada em forma de pequenos conjuntos, os quais especificam todos os possíveis caminhos no dígrafo resultante da falha. KRAMER E PALOWITCH(1989) desenvolveram

um método para derivar regras para diagnóstico de falhas a partir de dígrafos sinalizados. Enquanto o método deles é muito eficiente e necessita somente de um passo através de todas as regras para diagnosticar a falha, tem uma séria limitação, só serve para casos de falhas simples. ULERICH e POWERS(1987) usaram modelos dígrafos para incluir ações do operador humano e modelos de falhas devido a ações do operador. Por exemplo, quando um incrustamento ocorre num trocador de calor, a temperatura da corrente quente aumenta. Isto forçará o controlador presente no tanque de resfriamento aumentar a quantidade de fluido refrigerante. Se a válvula para o resfriamento já estiver totalmente aberta, o operador pode intervir e diminuir o *set point* no controlador da corrente quente. O operador pode, acidentalmente, aumentar ao invés de diminuir o *set point*. Isto mostra um exemplo de falha induzida pelo operador.

3.6 ABORDAGENS BASEADAS EM MODELO QUANTITATIVO

Matematicamente representa as inconsistências entre os valores reais e os esperados, em forma de resíduos.

3.6.1 DETECÇÃO E ISOLAMENTO DE FALHA BASEADO EM OBSERVADOR

A principal preocupação do mecanismo de isolamento de falha baseado em observador é a geração de um conjunto de resíduos os quais detectam e de forma unívoca identifica falhas diferentes. A forma de determinação destes resíduos deve ser robusta de tal forma que as decisões não sejam corrompidas em face de entradas desconhecidas. Entradas desconhecidas aqui incluem incertezas não estruturadas tais como ruídos nos processos e nas medidas e incertezas de modelagem. O método é composto de um conjunto de

observadores onde cada um é sensível a um subconjunto de falhas enquanto é insensível às falhas remanescentes e às entradas desconhecidas. É possível se construir tais observadores usando graus de liberdade extras no projeto dos observadores e usando redundância nas medidas e nos modelos. A idéia básica é que se não existem falhas, a trajetória dos observadores segue o curso do processo e os resíduos dependem somente das entradas desconhecidas. Se um falha ocorre, todos os observadores insensíveis àquela falha continuam a apresentar pequenos resíduos refletindo somente as entradas desconhecidas. Por outro lado, observadores que sejam sensíveis para aquela falha não acompanham o processo e resultam em resíduos de magnitudes mais elevadas. O conjunto de observadores é projetado de tal forma que os resíduos a partir dos observadores resultam em padrões distintos para diferentes falhas tal que estas falhas possam ser unicamente identificadas a partir destes padrões. Indicações únicas para as falhas são garantidas por projetos onde os observadores mostram desacoplamento e invariância para distúrbios desconhecidos independente da falha e da natureza das perturbações.

Para ver como se constrói uma entrada desconhecida para um observador, considere o seguinte sistema:

$$\mathbf{x(k+1) = Ax(k) + Bu(k) + Ed(k) + Kf(k)} \tag{3.7}$$

$$\mathbf{y(k) = Cx(k)} \tag{3.8}$$

Um observador é um modelo dinâmico e da forma

$$\mathbf{xp(k+1) = Fxp(k) + Ju(k) + Gy(k)} \tag{3.9}$$

o objetivo é fazer o modelo seguir o curso do processo o mais próximo possível.

Definindo:

$$\mathbf{e(k+1) = \text{estimativa do erro ou caminho do erro do observador} = xp(k+1) - Tx(k+1)}$$

$$\mathbf{e(k+1) = Fxp(k) + Ju(k) + Gy(k) - TAx(k) - TBu(k) - TEd(k) - TKf(k)} \quad (3.10)$$

$$\mathbf{e(k+1) = Fxp(k) + (J - TB)u(k) + Gy(k) - TAx(k) - TEd(k) - TKf(k)} \quad (3.11)$$

Para que o observador siga o caminho[15] do sistema, independente das entradas desconhecidas $\mathbf{d(k)}$, é preciso tomar a matriz $\mathbf{T}$ de tal forma que $\mathbf{TE = 0}$. O caminho do observador pode ser feito independentemente da entrada do processo $\mathbf{u(k)}$ escolhendo-se a matriz $\mathbf{J}$ tal que $\mathbf{J = TB}$.

Então:

$$\mathbf{e(k+1) = Fxp(k) + Gy(k) - TAx(k) - TKf(k)}$$

$$\mathbf{e(k+1) = Fxp(k) + GCx(k) - TAx(k) - TKf(k)},$$

de (3.8)

$$\mathbf{e(k+1) = Fxp(k) + (GC - TA)x(k) - TKf(k)}$$

Escolhendo a matriz $\mathbf{G}$ tal que: $\mathbf{GC - TA = -FT}$, então

$$\mathbf{e(k+1) = Fxp(k) - FTx(k) - TKf(k)}$$

---

[15] Porque se refere a uma função.

$$\mathbf{e(k+1) = F(xp(k) - Tx(k)) - TKf(k)}$$

$$\mathbf{e(k+1) = Fe(k) - TKf(k)} \tag{3.12}$$

Se nenhuma falha ocorre no processo, $\mathbf{f(k) = 0}$ então

$$\mathbf{e(k+1) = Fe(k)} \tag{3.13}$$

Se o valor absoluto do autovalor de $\mathbf{F}$ é menor que 1, então $\mathbf{e(k) \to 0}$, quando $\mathbf{k \to \infty}$. Na ausência de qualquer falha, este observador seguirá o curso do processo independentemente das entradas desconhecidas $\mathbf{d(k)}$ do processo. Consequentemente, ele é conhecido como observador de entrada desconhecida.

Quando uma falha ocorre no sensor, a equação (3.8) se transforma em:

$\mathbf{y(k) = (C + \Delta C)x(k)}$, então:

o erro da trajetória do observador será:

$$\mathbf{e(k+1) = Fe(k) + G\Delta Cx(k)}$$

assim, o erro carrega a 'assinatura' do sensor de falhas.

Similarmente, uma falha no atuador será representada como mudança em $\mathbf{B}$, na equação (3.7), de tal modo que $\mathbf{B}$ muda para $\mathbf{B + \Delta B}$.

A trajetória do erro do observador será:

$$\mathbf{e(k+1) = Fe(k) - T\Delta Bu(k)} \tag{3.15}$$

3.6.2 ABORDAGEM ATRAVÉS DO ESPAÇO DE PARIDADE

Equações de paridade são formas convenientemente arranjadas do modelo entrada-saída da planta. O objetivo básico é checar a paridade (consistência) destes modelos da planta usando o sensor de saídas (medidas) e entradas conhecidas do processo. O valor das equações de paridade, chamado residual, sob condições de operações em estado estacionário ideal é zero. Sob reais condições, os resíduos não são zero devido a ruídos nas medidas e no processo, imprecisões no modelo, erros grosseiros nos sensores e nos atuadores, e falhas na planta. A idéia da abordagem é rearranjar a estrutura do modelo tal que se consiga o melhor isolamento das falhas. O isolamento de falhas requer a habilidade para gerar vetores residuais os quais sejam ortogonais a diferentes vetores representativos das falhas. BEN e HAIM(1980) usaram redundância no balanço de equações para gerar resíduos ortogonais para diferentes classes de falhas. Estenderam, num trabalho de (1983) a abordagem para o caso de sistemas dinâmicos para garantir a isolabilidade sob condições ideais.

3.6.2.1 ABORDAGEM ATRAVÉS DO ESPAÇO DE PARIDADE COM REDUNDÂNCIA ANALÍTICA DIRETA E/OU DE HARDWARE

Segundo esta abordagem, a consistência do modelo é melhor explicada seguindo-se a idéia básica de FRANK(1990). Assuma **y** como sendo um vetor (n x 1) das medidas. Assuma **x** como sendo os valores verdadeiros das **m** variáveis do processo. Existirá redundância se n > m. Sob condições livre de falha, **y** e **x** se relacionam de tal modo que

$$\mathbf{y} = \mathbf{C}\mathbf{x}$$

quando uma falha ocorre com uma das medidas, tem-se

$\mathbf{y} = \mathbf{C}\mathbf{x} + \Delta\mathbf{y}$. $\mathbf{C}$ é uma matriz n x m.

Fazendo $\mathbf{V}$ como sendo o espaço nulo de $\mathbf{C}$, tem-se que $\mathbf{V}$ é uma matriz ((n-m) x n) que satisfaz:

$$\mathbf{V}\mathbf{C} = \mathbf{0}$$

$$\mathbf{V}^{\mathrm{T}}\mathbf{V} = \mathbf{I}_{\mathrm{n}} - \mathbf{C}\left(\mathbf{C}^{\mathrm{T}}\mathbf{C}\right)^{-1}\mathbf{C}^{\mathrm{T}}$$

e as linhas de $\mathbf{V}$ são ortogonais, isto é, $\mathbf{V}\mathbf{V}^{\mathrm{T}} = \mathbf{I}_{\mathrm{n-m}}$.

Fazendo-se

$$\mathbf{p} = \mathbf{Vy} = \mathbf{VCx} + \mathbf{V}\Delta\mathbf{y} = \mathbf{V}\Delta\mathbf{y}$$

$\mathbf{p} = \mathbf{Vy}$ é o conjunto de equações de paridade cujos resíduos 'carregam a assinatura' das falhas nas medidas. Na ausência de qualquer falha, $\mathbf{p} = \mathbf{0}$.

Escolhendo-se uma falha no i-ésimo sensor isolado, tem-se:

$$\Delta\mathbf{y} = [\mathbf{0} \quad \mathbf{0} \quad \mathbf{0} \quad \Lambda \quad \Delta\mathbf{y}_i \quad \Lambda \quad \mathbf{0}]^T$$

$$\mathbf{V}\Delta\mathbf{y} = \Delta\mathbf{y}_i.(\mathbf{i} - \textbf{ésima coluna de } \mathbf{V})$$

Assim as colunas de V definem n direções distintas de falhas associadas às **n** falhas nos sensores. Isto assegura que a identificação de cada falha seja distinta e consequentemente garante sua identificação. O procedimento acima considera redundância direta. No próximo item, um esquema geral para a consideração da redundância temporal, simultaneamente com a direta, será apresentado.

3.6.2.2 PROCEDIMEMTO DE CHOW-WILLSKY PARA GERAR EQUAÇÕES DE PARIDADE

As equações de estado da planta são assumidas serem da forma:

$$\mathbf{x}(\mathbf{t}) = \mathbf{Ax}(\mathbf{t}-\mathbf{1}) + \mathbf{Bu}(\mathbf{t}-\mathbf{1}) \tag{3.16}$$

As medidas do processo (saídas) são assumidas serem da forma:

$$\mathbf{y(t) = Cx(t) + Du(t)} \tag{3.17}$$

das equações (3.16) e (3.17):

$$\mathbf{y(t+1) = CAx(t) + CBu(t) + Du(t+1)} \tag{3.18}$$

Similarmente, $\mathbf{y(t+s)}$, para s>0, é obtida como:

$$\mathbf{y(t+s) = CA^{s}x(t) + CA^{s-1}Bu(t) + ... + CBu(t+s-1) + Du(t+s)} \tag{3.19}$$

tomando-se estas equações para $\mathbf{s = 0, 1, ..., n}$, dá:

$$\mathbf{Y(t) = Qx(t-n) + RU(t)} \tag{3.20}$$

Multiplicando a equação (3.20) por $\mathbf{w^T}$, tem-se a seguinte equação escalar:

$$\mathbf{w^T Y(t) = w^T Qx(t-n) + w^T RU(t)} \tag{3.21}$$

Em geral, esta equação somente conterá variáveis de entrada, variáveis de saída e variáveis de estado desconhecidas. Ela será qualificada como uma equação de paridade somente se as variáveis de estado desaparecem. E isto requer que:

$$\mathbf{w^T Q = 0}$$

Isto é um conjunto de equações lineares homogêneas. Se o sistema é observável, estas **n** equações são independentes.

3.6.3 DETECÇÃO DE FALHAS BASEADA EM MODELAGEM E ESTIMAÇÃO

O diagnóstico de muitos parâmetros os quais não são mensuráveis diretamente requer métodos de estimação de parâmetros *on line*. Modelos de processos os mais reais possíveis são necessários, usualmente em domínio contínuo, e na forma de equações diferenciais parciais e ordinárias. Os modelos de sistemas mostrados no item anterior assumem parâmetros de processos como sendo constantes ou dependentes somente de variáveis de estado. Neste sentido, são sistemas de malha aberta.

Técnicas de estimação de parâmetros são uma versão de busca em malha fechada para os modelos de processos no espaço dos parâmetros. O problema de se manusear falhas que ocorrem como sendo dependentes do tempo pode ser feito através dos métodos de estimação de parâmetros.

Segundo ISERMANN(1984), este procedimento pode ser feito como segue.

Obtenha um modelo do processo somente com as entradas e saídas medidas na forma:

$$\mathbf{y}(\mathbf{t}) = \mathbf{f}(\mathbf{u}(\mathbf{t}), \Theta).$$

Os parâmetros do modelo, Θ, são estimados quando as medidas $\mathbf{y}(\mathbf{t})$ e $\mathbf{x}(\mathbf{t})$ estão disponíveis. As variáveis Θ estão relacionadas a parâmetros $\mathbf{p}$ no processo através de: $\Theta = \mathbf{g}(\mathbf{p})$. Mudanças nos parâmetros $\mathbf{p}$ e $\Delta\mathbf{p}$, são determinados através desta relação.

Usando métodos de reconhecimento de padrões, pode-se relacionar as mudanças em $\Delta \mathbf{p}$ às falhas do processo.

ISERMANN(1984) pesquisou diferentes técnicas de estimação de parâmetros tais como mínimos quadrados, variáveis instrumentais e estimativa via modelos de tempo discreto. Estes modelos requerem a disponibilidade de modelos dinâmicos precisos do processo e por isto são computacionalmente 'pesados' para grandes processos.

3.7 DIAGNÓSTICO DE FALHAS ATRAVÉS DE CLASSIFICAÇÃO POR PADRÕES

Sistemas especialistas têm uma habilidade limitada para aprender dinamicamente e melhorar seu desempenho. Além do mais, a tarefa de aquisição de conhecimento é de natureza altamente tediosa. Sistemas especialistas necessitam de um elevado esforço computacional e não são muito convenientes para tarefas em tempo real. Algoritmos de classificação por padrões mostram vantagens em relação aos sistemas especialistas nestes aspectos, o que tem estimulado o aparecimento de muitos trabalhos na literatura recente a respeito de aplicação de redes neurais para o diagnóstico de falhas através de classificação por padrões. Explorar o conhecimento do modelo do processo por meio de técnicas baseadas em técnicas de considerações possibilita uma forma melhor de classificação. Redes neurais têm mostrado resultados promissores consideráveis no seu uso como ferramenta para modelar sistemas não lineares e desta forma possibilitando o seu uso como referência para o controle de processos. Este trabalho está restrito à discussão a respeito de detecção e diagnóstico de falhas de forma que a utilização do procedimento de utilização de padrões para controle não será aqui detalhada.

3.7.1 CLASSIFICADORES PARAMÉTRICOS E NÃO PARAMÉTRICOS

Métodos de classificação estatísticos, como os descritos em FUKUNAGA(1982) e DUDA e HART(1973), podem ser classificados como métodos paramétricos ou não paramétricos. Métodos paramétricos assumem que a forma da função de densidade probabilística está disponível para as classes a serem classificadas. O problema do cálculo da função densidade então simplesmente resume-se à estimativa dos parâmetros desconhecidos na função densidade usando-se as amostras como padrões representativos. Por exemplo, se se sabe que os dados são normalmente distribuídos, necessita-se somente determinar a média e a matriz covariância para que a função densidade seja completamente conhecida. A vantagem de ter a função densidade é que se pode determinar a validade da classificação de um padrão particular estimando-se o erro através da função densidade.

Uma categoria importante de classificadores paramétricos clássicos é o classificador quadrático. Entre os classificadores quadráticos estão o classificador baseado em distância (não paramétrico) e o classificador de Bayes (paramétrico). Classificadores de distância usam a distância linear para calcular a distância de um dado padrão a partir das médias de várias classes e então atribuir o padrão a uma classe onde este esteja mais próximo. Entre os classificadores baseados em distância, o mais básico é o baseado na distância Euclidiana. É simples no princípio e fácil de implementar[16].

Quando a matriz covariância das distribuições não são matrizes identidade (ou seus múltiplos escalares), as distribuições não são esfericamente simétricas. Em tais casos, a distância preferida não deve ser a Euclidiana. Na verdade, mesmo quando as distribuições são esfericamente distribuídas, a distância Euclidiana só é conveniente quando as classes têm a mesma covariância. Para a distribuição Gaussiana, a medida da distância é uma forma adaptada da distância Euclidiana.

---

[16] Ver Capítulos 5 e 6 para os algoritmos de implementação.

O classificador Bayesiano[17] é um classificador ótimo (erro mínimo de classificação quando as classes se sobrepõem) quando as classes são distribuições Gaussianas e a informação sobre a distribuição está disponível.

Classificadores não paramétricos são procedimentos independentes das formas funcionais das densidades das probabilidades das classes e são ditos serem livres da distribuição. Entretanto, o procedimento troca a necessidade de considerações precisas sobre as formas funcionais das densidades, pela necessidade de uma grande quantidade de amostras para estimar bem as densidades.

3.7.2 REDES NEURAIS UTILIZADAS COMO CLASSIFICADORES

Classificadores baseados em redes neurais são não paramétricos e a escolha é feita somente quanto à forma funcional. A escolha da estrutura da rede é o mesmo que escolher a forma funcional do classificador. Entretanto, redes neurais são particularmente atrativas por causa da possibilidade de aproximar funções arbitrárias. Deste modo, para se classificar usando rede neural é preciso encontrar uma função discriminante que separe as classes e não encontrar funções densidade. Não se pode negar que a utilização da rede neural é menos recomendável que a utilização de classificadores paramétricos quando se tem disponível informações a respeito das densidades. No entanto, para o problema de classificação geral, como uma primeira escolha, não é conveniente se escolher através da densidade, mesmo porque, os dados disponíveis para a classe podem não ser suficientes para desenvolver as aproximações necessárias à função densidade.

Escolhendo-se uma topologia específica para a rede neural, o problema, então é reduzido à estimativa dos pesos. Isto faz o procedimento uma boa escolha para classificação de falhas, uma vez que as redes são capazes de gerar, consequentemente classificar, regiões arbitrárias no espaço, CYBENKO(1989).

---

[17] Ver Capítulos 5 e 6.

Redes neurais têm sido largamente usadas em problemas de classificação em geral e de classificação de falhas em particular. Uma boa amostra pode ser encontrada no capítulo 2 deste trabalho.

3.8 RELAÇÃO DO DIAGNÓSTICO DE FALHAS COM OUTRAS OPERAÇÕES DO PROCESSO

A necessidade de se controlar totalmente um processo inclui tarefas para a identificação de eventos que ocorrem fora das condições de normalidade do processo, diagnosticando a raiz do problema de modo a poder se implementar uma ação corretiva.

O diagnóstico de falhas compartilha com outras operações do processo a possibilidade de que, com esquemas de representação bem elaborados, se pode conseguir a representação do conhecimento especializado de operadores e engenheiros de controle, conseguidos em anos de experiência. O conhecimento específico do processo pode ser usado para melhorar metodologias de propósito geral. Por exemplo, considere o uso de modelos causais. Se se for usar métodos gerais de busca para identificar todas as fontes de mau funcionamento do sistema, o tempo de busca será exageradamente alto em comparação com as necessidades de diagnóstico em tempo real numa planta de grande escala. O conhecimento específico do processo, também chamado de conhecimento compilado do comportamento do processo, pode ser usado para para dar uma rápida identificação do potencial suspeito. Este conhecimento compilado pode ser adquirido ou aprendido durante as operações, RICH e VENKATASUBRAMANIAN(1987) e (1989). Há a necessidade de se projetar sistemas que raciocinem inteligentemente em tempo real e possibilite suporte para uma decisão a ser tomada, reduzindo assim as dificuldades dos operadores, pois dada uma planta de grande escala e diferentes requerimentos sobre informações das tarefas, é necessário raciocinar com o conhecimento a diferentes níveis de detalhes. Isto é muito difícil pois tem-se que assegurar a consistência da informação. Dado um conjunto de informações, é preciso que se organize de forma hierárquica as variáveis,

em ordem de importância de forma a extrair características qualitativas importantes do comportamento do processo a partir dos dados dos sensores.

3.9 CARACTERÍSTICAS DESEJÁVEIS DE UM SISTEMA DE DIAGNÓSTICO DE FALHAS

Há algumas características desejáveis que um sistema de diagnóstico deveria conter. Estas características são úteis para se comparar vários classificadores em termos de informações que eles necessitam, confiabilidade em suas soluções, sua generalidade e suas dificuldades computacionais. Segundo KASSIDAS et al.(1998), existe um conjunto de requerimentos básicos que define uma estrutura robusta para o método de diagnóstico de falhas baseado em reconhecimento de padrões.

1.RAPIDEZ NA DETECÇÃO: responder rapidamente à uma situação de falha.

Responder rapidamente a uma falha e ter um desempenho tolerável durante as condições de normalidade são duas metas conflitantes, WILLSKY(1976). Um sistema que é projetado para detectar uma falha (particularmente mudanças significativas), rapidamente, deve necessariamente ser sensível a influencias de alta freqüência. Isto por sua vez faz o sistema ser sensível a ruídos, o que pode levar freqüentemente a falsos alarmes durante a operação normal. Falhas abruptas, como as mudanças em degrau, são causadas principalmente por falhas de *hardware*[18] e são importantes para questões de segurança. A detecção imediata de falhas de mudanças de velocidades de escoamento, por exemplo, as quais podem ocorrer devido à degradação gradual do equipamento, tal como por incrustações, é importante para manutenção das características originais do funcionamento do processo. O tempo de detecção é crucial para falhas abruptas.

[18] Aqui entendidos como equipamentos da planta.

2.ISOLABILIDADE: Permitir saber onde se originou a falha.

Isto é conhecido como critério de isolabilidade e é a habilidade de um classificador distinguir entre diferentes falhas. Sob condições ideais, livre de ruídos e de incertezas de modelagem, significa dizer que o classificador deve ser capaz de gerar uma saída que seja ortogonal às falhas que não tenham ocorrido.

3. INDEPENDÊNCIA QUANTO À MAGNITUDE DA FALHA: Permitir se identificar uma falha independentemente de sua magnitude.

Uma mesma falha pode ocorrer a diversas magnitudes (por exemplo o bias no sensor, o passo de mudança na corrente de alimentação) e um sistema de detecção deve ser capaz de identificar a mesma corretamente mesmo que o padrão correspondente à falha conhecida seja diferente.

4. INDEPENDÊNCIA QUANTO À DURAÇÃO DA FALHA E QUANTO AO PONTO DE OCORRÊNCIA DA MESMA: Permitir identificar uma falha que ocorra a qualquer tempo durante sua operação.

O diagnóstico deve ser independente do tempo de duração da falha e do ponto de sua localização na planta. Processos contínuos funcionam a várias condições de operações dependendo das características da demanda e dos objetivos de qualidade. Níveis diferentes de produção resultam em dinâmicas mais lentas ou mais rápidas. Uma falha pode ocorrer em qualquer dos modos de operação, e devem ser classificadas corretamente.

5. DIREÇÃO DA FALHA: Permitir identificar uma falha positiva ou negativa.

Por exemplo, uma composição pode aumentar ou diminuir durante a operação de uma planta. Se a nova falha é negativa, o esquema de identificação deve ser capaz de identificá-la como tal, mesmo que o padrão para o caso positivo seja a única evidência no conjunto de dados.

6. ROBUSTEZ EM RELAÇÃO AO TEMPO DE OCORRÊNCIA DA FALHA: Permitir que uma falha seja detectada mesmo que somente seja conhecida dentro de uma janela de tempo específica, diferente da que se está utilizando na detecção da falha. O sistema deve ser robusto o suficiente para este tipo de incerteza, principalmente em uma implementação *on-line*.

7.ROBUSTEZ: Robusto em relação a ruídos do processo e às medidas bem como às incertezas de modelagem.

Robustez, em um sentido geral, é dado pelo princípio de degradação lenta o qual assegura que mesmo que se vá diminuindo a possibilidade de responder corretamente à questão de diagnóstico, no mínimo um aspecto da resposta correta o mecanismo deve prover. Isto não permite isolabilidade determinística, a qual permite testar *thresholds* próximos de zero. Em caso de ruído, estes *thresholds* podem ter sido colocados em limites mais conservadores, a fim de diminuir o efeito das interferências dos mesmos na classificação. Incertezas da modelagem podem surgir no caso de classificadores que utilizam o modelo do processo em seu projeto e o sucesso do detector de falha baseado no modelo depende fortemente da precisão do modelo que é usado.

Alternativamente, pode-se usar métodos de classificação que não dependam da disponibilidade de modelos precisos. Ao invés de modelo complexos, estes métodos somente requerem a especificação de modelos aproximados. Em muitos casos, o comportamento da planta nem é perfeitamente conhecido. Mesmo modelos rigorosos não são adequados para predizer o comportamento com precisão satisfatória, visto que, a configuração das plantas mudam durante a sua vida útil, sem contar que as condições de operação podem variar com as demandas dos diferentes produtos produzidos na planta. Tudo isto força o operador a fazer suas decisões de operação com modelos aproximados do comportamento do processo. Há técnicas para desenvolver modelos qualitativos e aproximados, fazendo raciocínios inexatos, a fim de contornar situações como a apresentada. O problema está na precisão das predições. Disto surgem as relações entre a qualidade e a resolução. Qualidade basicamente diz que ao invés do compromisso com a precisão da solução (identificação da falha), permite que a solução especifique mais que

uma possível causa da falha com a garantia de que a verdadeira causa está entre elas. A resolução basicamente diz quão pequeno é o conjunto em relação ao número total de falhas.

8.IDENTIFICABILIDADE DE FALHAS NOVAS: Distinguir entre a situação normal e a anormal e distinguir novos comportamentos anormais a partir de comportamentos anormais conhecidos.

Na prática, a partir das operações do dia-a-dia da planta, padrões suficientes do comportamento normal podem ser anotados o que torna este tipo de comportamento bem definido. Quando um padrão cai em uma região normal, o classificador de falha deve ser capaz de identificar que o processo está normal. De outra forma, deve reconhecer a operação como anormal. Para comportamento anormal, muito menos padrões estão disponíveis seja por meio de simulações, seja pelo histórico do processo, cobrindo deste modo somente porções da região de anormalidade. No caso de comportamento anormal, o classificador deveria identificar se o padrão está numa região cuja classe da falha seja conhecida e desta forma identificá-la. Se o comportamento for um padrão novo, o classificador deverá indicar o comportamento anormal indicando que sua causa é desconhecida.

9.ADAPTABILIDADE: Facilidade de adaptação às mudanças de condições de operação do sistema.

Um sistema deve ser gradualmente desenvolvido de forma a poder ser atualizado quando novos casos e problemas surgem, ou quando mais informações se tornam disponíveis. As condições de operação do processo podem mudar não somente devido a perturbações mas também devido a mudanças no meio ambiente, tais como mudanças nas quantidades produzidas, de acordo com a demanda, mudanças na qualidade das matérias-primas, etc.. Quando estas mudanças são graduais, é melhor modificar o classificador também de forma gradual, à medida que as informações a partir das medidas do processo se tornam disponíveis.

10.IDENTIFICABILIDADE DE MÚLTIPLAS FALHAS: Capacidade de manipular situações com mais que uma falha.

A capacidade de identificar múltiplas falhas é importante e é muito difícil por algumas abordagens. Abordagens qualitativas usando modelos causais administram problemas de complexidade combinatorial os quais consideram múltiplas falhas. Este problema pode às vezes ser contornado assumindo-se que as falhas não estão interagindo de tal forma que possam ser separadas e identificadas.

11.FACILIDADE DE EXPLICAÇÃO: Explicar a seqüência de eventos que podem ter levado ao mal funcionamento.

Ao lado da capacidade de identificar a fonte do mau funcionamento, um sistema de diagnóstico deveria também prever uma forma de explicar como a falha se propagou levando à situação atual. Isto requer a possibilidade de raciocinar sobre as relações de causa-efeito. Um sistema de diagnóstico precisa justificar suas recomendações de tal modo que o operador possa consequentemente avaliar e agir usando sua experiência.

12.INTEGRAÇÃO: Para ser um módulo em um sistema integrado, o procedimento não deve comprometer os outros módulos.

Um requerimento importante e geral que o sistema de diagnóstico precisa satisfazer para ser um módulo em um sistema grande e integrado é no mínimo o princípio do comprometimento., onde o módulo não interfira nas demais partes do sistema não analisado. Ele basicamente requer que o sistema não faça alguma coisa que precise ser desfeita mais tarde. Isto significa que o módulo não pode pressupor a função de um outro módulo, não pressupor que um outro módulo vá executar uma função que seja dele. O módulo de diagnóstico, para ser parte de um sistema integrado, deve esperar o mínimo de comprometimento dos outros módulos. Isto é importante para o próprio funcionamento dos outros módulos.

13.REQUERIMENTOS COMPUTACIONAIS:

Há uma cumplicidade entre a complexidade computacional e o desempenho do sistema. É desejável ter um projeto que possa explorar os instrumentos computacionais tais como implementações modulares ou paralelas.

3.10 COMPARAÇÃO DAS DIFERENTES ABORDAGENS

Sistemas especialistas podem ser usados onde as exigências dos princípios fundamentais esteja faltando, ou seja, onde exista uma abundância de experiência mas não existam detalhes disponíveis suficientes para se desenvolver um modelo quantitativo preciso. Modelos causais são uma boa alternativa quando os modelos quantitativos não estão disponíveis mas as dependências funcionais são compreendidas. O raciocínio qualitativo pode ser computacionalmente intensivo para processos em grande escala. Abstração hierárquica ajuda a focar a atenção do sistema de diagnóstico rapidamente em áreas-problema. Uma das vantagens dos modelos qualitativos baseado no conhecimento profundo é que eles podem prover uma explicação do caminho de propagação da falha. Isto é indispensável quando se torna suporte para decisão do operador.

O raciocínio qualitativo usado nos modelos causais têm o mérito de garantir que o comportamento do processo está presente entre os muitos possíveis comportamentos que resultam. Além disto, é atrativo para domínios mal definidos em problemas como diagnóstico onde tem-se que lidar com conhecimento incompleto e incerto. Entretanto, o número de comportamentos gerados pelos modelos qualitativos pode ser muito grande dando uma resolução mais pobre que a que se pode obter a partir de modelos quantitativos. Um modo de melhorar a resolução é explorando a redundância. Alternativamente, pode-se usar uma elegante escala de raciocínio chamada de Análise de Ordem de Magnitude, devida a MAVROVOUNIOTIS e STEPHANOPOULOS(1987) e a Análise do Intervalo.

O raciocínio quantitativo sozinho não pode manusear informações incompletas e não pode garantir qualidade no diagnóstico pois existe uma lacuna na sua capacidade de explicação, enquanto o raciocínio simbólico permite um raciocínio de alto nível. A melhor

abordagem seria uma estrutura que combinasse modelos com raciocínio qualitativo ao quantitativo. Isto requer uma metodologia que (a) permita representar os modelos qualitativos e quantitativos de uma forma uniforme, e (b) possa raciocinar e desenvolver a busca de um modo uniforme através das alternativas existentes. Raciocínios qualitativos e quantitativos podem ser incorporados dentro dos conjuntos em consideração. Tais abordagens possibilitam uma estrutura geral para integrar raciocínios qualitativos e quantitativos. Estas abordagens desenvolvem duas tarefas: (a) avaliar a consistência dos grupos de considerações, e (b) atribuir crédito às inconsistências das diferentes considerações. Raciocínio qualitativo e/ou quantitativo pode ser incorporado à avaliação de conjuntos de considerações. A avaliação pode ser desenvolvida através de lógica simbólica ou pela avaliação de restrição quantitativa. Após o conjunto de considerações ter sido avaliado quanto à consistência, a atribuição de crédito deve ser desenvolvida tal que as considerações que são fontes de inconsistência possam ser identificadas.

Espaço de paridade e abordagens baseadas no observador são mostradas como sendo equivalentes em GETLER(1991). Méritos e deméritos de um grupo são conduzidos para o outro. Se se tivesse conhecimento de todas as entradas e saídas do sistema, incluindo todas as formas de interação com o meio ambiente, o diagnóstico de falhas seria apenas um problema bem definido de consideração do número de falhas presentes. Por outro lado, se há somente um sensor simples indicando se o sistema está operando em condições normais ou de falha, então nada pode ser diagnosticado inclusive o próprio funcionamento do sensor. A eficiência de qualquer procedimento de diagnóstico está limitado pela disponibilidade das informações dos sensores. Na prática, a informação dos sensores está entre dois extremos. Os modelos quantitativos permitem deduzir de antemão, dado uma escolha particular de medidas, se a falha pode ser isolada. Eles também permitem projetar modelos nos quais as perturbações desconhecidas podem ser minimizadas. Entretanto, devido à natureza do seu procedimento, não contemplam uma explicação fácil. Quando se considera um processo em larga escala, o tamanho do banco de filtros pode ser muito grande, aumentando a complexidade computacional.

O estado geral do espaço de um modelo é dado por:

$$\mathbf{x}(\mathbf{k}+\mathbf{1}) = \mathbf{F}\mathbf{x}(\mathbf{k}) + \mathbf{G}\mathbf{u}(\mathbf{k}) + \mathbf{E}_1\mathbf{d}(\mathbf{k}) + \mathbf{F}_1\mathbf{f}(\mathbf{k}) \tag{3.23}$$

$$\mathbf{y(k) = Cx(k) + Du(k) + E_2(k) + F_2 f(k)} \tag{3.24}$$

sendo $\mathbf{E_1}$ e $\mathbf{E_2}$ as matrizes distribuição dos distúrbios $\mathbf{d}$, os quais, no caso geral, inclui incertezas estruturadas e não estruturadas. O termo $\mathbf{Ed(k)}$ caracteriza a entrada desconhecida ou o distúrbio e representa todas as incertezas agindo sobre o sistema. O principal problema está na aproximação simplista das perturbações que inclui a modelagem dos erros. Entretanto, para todas as abordagens com distúrbios desacoplados, uma consideração importante é que a sua distribuição (matriz $\mathbf{E}$) seja conhecida. No projeto de um observador de entrada não linear, SELIGER e FRANK(1991), modelos mais gerais foram considerados:

$$\frac{\mathbf{dx}}{\mathbf{dt}} = \mathbf{A(x) + B(x)u + E(x)d + K(x)f} \tag{3.25}$$

$$\mathbf{y = C(x)} \tag{3.26}$$

Neste caso, $\mathbf{d}$ inclui somente incertezas aditivas. Na prática, entretanto, modelagens completas de incertezas devido à tendências nos parâmetros são colocados na equações na forma de incertezas multiplicativas.

Os classificadores são construídos baseados somente nos dados históricos do processo. Redes neurais já demonstraram que funcionam muito bem em termos de robustez e de capacidade de isolamento dos requerimentos. Contudo, há algumas limitações para os métodos que são baseados somente nos dados do histórico do processo. É a sua limitada capacidade de generalização fora do conjunto de dados usados para treinamento. Este problema é amenizado utilizando-se funções radiais[19] ou elipsoidais, evitando-se assim tomar-se uma decisão em caso que não haja padrões de treinamento similares naquela região. Isto permite à rede detectar situações novas do processo, quando falhas novas aparecem. Além disto, por causa desta característica de má 'extrapoladora', redes neurais

têm dificuldade para tratar dados com múltiplas falhas[20]. Este é um ponto crucial na distinção entre os procedimentos que usam abordagens baseadas em modelo e classificadores baseados no histórico do processo. A limitação geral dos métodos baseados no histórico do processo não está nos classificadores que fazem parte deles mas na distribuição das classes no espaço de medidas. Considere por exemplo um espaço de medidas tridimensional como o mostrado na Figura (3.10). Três classes de falhas são mostradas, representadas por círculos sombreados, dispostas em três vértices de um cubo. Três padrões de medidas – x1, x2 e x3 – são considerados no espaço das medidas para generalização. Sem assumir qualquer classificador, pode-se esperar que:

x1 pertença às classes 1 e 2

x2 pertença às classes 1 e 3

x3 pertença às classes 2 e 3

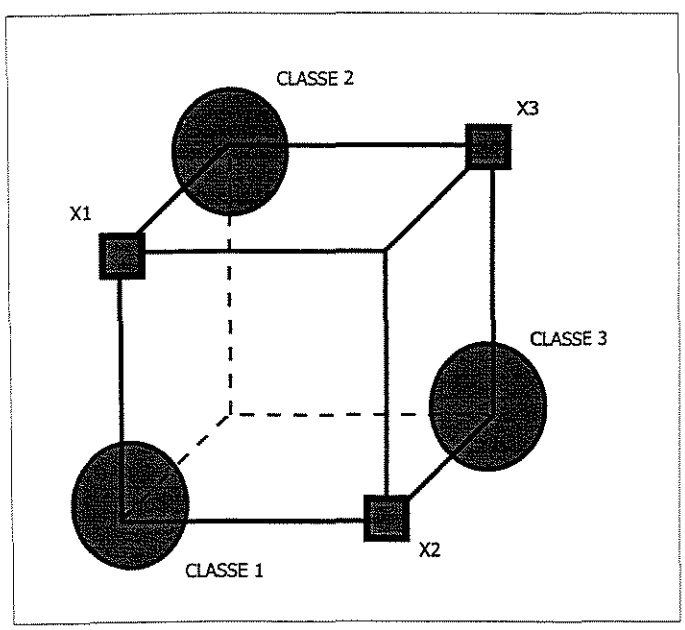


Figura (3.10): Localização de múltiplas falhas em um espaço de saída.

[19] Porque delimitam uma região no espaço em torno de um centro dado.
[20] Ver no capítulo de reseultados numéricos a abordagem proposta a fim de se contornar esta deficiência.

Se se considera transformações lineares, então pode-se esperar uma estrutura de dados semelhantes tanto à entrada quanto à saída. Isto revela um ponto importante – que a estrutura do espaço de entrada tem fortes implicações no espaço de saída esperado.

Se se usa como entrada os resíduos das equações de paridade, devido à sua propriedade de isolabilidade, as entradas seriam como esperado na Figura (3.10). Assim a estrutura requerida é imposta pelo espaço de entrada. Isto sugere que, quando informações do modelo estão disponíveis, deve-se usar um banco de filtros para gerar as entradas. Como bancos de filtros são computacionalmente caros, pode-se usar de forma eficiente as hierarquias de abstração a fim de reduzir rapidamente o foco do diagnóstico e então usar os filtros somente na região do foco.

3.11 DESEMPENHO DO SISTEMA DE DETECÇÃO/DIAGNÓSTICO

O desempenho de um sistema de teste é avaliado determinando-se a porcentagem de diagnósticos feita com sucesso e o tamanho médio do conjunto de ambigüidades (quando o sistema isola uma falha em um conjunto possível contudo não consegue indicá-la de forma única). Em sistemas químicos, deve-se lidar com sensores cujas medidas podem estar corrompidas por ruídos, sistemas funcionando fora de seu estado nominal de operação e mais importante, modelos imprecisos da unidade sob teste.

CAPÍTULO 4

4.1 MODELO DINÂMICO DE UMA COLUNA DE DESTILAÇÃO

Por definição, processos de destilação separam os componentes de uma mistura fluida baseada em seus diferentes pontos de ebulição, removendo e adicionando calor para a condensação e para a evaporação, respectivamente.

A separação é conseguida porque, usualmente, existe uma diferença de composição entre uma mistura de líquido e o vapor em equilíbrio com o mesmo. Por gerenciamento cuidadoso da vaporização e condensação de uma mistura de líquidos, a separação dos componentes puros pode ser conseguida.

Cada um dos procedimentos de resolução das equações do modelo dinâmico está preocupado em encontrar uma solução em estado estacionário.

Os motivos que levam a um melhor modelo para simulação de uma coluna de destilação estão intimamente ligados à caracterização que se faça do soistema líquido-vapor e à disponibilidade e confiabilidade de bancos de dados físico-químicos.

Hipóteses simplificadoras consideradas para se descrever o sistema, SAMPAIO (1994),

1.Não existem trocas de calor da coluna com o meio (adiabática),

2. O comportamento da pressão ao longo da coluna obedece a uma função linear,

3. Não são considerados os efeitos térmicos da mistura,

4. O condensador e o refervedor têm volumes de líquidos constantes durante a operação,

5. O condensador é total,

6. Há equilíbrio termodinâmico de fases no refervedor,

7. Existe uma relação definida entre as composições de líquido e vapor que deixam o prato,

8. O líquido e o vapor deixam o prato em equilíbrio térmico,

9. A composição é constante ao longo do prato e de valor igual à do líquido que o deixa,

10. Despreza-se o acúmulo de vapor,

11. São desprezados, nas equações de balanço, os efeitos de choro de líquido e arraste de líquido e vapor,

12. Não é considerada saída lateral de vapor.

NC: número de componentes considerados

NS: número de pratos

4.1.1 BALANÇO DE MASSA

Sejam os componentes representados por **i**, e cada um dos pratos representados por **j** conforme a Figura (4.1).

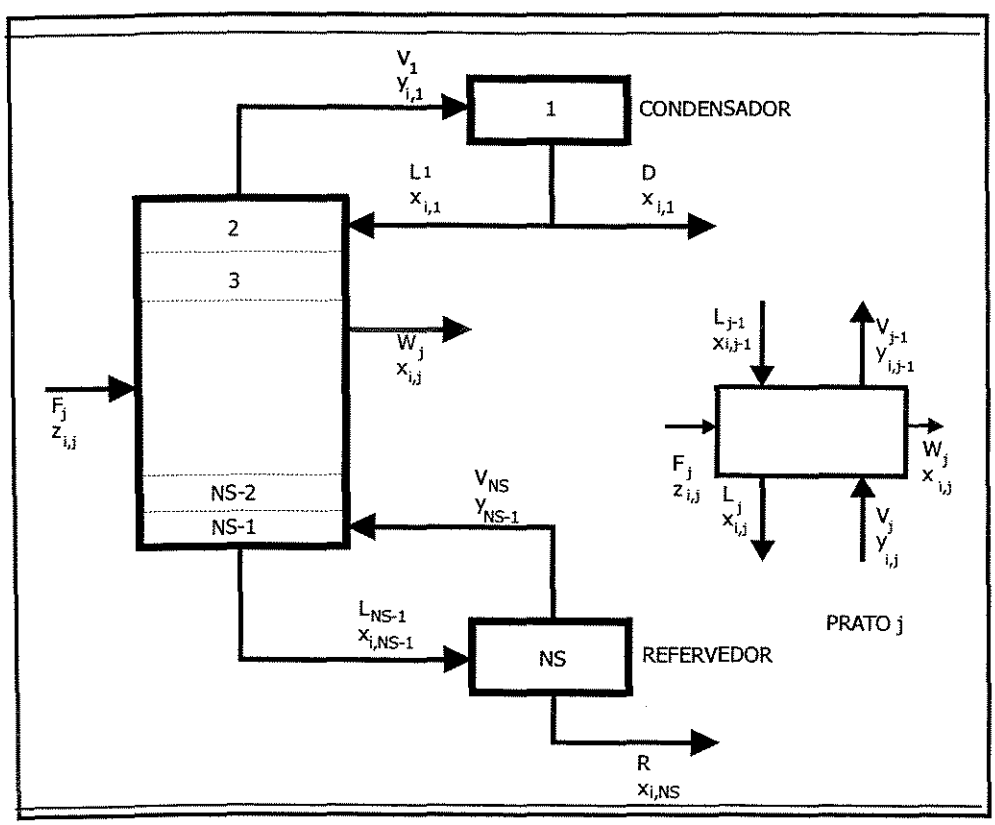


Figura (4.1): Esquema da Coluna de Destilação – Modelo Clássico.

O balanço de massa para os componentes, em cada prato:

$$\frac{d(M_j.x_{ij})}{dt} = L_{j-1}.x_{i,j-1} + V_j.y_{i,j} + F_j.z_{i,j} - V_{j-1}.y_{i,j-1} - L_j.x_{i,j} - W_j.x_{i,j}, \qquad (4.1)$$

$j = 2,...,NS-1$

O balanço global de massa em cada prato:

$$\frac{dM_j}{dt} = L_{j-1} + V_j + F_j - L_j - V_{j-1} - W_{j-1}, \qquad j = 2, \ldots, NS-1 \qquad (4.2)$$

O balanço de massa para cada componente no refervedor

$$\frac{d(M_{NS}.x_{i,NS})}{dt} = L_{NS-1}.x_{i,NS-1} - V_{NS-1}.y_{i,NS-1} - R.x_{i,NS} \qquad (4.3)$$

O balanço de massa global no refervedor

$$\frac{dM_{NS}}{dt} = L_{NS-1} - V_{NS-1} - R \qquad (4.4)$$

O balanço de massa para cada componente no condensador

$$\frac{d(M_1.x_{i,1})}{dt} = V_1.y_{i,1} - L_1.x_{i,1} - D.x_{i,1} \qquad (4.5)$$

O balanço global no condensador

$$\frac{dM_1}{dt} = V_1 - L_1 - D \tag{4.6}$$

tem-se que

$$\frac{d(M_j . x_{i,j})}{dt} = x_{i,j} . \frac{dM_j}{dt} + M_j . \frac{dx_{i,j}}{dt}, \tag{4.7}$$

Combinando-se assim as equações de balanço de massa globais com as equações de balanço de massa por componente, tem-se

Para cada prato j

$$M_j \frac{dx_{i,j}}{dt} = L_{j-1} . x_{i,j-1} + V_j . y_{i,j} + F_j . z_{i,j} - V_{j-1} . y_{i,j-1} - L_j . x_{i,j} -$$

$$W_j . x_{i,j} - \left[x_{i,j} . (L_{j-1} + V_j + F_j - L_j - V_{j-1} - W_j)\right], \quad j = 2, \ldots, NS-1 \tag{4.8}$$

Para o refervedor

$$M_{NS} . \frac{dx_{i,NS}}{dt} = L_{NS-1} . (x_{i,NS-1} - x_{i,NS}) + V_{NS-1} . (x_{i,NS} - y_{i,NS-1}) \tag{4.9}$$

Para o condensador

$$M_1 . \frac{dx_{i,1}}{dt} = V_1 . (y_{i,1} - x_{i,1}) \tag{4.10}$$

4.1.2 BALANÇO DE ENERGIA

Para cada prato

$$\frac{d(M_j.h_j)}{dt} = L_{j-1}.h_{j-1} + V_j.H_j + F_j.hf_j - L_j.h_j - V_{j-1}.H_{j-1} - W_j.h_j, \qquad (4.11)$$

j = 2, ... , NS-1

Para o refervedor

$$\frac{d(M_{NS}.h_{NS})}{dt} = L_{NS-1}.h_{NS-1} - V_{NS-1}.H_{NS-1} - R.h_{NS} + Q_R \qquad (4.12)$$

Das equações do balanço de massa e de energia, pode-se escrever

Para cada prato

$$M_j.\frac{dh_j}{dt} = L_{j-1}.h_{j-1} + V_j.H_j + F_j.hf_j - L_j.h_j - V_{j-1}.H_{j-1} - W_j.h_j - h_j.(L_{j-1} + V_j + F_j - L_j - V_{-1j} - W_j) \quad , j = 2 , \dots , NS-1 \qquad (4.13)$$

Para o refervedor

$$M_{NS}.\frac{dh_{NS}}{dt} = L_{NS-1}(h_{NS-1} - h_{NS}) + V_{NS-1}.(H_{NS} - H_{NS-1}) + Q_R \qquad (4.14)$$

4.2 MODELO TERMODINÂMICO. EQUILÍBRIO LÍQUIDO-VAPOR

O critério termodinâmico a ser observado a fim de se ter equilíbrio termodinâmico entre uma fase líquida e uma fase vapor é dada pelo sistema de equações

$$\mathbf{f}_i^v = \mathbf{f}_i^l, \text{ para } i = 1,2, \ldots, NC \qquad (4.15)$$

onde

$\mathbf{f}_i^v$ é a fugacidade do componente i na fase vapor

$\mathbf{f}_i^l$ é a fugacidade do componente i na fase líquida

As equações (4.15), são no entanto, de pouca utilidade prática, e desta forma tem-se que variáveis auxiliares precisam ser definidas, como por exemplo o coeficiente de atividade e o coeficiente de fugacidade. Diferentes métodos podem ser utilizados na determinação destes coeficientes, PRAUSNITZ et al.(1980), FREDESLUND et al.(1977).

4.2.1 COEFICIENTE DE FUGACIDADE

Apesar de na modelagem apresentada neste trabalho ter-se considerado a possibilidade de determinação do coeficiente de fugacidade, na prática, a fase gasosa foi considerada como tendo comportamento ideal. Esta consideração foi adotada porque, segundo POIANI(1984), o tratamento termodinâmico para o sistema metanol-água, que é o

utilizado neste trabalho, pode ser simplificado, quando se tem baixas pressões, considerando-se o coeficiente de fugacidade como sendo unitário. Pode-se mostrar como se faz o cálculo deste coeficiente seguindo os passos descritos em PRAUSNITZ et al(1980).

Para um componente numa mistura gasosa, o coeficiente de fugacidade é definido por:

$$\psi_i = \frac{\mathbf{f_i^v}}{\mathbf{y_i P}} \tag{4.16}$$

Utilizando-se a equação virial de estado, truncada após o segundo termo, tem-se:

$$\mathbf{\ln \psi_i} = \frac{\mathbf{P}}{\mathbf{RT}}\left(2\sum_j \mathbf{Y_j B_{i,j}} - \mathbf{B}\right) \tag{4.17}$$

com

$$\mathbf{B} = \sum_i \sum_j \mathbf{Y_i Y_j B_{i,j}} \text{ , } i,j = 1,2, \ldots, NC \tag{4.18}$$

onde

$\mathbf{B_{i,j}}$ representa a interação entre as moléculas dos componentes i e j.

A dificuldade em se encontrar valores para os parâmetros da equação (4.17), faz com que muitas vezes os valores dos coeficientes tenham que ser calculados através da teoria química. Deve-se saber também que para compostos a pressões elevadas, uma equação mais conveniente que a virial é a equação de estado de Ridlich-Kwong. A temperatura e a pressão do sistema determina as condições de utilização de uma ou outra.

Para a determinação do parâmetro $\mathbf{B_{i,j}}$ pode-se utilizar uma correlação apresentada por TSONOPOULOS(1974) e que segundo o autor é ligeiramente superior à de Pitzer e Curl, apresentada por PRAUSNITZ(1980), para misturas não associadas ou fracamente associadas como por exemplo o metanol-água.

4.2.2 COEFICIENTE DE ATIVIDADE

Diferentemente da fase vapor, a fase líquida não foi considerada como ideal, uma vez que esta consideração é muito menos válida que a outra na maioria dos casos.

O coeficiente de atividade para um componente i na fase líquida é definido por:

$$\gamma_i = \frac{\mathbf{f_i^l}}{\mathbf{f_i^{ol}}.\mathbf{x_i}} \tag{4.19}$$

Esta equação relaciona a fugacidade do componente na fase líquida à sua fração molar e à uma fugacidade no estado de referência, $\mathbf{f_i^{ol}}$.

Os coeficientes de atividade estão diretamente relacionados à energia molar em excesso de Gibbs, $\mathbf{g^E}$, a qual é dada por

$$\mathbf{g^E} = \mathbf{R.T.}\sum_i \mathbf{x_i}.\ln\gamma_i \tag{4.20}$$

ou de outra maneira

$$\ln\gamma_i = \frac{\mathbf{1}}{\mathbf{R.T}}.\frac{\partial\left(\mathbf{n_t}.\mathbf{g^E}\right)}{\partial\mathbf{n_i}}, \qquad \mathbf{T,P,n_{j\neq i}} \tag{4.21}$$

onde n_i é o número de moles do componente i. E,

$$\mathbf{n_t = \sum_j n_j} \qquad \text{e} \qquad \mathbf{x_i = \frac{n_i}{n_t}} \tag{4.22}$$

Dentre os modelos existentes na literatura para a determinação do coeficiente de atividade, os mais utilizados são segundo PRAUSNITZ(1980), o de Wilson, o NRTL e UNIQUAC. No programa de simulação aqui desenvolvido foram implementados os modelos UNIQUAC e Wilson, de forma a se poder usar um ou outro conforme o sistema e conforme a disponibilidade de valores para os parâmetros referentes a cada substância.

4.2.2.1 MODELO UNIQUAC

O modelo UNIQUAC é devido a Abrams.

$$\mathbf{\ln \gamma_i = \ln \frac{\phi_i}{x_i} + \left(\frac{Z_e}{2}\right).q_i.\ln \frac{\theta_i}{\phi_i} + l_i - \frac{\phi_i}{x_i}\sum_j x_j l_j - q_i'.\ln\left(\sum_j \theta_j'.\tau_{j,i}\right) + q_i' - q_i'\sum_j \frac{\theta_j'.\tau_{i,j}}{\sum_k \theta_k'\tau_{k,j}}}$$

(4.23)

onde

$$\tau_{j,i} = \exp\left(-\frac{a_{j,i}}{T}\right) \qquad \text{e} \qquad \tau_{i,j} = \exp\left(-\frac{a_{i,j}}{T}\right) \tag{4.24}$$

$$l_j = \frac{Z_e}{2}(r_j - q_j) - (r_j - 1) \tag{4.25}$$

$$\phi_i = \frac{r_i x_i}{\sum_j r_j x_j} \tag{4.26}$$

$$\theta_i = \frac{q_i . x_i}{\sum_j q_j . x_j} \qquad \text{e} \qquad \theta_i' = \frac{q_i' . x_i}{\sum_j q_j' . x_j} \tag{4.27}$$

Os parâmetros **r**, **q**, $\mathbf{q}'$ e $\mathbf{Z_e}$ são constantes cujas tabelas de valores estão disponíveis em PRAUSNITZ(1980).

Os parâmetros binários $\mathbf{a_{j,i}}$ e $\mathbf{a_{i,j}}$ são ajustáveis a partir de dados experimentais e são função da temperatura. À falta de dados relacionando esta dependência, despreza-se a influência da temperatura e à falta de dados experimentais, pode-se também utilizar a contribuição dos grupos para estimá-los.

4.2.2.2 MODELO DE WILSON

Conforme a apresentação de REID et al.(1987), o coeficiente de atividade calculado pela correlação de Wilson tem a seguinte fórmula:

$$\ln \gamma_i = 1 - \ln\left(\sum_j x_j . G_{i,j} \right) - \sum_j \left(\frac{x_j . G_{j,i}}{\sum_k x_k . G_{j,k}} \right) \tag{4.28}$$

onde

$$G_{i,j} = \frac{v_{m_j}}{v_{m_i}} . \exp\left(\frac{-a_{i,j}}{R.(T + 273{,}15)} \right), \text{ para } i \neq j \tag{4.29}$$

$$G_{i,j} = 1{,}0, \text{ para } i = j \tag{4.30}$$

Esta correlação tem a vantagem de considerar $\mathbf{a}_{i,,j}$ como sendo independente da temperatura e da composição de modo a se poder usar o valor de $\mathbf{a}_{i,,j}$ de uma mistura binária no cálculo de uma mistura múlti componente, mesmo que a uma temperatura diferente. Para o caso de uma mistura de 3 ou mais componentes, a correlação de Wilson pode ser substituída pela correlação de Chao-Sader, por não exigir o conhecimento de parâmetros de interação binária e de fornecer bons resultados quando aplicada a uma mistura de hidrocarbonetos, segundo REID et al.(1987).

Esta última equação é dada por:

$$\ln \gamma_i = \frac{Vb_i \cdot \left(s_i - \sum_m \delta_m . s_m \right)^2}{RT} \tag{4.31}$$

e

$$\delta_m = \frac{x_m . Vb_m}{\sum_k x_k . Vb_k} \tag{4.32}$$

4.2.3 CÁLCULO DA FUGACIDADE NO ESTADO PADRÃO PARA COMPONENTES CONDENSÁVEIS

O coeficiente de atividade fica completamente definido somente se a fugacidade no estado padrão, f_i^{ol}, estiver claramente definida. Esta definição é arbitrária e feita de forma conveniente para as condições de operação do processo.

Em misturas a baixas pressões e longe das condições críticas, os coeficientes de atividade podem ser considerados independentes da pressão. Nestas condições é comum se adotar como referência o componente **i** puro à temperatura e pressão de equilíbrio, ou seja:

$$f_i^{ol} = f_i^S . \exp \int_{P_i^S}^{P} \frac{v_i^L}{RT} dp \tag{4.33}$$

onde,

$$f_i^S = \phi_i^S . P_i^S \tag{4.34}$$

ϕ_i^S = **coeficiente de fugacidade do componente i nas condições de saturação**

P_i^S = **pressão de saturação**

Supondo-se que os volumes parciais molares dos componentes líquidos sejam função apenas da temperatura, $\bar{v}_i^L = v_i^L$, tem-se que

$$f_i^{ol} = \psi_i^S . \exp\left(\frac{P - P_i^S}{RT} . v_i^L\right) \tag{4.35}$$

Todo composto líquido tem sua própria curva específica de pressão de vapor, dependendo do balanço entre a fração de moléculas com energia suficiente para passar do líquido para o gás e do número de moléculas do gás que retornam ao líquido. Todos os compostos têm uma relação, que apesar de particular, tem a mesma forma exponencial.

A equação em questão é a equação de Antoine, a qual relaciona a pressão de vapor de um líquido saturado a uma determinada temperatura.

$$\ln(P) = A - \frac{B}{T + C} + DT + E \ln(T) \tag{4.36}$$

sendo que para a maior parte dos compostos e a uma pressão não muito elevada (cerca de 1000bar), os dois primeiros termos dão um ótimo ajuste. É esta a equação a ser usada para determinar a pressão vapor.

Para determinação do volume v_i^L, utilizou-se a equação de Rackett modificada por Spencer e Danner.

$$v = \frac{RT_c}{P_c}.\tau_a^{\tau} \tag{4.37}$$

com,

$$\tau = 1 + \left(1 - \frac{1}{T_c}\right)^{\frac{2}{7}}, \text{ para } \frac{T}{T_c} \leq 0.75 \tag{4.38}$$

$$\tau = 1.60 + 0.00693026.\left(T/T_c - 0.655\right)^{1}, \text{ para } T/T_c \geq 0.75$$

Os parâmetros necessários para a determinação das equações acima encontram-se em PRAUSNITZ et al.(1980).

4.2.4 CÁLCULO DA CONSTANTE DE EQUILÍBRIO LÍQUIDO-VAPOR

As equações requeridas para calcular o equilíbrio líquido-vapor em sistemas múlti componentes são, em princípio, as mesmas requeridas para sistemas binários. Em um sistema contendo NC componentes, deve-se resolver NC equações simultaneamente. É necessária a pressão de vapor de cada componente, quando líquido puro, à temperatura de interesse. Se todas as pressões de vapor são baixas, a pressão total é baixa também, de forma que se pode considerar o coeficiente de fugacidade como sendo unitário. Coeficientes de atividade são dados conforme a definição no item 4.2.2.

Da definição de equilíbrio termodinâmico entre um vapor e um líquido, tem-se que em cada estágio, **j**:

$$y_{i,j} = K_{i,j} \cdot \left(T_j, P_j, \frac{M_{i,j}}{\sum_i M_{i,j}} \right) \cdot \frac{M_{i,j}}{\sum_i M_{i,j}} \tag{4.39}$$

sendo que M_j é o acúmulo de líquido em cada estágio e a relação $\frac{M_{i,j}}{\sum_i M_{i,j}}$ é a fração molar do componente no líquido, x_i, em cada estágio.

Ou de outra forma, o coeficiente de distribuição ou constante de equilíbrio é dada por:

$$K_i = \left(\frac{y_i}{x_i} \right)_{\text{equilíbrio}}, \; i = 1,2, \ldots, NC. \tag{4.40}$$

Utilizando-se as equações do coeficiente de fugacidade, e o coeficiente de atividade, tem-se:

$$K_i = \left(\frac{y_i}{x_i} \right) = \left(\frac{\gamma_i . f_i^{ol}}{P \psi_i} \right)_{\text{equilíbrio}}, \; i = 1,2, \ldots, NC \tag{4.41}$$

GIORDANO et al.(1984), através da integração da equação da coexistência, estudaram a consistência termodinâmica de dados experimentais do equilíbrio líquido-vapor, para diferentes sistemas. Os dados considerados consistentes foram utilizados no ajuste de correlações para coeficientes de atividades através do método da máxima verossimilhança. Os modelos testados foram: UNIQUAC, NRTL, (2 e 3 parâmetros), Wilson, (2 e 3 parâmetros), van Laar, Margules, (2 e 3 parâmetros), Redlich-Kister (2,3,4 e 5 parâmetros).

As equações UNIQUAC, van Laar e Ridlich-Kister (4 parâmetros) foram as que melhor representaram os resultados experimentais.

4.2.5 CÁLCULO DAS ENTALPIAS DAS FASES LÍQUIDO E VAPOR

Para a resolução do balanço de energia da coluna é necessário o conhecimento das entalpias da fase líquida e da fase vapor. Neste item, a entalpia da fase líquida será representada por '**h**' e a entalpia da fase vapor por '**H**'.

ENTALPIA DA FASE VAPOR

Segundo PRAUSNITZ(1980), a entalpia da fase vapor é obtida em duas etapas. Primeiro, a partir das capacidades caloríficas dos componentes puros e depois a partir dos efeitos de mistura e da pressão.

$$\mathbf{H} = \mathbf{H}^{\mathbf{I}} + \Delta\mathbf{H} \tag{4.42}$$

Para uma mistura de componentes ideais, o segundo termo da última equação é nulo. O termo $\mathbf{H}^{\mathbf{I}}$ é a soma das entalpias de cada um dos componentes puros do vapor ideal e o termo $\Delta\mathbf{H}$ é o fator de correção molar da entalpia à temperatura T e pressão P, relativa ao vapor ideal à mesma temperatura e composição.

$$\mathbf{H}^{\mathbf{I}} = \sum \mathbf{y}_i . \mathbf{H}_i^{\mathbf{I}} \tag{4.43}$$

Para se determinar a entalpia do vapor ideal de cada um dos componentes, tem-se a seguinte relação:

$$\mathbf{H_i^I} = \int_{T_0}^{T} \mathbf{Cp_i^0 dT} \tag{4.44}$$

A capacidade calorífica de um vapor ideal é uma função monotônica da temperatura, aqui expressa por:

$$\mathbf{Cp_i^0} = \mathbf{D_{1i}} + \frac{\mathbf{D_{2i}}}{\mathbf{T}} + \mathbf{D_{3i}}.\mathbf{T} + \mathbf{D_{4i}}.\ln \mathbf{T} \tag{4.45}$$

sendo que $\mathbf{Cp_i^0}$ é expressa em Joules/moles.K e a temperatura é dada em K. Segundo PRAUSNITZ, a forma da equação (4.45) é a melhor forma disponível para representar a capacidade calorífica de vapores dentro de uma faixa de 200 a 600K. Os dados para a determinação dos valores de $\mathbf{Cp_i^0}$ estão no apêndice C da referência citada.

Substituindo-se a equação (4.45) na equação (4.44) e integrando, tem-se que:

$$\mathbf{H_i^I} = \mathbf{D_{1,i}}\left(\mathbf{T} - \mathbf{T_0}\right) + \mathbf{D_{2i}}.\ln\left(\frac{\mathbf{T}}{\mathbf{T_0}}\right) + \frac{\mathbf{D_{3i}}.\left(\mathbf{T^2} - \mathbf{T_0^2}\right)}{\mathbf{2}} +$$

$$\mathbf{D_{4i}}.\left(\mathbf{T}.\ln \mathbf{T} - \mathbf{T_0} - \mathbf{T_0}.\ln \mathbf{T_0} + \mathbf{T_0} - \mathbf{T}\right) \tag{4.46}$$

Da equação (4.46) na equação (4.43), tem-se que o termo referente ao vapor ideal fica:

$$\mathbf{H^I} = \sum_{i=1}^{NC} \mathbf{y_i}\left[\mathbf{D_{1i}}\left(\mathbf{T} - \mathbf{T_0}\right) + \mathbf{D_{2i}}.\ln\left(\frac{\mathbf{T}}{\mathbf{T_0}}\right) + \frac{\mathbf{D_{3i}}.\left(\mathbf{T^2} - \mathbf{T_0^2}\right)}{\mathbf{2}}\right] +$$

$$\sum_{i=1}^{NC} \mathbf{y_i}\left[\mathbf{D_{4i}}.\left(\mathbf{T}.\ln \mathbf{T} - \mathbf{T_0} - \mathbf{T_0}.\ln \mathbf{T_0} + \mathbf{T_0} - \mathbf{T}\right)\right] \tag{4.47}$$

Quando se tem uma mistura de vapores reais (não ideais) é necessária a utilização do termo de correção ΔH. Este fator é calculado como:

$$\Delta H = \int_0^P \left[v - T\left(\frac{\partial v}{\partial T}\right)_{P,y} \right] dP \qquad (4.48)$$

onde **v** é o volume molar da fase vapor.

Para se calcular a integral na equação (4.48), considerou-se a mistura como sendo não associada e utilizou-se a equação virial, truncada após o segundo termo. Desta forma, tem-se:

$$\Delta H = J.P.\sum_{i=1}^{NC}\sum_{j=1}^{NC} y_i.y_j.\left[B_{i,j} - T.\frac{dB_{i,j}}{dT} \right] \qquad (4.49)$$

onde **J** é um fator de conversão (0.0988 joules/cm^3-bar).

O fator **B** é calculado pelo método de TSONOPOULOS(1974), já citado anteriormente.

Em misturas de vapor associadas, onde ocorre forte dimerização, a determinação do $\Delta H_{assoc.}$ é feita através da utilização da teoria química.

ENTALPIA DA FASE LÍQUIDA

A entalpia de uma mistura líquida é dada por:

$$h = h^I + \sum x_i \overline{\Delta h_i} \qquad (4.50)$$

onde $\mathbf{h}^{I}$ é dado pela mesma expressão de $\mathbf{H}^{I}$, equação (4.47), bastando trocar 'x' pelo 'y'. O termo $\overline{\Delta \mathbf{h}_i}$ é a entalpia parcial molar do componente i, à temperatura **T**, pressão **P**, e composição **x**, relativa ao vapor puro, ideal, e à mesma temperatura.

A entalpia parcial molar para todo componente é encontrada a partir de uma forma da equação de Gibbs-Helmholtz. O detalhamento dos passos para se chegar à equação final do cálculo da entalpia da fase líquida pode ser acompanhada no Capítulo 5 de PRAUSNITZ(1980).

De modo simplificado, tem-se que se considerando a equação UNIQUAC, com todos os parâmetros $\mathbf{a}_{i,j}$ e $\mathbf{a}_{j,i}$ como sendo independentes da temperatura, obtém-se, a forma final da equação da entalpia da fase líquida:

$$\mathbf{h} = \mathbf{h}^{I} + \mathbf{R}\sum_{i=1}^{NC}\left[\frac{\mathbf{q}_i^{'}.\mathbf{x}_i}{\sum_{j=1}^{NC}\theta^{'}.\tau_{j,i}}.\left(\sum_{j=1}^{NC}\theta_j^{'}.\tau_{j,i}.\mathbf{a}_{j,i}\right)\right] - \mathbf{R}.\sum_{i=1}^{NC}\mathbf{x}_i.\left(-\mathbf{C}_{2i} + \mathbf{C}_{3i}.\mathbf{T}^2 + \mathbf{C}_{4i}.\mathbf{T} + 2\mathbf{C}_{5i}.\mathbf{T}^3\right) \tag{4.51}$$

sendo o segundo termo da equação (4.51), o termo referente à entalpia em excesso. O termo referente à entalpia de mistura foi desconsiderada devido à pouca contribuição para a entalpia total do líquido e à dificuldade em se determiná-la devido à falta de dados disponíveis para tanto.

4.2.6 EFICIÊNCIA DO PRATO

O vapor ao passar de um estágio a outro dentro da coluna de destilação, e entrar em contato com o líquido no prato superior realiza trocas de substâncias com este último de modo que os componentes mais voláteis tendem a 'acompanhar' o vapor e os menos voláteis o líquido. Caso o tempo de permanência de contato entre as fases fosse tal que

houvesse condições de se promover o equilíbrio termodinâmico entre ambas, dentro do prato, a fase vapor adquiriria uma concentração do equilíbrio para cada um dos componentes, dentro de cada prato, que será representada pelo símbolo $\mathbf{y}_i^*$. No entanto, isto raramente ocorre tendendo sempre, o vapor, a adquirir, ao sair do prato, uma composição menor que a máxima, representada por $\mathbf{y}_i$, dentro de cada prato.

Seguindo este raciocínio, pode-se definir um termo que mede a eficiência de troca de substâncias entre as fases líquida e vapor. Pode-se determinar a razão da variação real da concentração de determinada substância em relação ao que se esperaria em teoria. Desta forma tem-se uma relação entre a variação da composição do vapor de um componente i entre a entrada e a saída de um prato **j** da coluna de destilação em relação à variação da concentração do mesmo componente, no mesmo prato, caso o equilíbrio fosse atingido. Para um determinado ponto da coluna, tem-se que o índice **j** indica o prato ou estágio sobre o qual está sendo calculado qualquer propriedade e o índice **j-1**, indica o prato ou estágio anterior; pode-se a partir do exposto em HOLLAND(1963), definir um coeficiente de eficiência, tal que:

$$\varepsilon_{i,j}^{v} = \frac{\mathbf{y}_{i,j} - \mathbf{y}_{i,j-1}}{\mathbf{y}_{i,j}^{*} - \mathbf{y}_{i,j-1}} \tag{4.52}$$

o índice **v** se refere à fase vapor, contudo, a eficiência poderia, de forma análoga, ter sido definida para a fase líquida.

4.2.6.1 MODELOS DE EFICIÊNCIA DO PRATO

Pode-se falar em pelo menos três tipos ou três aspectos diferentes quando se trata de eficiência de troca de substâncias entre as fases presentes em um estágio de uma coluna de destilação. O primeiro aspecto diz respeito à distribuição da concentração do líquido que o vapor que borbulha em um estágio encontra, conforme a posição de entrada do vapor em

relação ao vertedouro de entrada e ao de saída do líquido. A rigor, a concentração do líquido não é homogênea fazendo com que fosse necessária a definição de um fator de eficiência local.

Do ponto de vista do estágio em si, tem-se que o vapor que sai do prato e atinge o prato superior àquele considerado, possui uma composição resultante da mistura de várias porções de vapor emergentes de diferentes pontos e, com composições diferentes, portanto, apresentando uma composição média. Se fossem utilizados esses valores de concentração na fórmula de Murphree, ter-se-ia a eficiência global do prato, de forma que a equação (4.52), representaria esta eficiência global com valores de concentrações médios. Nesta eficiência, o valor de $\mathbf{y}^{*}_{i,j}$ é a composição que teria o vapor se estivesse em equilíbrio com o líquido que deixa o prato no vertedouro, $\mathbf{x}_{i,j}$.

Finalmente, do ponto de vista da coluna, o terceiro tipo de eficiência, também global, da coluna é definido pela relação entre o número de estágios ideais ou teóricos que seriam necessários para se processar a separação e o número de estágios reais para se obter a mesma separação.

O valor assumido para a eficiência, neste trabalho, será considerado constante no tempo e igual para todos os componentes da mistura sendo implementado na modelagem dinâmica da coluna através da eficiência global do prato, ou seja, o modelo de eficiência caracterizado a partir do ponto de vista do estágio.

Introduzindo-se o termo de eficiência global na relação de equilíbrio líquido-vapor para um prato real, dada na equação (4.39), tem-se que aquela equação pode ser reescrita de forma modificada, como:

$$\mathbf{y}_{i,j} = \varepsilon.\mathbf{K}_{i,j}\left(\mathbf{T}_j, \mathbf{P}_j, \frac{\mathbf{M}_{i,j}}{\sum_i \mathbf{M}_{i,j}}\right).\frac{\mathbf{M}_{i,j}}{\sum_i \mathbf{M}_{i,j}} \tag{4.53}$$

ou explicitando-se o termo da eficiência:

$$\varepsilon = \frac{\mathbf{y}_{i,j} - \mathbf{y}_{i,j-1}}{\mathbf{y}^{*}_{i,j} - \mathbf{y}_{i,j-1}} = \frac{\mathbf{y}_{i,j} - \mathbf{y}_{i,j-1}}{\mathbf{K}_{i,j}.\mathbf{x}_{i,j} - \mathbf{y}_{i,j-1}} \tag{4.54}$$

Uma segunda modificação deve ser feita no modelo para a consideração da eficiência. É sobre a equação do ponto de bolha que continua válida, mas somente nas condições da interface, onde as fases estão em equilíbrio, PRAUSNITZ(1980). Deste modo a equação se transforma em:

$$\sum_{i} \mathbf{K}_{i,j} \cdot \mathbf{x}_{i,j} - \mathbf{1} = \mathbf{0} . \tag{4.55}$$

4.3 IMPLICAÇÕES HIDRÁULICA E FLUIDODINÂMICA DOS PRATOS

Apesar de se ter conhecimento da importância do estudo da fluidodinâmica dos dispositivos de contato (pratos ou bandeja) em colunas de destilação, neste trabalho foi desenvolvida uma modelagem dinâmica que não levou em conta estes aspectos.

Existem inúmeras correlações para se determinar o comportamento fluidodinâmico na literatura, contudo optou-se por uma simplificação devido ao fato de não ser a simulação tão rigorosa uma meta a ser perseguida neste trabalho. Aqui, conforme já dito anteriormente, a simulação dinâmica é apenas uma ferramenta para disponibilizar dados do comportamento de uma parte de um sistema de processos químicos a fim de se testar a viabilidade do procedimento proposto para detecção/diagnóstico das falhas.

Assim, a rigor, tanto para os pratos quanto para o refervedor e o condensador, seria necessário utilizar correlações as quais levassem em conta as vazões molares de líquido e vapor em cada estágio bem como os dados referentes à geometria dos mesmos. De modo simplificado, usou-se neste trabalho as seguintes correlações:

$$\mathbf{M}_1 = \mathbf{V}_C \cdot \rho_1 \tag{4.56}$$

$$\mathbf{M_{NS}} = \mathbf{V_R} \cdot \rho_{NS} \tag{4.57}$$

$$\rho_i = \mathbf{a_i} + \mathbf{b_i T} \tag{4.58}$$

para o cálculo dos acúmulos molares no condensador e no refervedor.

Para relacionar o acúmulo molar de líquido nos pratos à vazão de líquido foi utilizada a equação de Francis, válida para pratos com borbulhador, SMITH(1963).

$$\mathbf{L_j} = \left[\left(\frac{\mathbf{M_j}}{\mathbf{A_p} \cdot \rho_j} - \mathbf{h_v}\right) \cdot \frac{\mathbf{1}}{\mathbf{e}}\right]^{1.5} \cdot \rho_j \cdot \mathbf{L_w}, \; j = 2, \text{NS-1} \tag{4.59}$$

Assumindo-se que a massa específica e a razão de refluxo sejam dadas por:

$$\frac{\mathbf{1}}{\rho} = \sum_{i=1}^{NC} \frac{\mathbf{1}}{\rho_i} \tag{4.60}$$

e

$$\mathbf{RR} = \frac{\mathbf{L_1}}{\mathbf{D}} \tag{4.61}$$

pode-se calcular a vazão de líquido no estágio 1, $\mathbf{L_1}$, a partir das equações (4.6), (4.10), (4.57), (4.60) e (4.61):

$$\mathbf{L_1} = \frac{\left[\mathbf{1} + \rho_1 \cdot \sum_{i=1}^{NC} \frac{\mathbf{1}}{\rho_{i,1}} \cdot \left(\mathbf{y_{i,1}} - \mathbf{x_{i,1}}\right)\right]}{\mathbf{1} + \mathbf{1}/\mathbf{RR}} \cdot \mathbf{V_1} \tag{4.62}$$

Das equações (4.4) e (4.57), pode-se determinar o produto de fundo:

$$\mathbf{R = L_{NS-1} - V_{NS-1} - V_R \cdot \frac{d\rho_{NS}}{dt}} \tag{4.63}$$

Para determinação do vapor que deixa o refervedor pode-se rearranjar as equações (4.44) e (4.43). Integra-se a equação (4.44) e substitui-se na equação (4.43), derivando esta última em função da temperatura e da composição da fase líquida, a fim de se encontrar expressões analíticas para a determinação da variação da entalpia com a temperatura e com a composição da fase líquida.

Da equação (4.44), na forma integrada, em (4.43), para a fase líquida, tem-se

$$\mathbf{H_i^I = Cp_i^0 .(T - T_0)} \tag{4.44a}$$

$$\mathbf{H^I = \sum x_i .Cp_i^0 (T - T_0)} \tag{4.64}$$

derivando-se a última equação com relação à composição e à temperatura, tem-se:

$$\mathbf{\frac{\partial H^I}{\partial T} = \sum_{i=1}^{NC} x_i .Cp_i^0} \text{ , para } \mathbf{Cp_i^0 = cons\,tan\,te} \tag{4.65}$$

e

$$\mathbf{\frac{\partial H^I}{\partial x_i} = (T - T_0).\sum_{i=1}^{NC} Cp_i^0} \tag{4.66}$$

A equação descrita em CHO e JOSEPH(1983b)[21] para o cálculo analítico da equação do balanço de energia tem a seguinte forma:

---

[21] O desenvolvimento desta equação está apresentada no trabalho referenciado.

$$\frac{dh}{dt} = -\frac{\partial h}{\partial T} \cdot \frac{\sum_{i=1}^{NC} K_i \cdot \frac{dx_i}{dt}}{\sum_{i=1}^{NC} x_i \cdot \frac{dK_i}{dT}} + \sum_{i=1}^{NC} \frac{\partial h}{\partial x_i} \cdot \frac{dx_i}{dt} \tag{4.67}$$

Substituindo-se a equação (4.67) em (4.14), tem-se que

$$\frac{L_{NS-1} \cdot (h_{NS-1} - h_{NS}) + V_{NS-1} \cdot (h_{NS} - H_{NS-1}) + Q_R}{M_{NS}} =$$

$$-\frac{\partial h_{NS}}{\partial T} \cdot \frac{\sum_{i=1}^{NC} K_{i,NS} \cdot \frac{dx_{i,NS}}{dt}}{\sum_{i=1}^{NC} x_{i,NS} \cdot \frac{dK_{i,NS}}{dT}} + \sum_{i=1}^{NC} \frac{\partial h_{NS}}{\partial x_{i,NS}} \cdot \frac{dx_{i,NS}}{dt} \tag{4.68}$$

finalmente, substituindo-se a equação (4.9) em (4.68), tem-se:

$$V_{NS-1} = \frac{L_{NS-1} \cdot \left[h_{NS} - h_{NS-1} + \sum_{i=1}^{NC} \alpha_{i,NS} \cdot (x_{i,NS-1} - x_{i,NS}) \right] - Q_R}{h_{NS} - H_{NS-1} + \sum_{i=1}^{NC} \alpha_{i,NS} \cdot (y_{i,NS-1} - y_{NS})} \tag{4.69}$$

que é a expressão para determinação do vapor que sai do refervedor, onde:

$$\alpha_{i,j} = \frac{-\frac{\partial h_j}{\partial T} \cdot K_{i,j}}{\sum_{k=1}^{NC} x_{k,j} \cdot \frac{dK_{k,j}}{dT}} + \frac{\partial h_j}{\partial x_{i,j}} \tag{4.70}$$

Para determinação do vapor que sai de cada prato tem-se, das equações (4.8), (4.13), (4.67) e (4.70), que:

$$V_{j-1} = \frac{\left[\sum_{i=1}^{NC} \alpha_{i,j} \cdot (A - B) \right]}{[C]} \tag{4.71}$$

onde,

$$\mathbf{A} = \left(\mathbf{L}_{j-1-}.\mathbf{x}_{i,j-1} + \mathbf{V}_j.\mathbf{y}_{i,j} + \mathbf{F}_j.\mathbf{z}_{i,j} - \mathbf{x}_{i,j}\left(\mathbf{L}_{j-1} + \mathbf{V}_j + \mathbf{F}_j\right)\right)$$

$$\mathbf{B} = \left(\mathbf{L}_{j-1}.\mathbf{h}_{j-1} - \mathbf{V}_j\mathbf{H}_j - \mathbf{F}_j.\mathbf{hf}_j - \mathbf{h}_j.\left(\mathbf{L}_{j-1} + \mathbf{V}_j + \mathbf{F}_j\right)\right)$$

$$\mathbf{C} = \left(\mathbf{h}_j - \mathbf{H}_{j-1} - \sum_{i=1}^{NC} \alpha_{i,j}.\left(\mathbf{x}_{i,j} - \mathbf{y}_{i,j-1}\right)\right), \text{ para } j = 2, \ldots, NS\text{-}1$$

Para o cálculo do vapor que entra no vaporizador, tem-se que substituindo-se a equação (4.62) em (4.71). tem-se

$$\mathbf{V}_1 = \left[\sum_{i=1}^{NC} \alpha_{i,2}.(\mathbf{A}) - \mathbf{B}\right] / [\mathbf{C} - \mathbf{D}.\mathbf{E}] \qquad (4.72)$$

sendo

$$\mathbf{A} = \mathbf{V}_2.\mathbf{y}_{i,2} + \mathbf{F}_2.\mathbf{z}_{i,2} - \mathbf{x}_{i,2}.\left(\mathbf{V}_2 + \mathbf{F}_2\right)$$

$$\mathbf{B} = \left[\mathbf{V}_2 + \mathbf{F}_2.\mathbf{hf}_2 - \mathbf{h}_2.\left(\mathbf{V}_2 + \mathbf{F}_2\right)\right]$$

$$\mathbf{C} = \left[\mathbf{h}_2 - \mathbf{H}_1 - \sum_{i=1}^{NC} \alpha_{i,2}.\left(\mathbf{x}_{i,2} - \mathbf{y}_{i,2}\right)\right]$$

$$\mathbf{D} = \frac{\left[1 + \rho_1.\sum_{i=1}^{NC} \frac{1}{\rho_{i,1}}.\left(\mathbf{y}_{i,1} - \mathbf{x}_{i,1}\right)\right]}{1 + \frac{1}{\mathbf{RR}}}$$

$$\mathbf{E} = \sum \alpha_{i,2}.\left(\mathbf{x}_{i,1} - \mathbf{x}_{i,2}\right) - \mathbf{h}_1 + \mathbf{h}_2$$

4.4 RESOLUÇÃO DAS EQUAÇÕES DIFERENCIAIS

O problema de determinação de um método adequado para integração das equações diferenciais na simulação de colunas de destilação está bem desenvolvido na literatura. MAH et al.(1962) discutiram a estabilidade e os erros característicos de um método bastante utilizado quando aplicado a sistemas de destilação contínua multi-componente.

Basicamente, as equações eram linearizadas e os parâmetros destas equações assumidos constantes através de cada incremento de tempo. DISTEFANO et al.(1968) estudaram uma série de métodos numéricos objetivando a simulação da destilação em batelada concluindo que a dificuldade encontrada na utilização dos algoritmos (todos eram explícitos), referia-se à estabilidade numérica associada a estes algoritmos. Para manter a estabilidade foram necessários pequenos passos de integração, o que aumentava muito o tempo de computação.

Na resolução das equações diferenciais deste trabalho, foram usados dois métodos diferentes de integração das mesmas. No primeiro caso utilizou-se o método de Runge-Kutta Gill de quarta ordem, com passo fixo. No segundo caso foi utilizada a rotina de cálculo DASSL, a qual tem a vantagem de resolver simultaneamente equações diferenciais e algébricas. Não foi notada, para o sistema em estudo vantagem de utilização do procedimento DASSL, de forma que no código final do simulador foi deixado apenas o Runge-Kutta Gill, por causa da praticidade de manuseio das rotinas de simulação. O DASSL é um código aberto e que está disponível na internet mas seu entendimento e aplicação requer um estudo detalhado do sistema de equações que se quer resolver. Para um estudo a respeito deste último, pode ser consultado os trabalhos de GEAR(1971) ou o de BRENAN et al.(1990).

4.5 RESULTADOS DA SIMULAÇÃO DA COLUNA DE DESTILAÇÃO UTILIZANDO O SISTEMA METANOL-ÁGUA. COM E SEM CONTROLADOR

A simulação em estado estacionário foi feita utilizando-se o simulador ***HYSIM***, que forneceu dados para o início da simulação dinâmica, os quais estão resumidos abaixo a partir da tabela 4.1.

Colunas	01
N^0 de Estágios	30
N^0 de Alimentações	01
Estágio Alimentação	18
Produtos	02
Componentes	Metanol/Água

Tabela 4.1: Dados gerais do sistema simulado.

As unidades de cada um dos parâmetros utilizados aqui estão descritos no começo deste trabalho sob o título "Lista de Símbolos". Aqui são apresentados alguns deles para facilitar a compreensão imediata dos dados apresentados neste capítulo, visto que podem estar em unidades diferentes daquelas efetivamente utilizadas no programa de simulação, o qual continha fatores de conversão de unidades quando necessário.

A tabela 4.2 resume os referidos parâmetros.

Parâmetro	*Unidade*
Fluxo	Kgmol/h
Temperatura	^0C
Pressão	Kpa
Entalpia Específica	KJ/kgmol
Fluxo de Calor	KJ/kgmol-C

Tabela 4.2: Unidades de medida de alguns parâmetros utilizados para simulação pelo ***HYSIM***

Valores numéricos dos parâmetros utilizados na simulação dinâmica da coluna de destilação, referentes ao estado estacionário inicial.

Estágio	FC	Pressão	Temp	L	V
01	-4,1141E+04	1,20000E+02	6,89401E+01	5,76807E-01	0,00000E-00
02	0,00000E+00	1,20000E+02	6,90529E+01	5,75880E-01	1,17477E-00
03	0,00000E+00	1,20000E+02	6,91755E+01	5,74895E-01	1,17384E-00
04	0,00000E+00	1,20000E+02	6,93094E+01	5,73844E-01	1,17286E-00
05	0,00000E+00	1,20000E+02	6,94564E+01	5,72712E-01	1,17181E-00
06	0,00000E+00	1,20000E+02	6,96186E+01	5,71481E-01	1,17068E-00
07	0,00000E+00	1,20000E+02	6,67993E+01	5,70125E-01	1,16945E-00
08	0,00000E+00	1,20000E+02	7,00012E+01	5,68609E-01	1,16809E-00
09	0,00000E+00	1,20000E+02	7,02287E+01	5,66891E-01	1,16658E-00
10	0,00000E+00	1,20000E+02	7,04867E+01	5,64915E-01	1,164859E00
11	0,00000E+00	1,20000E+02	7,07816E+01	5,62619E-01	1,16288E-00

Continuação

Estágio	FC	Pressão	Temp	L	V
12	0,00000E+00	1,20000E+02	7,11198E+01	5,59925E-01	1,16058E-00
13	0,00000E+00	1,20000E+02	7,15115E+01	5,56749E-01	1,15471E-00
14	0,00000E+00	1,20000E+02	7,19672E+01	5,5301E-01	1,15098E-00
15	0,00000E+00	1,20000E+02	7,24997E+01	5,48686E-01	1,15098E-00
16	0,00000E+00	1,20000E+02	7,31227E+01	5,43784E-01	1,14665E-00
17	0,00000E+00	1,20000E+02	7,38493E+01	5,38491E-01	1,14175E-00
→18	0,00000E+00	1,20000E+02	7,46875E+01	1,53863E-00	1,13646E-00
19	0,00000E+00	1,20000E+02	7,46846E+01	1,53903E-00	1,13659E-00
20	0,00000E+00	1,20000E+02	7,46821E+01	1,53954E-00	1,13699E-00
21	0,00000E+00	1,20000E+02	7,46846E+01	1,54023E-00	1,13751E-00
22	0,00000E+00	1,20000E+02	7,46871E+01	1,54078E-00	1,13819E-00
23	0,00000E+00	1,20000E+02	7,4709E+01	1,54073E-00	1,13876E-00
24	0,00000E+00	1,20000E+02	7,47803E+01	1,53942E-00	1,13869E-00
25	0,00000E+00	1,20000E+02	7,49990E+01	1,53541E-00	1,13740E-00
26	0,00000E+00	1,20000E+02	7,56783E+01	1,52355E-00	1,13338E-00
27	0,00000E+00	1,20000E+02	7,77946E+01	1,49083E-00	1,12152E-00
28	0,00000E+00	1,20000E+02	8,39001E+01	1,43518E-00	1,08879E-00
29	0,00000E+00	1,20000E+02	9,48563E+01	1,40818E-00	1,03314E-00
30	4,15431E+04	1,20000E+02	1,02268E+02	4,02032E-01	1,00615E-00

Tabela 4.3: Dados em estado estacionário para a coluna de destilação simulada, onde, **FC** = Fluxo de Calor; **Temp** = temperatura; **V** = Vazão de vapor; **L** = Vazão de Líquido.

Estágio	Entalpia L	Entalpia V	x	y
01	-2,3380E+04	1,16352E+04	0,993298	0,996925
02	-2,3430E+04	1,16395E+04	0,985347	0,993298
03	-2,3483E+04	1,16442E+04	0,976740	0,989397
04	-2,3542E+04	1,16494E+04	0,967379	0,985182
05	-2,3605E+04	1,16551E+04	0,957143	0,9806,6
06	-2,3675E+04	1,16612E+04	0,945891	0,975611
07	--2,3752E+04	1,16612E+04	0,933445	0,970132
08	-2,3836E+04	1,16757E+04	0,919620	0,964088
09	-2,3931E+04	1,16841E+04	0,904153	0,957386
10	-2,4036E+04	1,16937E+04	0,886755	0,949914
11	-2,4157E+04	1,17045E+04	0,867076	0,941540
12	-2,4296E+04	1,17171E+04	0,844696	0,932109
13	-2,4456E+04	1,17317E+04	0,819127	0,921438
14	-2,4644E+04	1,17489E+04	0,789812	0,909321
15	-2,4865E+04	1,17695E+04	0,756168	0,895529
16	-5,5120E+04	1,17939E+04	0,717685	0,879829
17	-2,5409E+04	1,18227E+04	0,674152	0,862032
18	-2.5721E+04	1,18557E+04	0,626023	0,842077
19	-2,5724E+04	1,18559E+04	0,626122	0,842143
20	-2,5727E+04	1,18565E+04	0,626204	0,842202
21	-2,5729E+04	1,18573E+04	0,6258965	0,842086
22	-2,5732E+04	1,18580E+04	0,624661	0,841562
23	-2,5739E+04	1,18589E+04	0,620704	0,839901

Continuação

Estágio	Entalpia L	Entalpia V	x	y
24	-2,5764E+04	1,18614E+04	0,608569	0,834792
25	-2,5842E+04	1,18695E+04	0,608569	0,834792
26	-2,6077E+04	1,18949E+04	0,571902	0,819110
27	-6,6742E+04	1,19734E+04	0,467916	0,771528
28	-2,8008E+04	1,21873E+04	0,254541	0,635144
29	-2,8688E+04	1,24874E+04	0,074617	0,347746
30	-2,8662E+04	1,25901E+04	0,015022	0,984301

Tabela 4.4: Dados em estado estacionário para a coluna de destilação simulada; onde, **Entalpia L** = Entalpia do Líquido; **Entalpia V** = Entalpia do Vapor; **x** = Fração molar do Metanol (**1-x** = fração molar da água); **y**= fração molar do vapor de Metanol (i–y = fração molar da água).

Valores numéricos dos parâmetros de construção física da coluna de destilação.

Volume do Condensador	**$1{,}13*10^5$ cm^3**
Volume do Refervedor	**$2{,}60*10^5$ cm^3**
Área do Prato	**$1{,}02*10^4$ cm^2**
Quantidade de Calor Ref.	**$1{,}812*10^7$Cal/min**
Temperatura de Aliment.	**75,16 °C**
Vazão de Alimentação	**$1{,}667*10^3$ mol/min**
Razão de Refluxo	**0,96461**
Eficiência dos pratos	**0,85**

Continuação

Altura do weir	**5,0cm**
Fator de Francis	**9,34E-03**
ℓ do vertedouro	**92cm**
Composição de Alim.	**(met)=0,6:(água)=0,4**

Tabela 4.5: Parâmetros de construção e alimentação da coluna de destilação.

Kc	-446,87
τ_C	0,11085
τ_D	0,01733

Tabela 4.6: Parâmetros de ajuste do controlador PID.

4.5.1 RESULTADOS DA SIMULAÇÃO DINÂMICA

Tendo em vista mostrar que o modelo dinâmico desenvolvido para a coluna de destilação é coerente com o comportamento que se espera de um equipamento real, fez-se um estudo onde são analisadas as respostas do sistema a determinadas perturbações.

CASO 1: Mudanças no valor da razão de refluxo

CASO 2: Mudanças na temperatura de alimentação da coluna

CASO 3: Mudanças na taxa de fornecimento de calor na base da coluna

CASO 4: Mudanças na composição de alimentação da coluna

CASO 5: Mudanças na vazão de alimentação.

As variações aqui estudadas são do tipo degrau. Os gráficos mostram o comportamento de algumas variáveis quando se simula o comportamento dinâmico da coluna de destilação durante cerca de 300 min. O primeiro patamar em cada um dos gráficos se refere ao sistema em funcionamento em estado estacionário, nos primeiros trinta minutos a partir da entrada em funcionamento. O valor de cada perturbação está indicado em cada uma das curvas. Este procedimento objetiva simular um falso transiente, BIARDI et al. (1987), originado ou pelo algoritmo numérico ou por uma não igualdade entre os perfis estáticos gerados.

Conforme descrito no capítulo 2 deste trabalho, quanto à justificativa para o uso dos dados dinâmicos de funcionamento do sistema, são apresentados aqui o comportamento de cinco variáveis monitoradas as quais servem de fonte de dados para testar o procedimento de detecção e o diagnóstico de falhas no sistema.

Em cada um dos casos acima descritos, foi dada, a partir de certo momento, uma perturbação no sistema que estava funcionando em estado estacionário. Dada esta perturbação, anotou-se o comportamento dinâmico de cinco variáveis do processo; a temperatura do topo da coluna, a temperatura da base da coluna, a vazão de destilado, a vazão do produto de fundo e a fração molar do produto mais volátil, no topo, no caso, o metanol, para sete diferentes níveis de falhas.

Estão apresentadas nas páginas seguintes, os gráficos de algumas das simulações feitas. Os percentuais de modificações em cada uma das perturbações aqui mostradas foram escolhidas de forma arbitrária, sempre pensando-se em perturbações positivas e negativas.

A dinâmica própria do sistema, representada pela sua capacitância e pela capacidade de amortecimento de alguns desvios nos valores das variáveis do sistema, são responsáveis pela pequena variação numérica nos valores de cada uma das variáveis monitoradas nos instantes iniciais da falha22 e foi o que possibilitou, conforme dito no capítulo 2, detectar falhas com níveis não utilizados antes para o treinamento da rede neural. A diferença numérica pequena entre os valores faz com que as curvas se

---

[22] O tempo pode variar de parâmetro para parâmetro. Devem ser observadas a direção da perturbação se positiva ou negativa, pois quando não apresenta resposta inversa, com determinados tipos e níveis de perturbações, já a partir do início da mesma, a direção de modificação dos valores das variáveis vai para direções diferentes, impossibilitando, obviamente, a utilização de dados de uma tendência para detectar falhas ocorridas com a tendência contrária.

sobreponham para vários níveis de falhas, poluindo os gráficos (ver Figura 4.30) e nada acrescentando de informações, devido à escala utilizada. Desta forma, optou-se por representar apenas um nível de falha em cada figura.

CASO 1: Respostas do sistema a mudanças no valor da razão de refluxo

Está aqui representado o comportamento dinâmico produzido nas cinco variáveis monitoradas quando se introduziu uma perturbação de 10% na razão de refluxo da coluna. Em cada uma das próximas cinco figuras estão representadas as curvas de comportamento das variáveis indicadas em cada uma delas.

Conforme era de se esperar, uma modificação positiva na razão de refluxo causou uma variação positiva na temperatura do topo, uma variação negativa na temperatura do fundo, uma variação negativa na vazão do destilado (menor quantidade de líquido saindo), uma variação positiva (maior quantidade de produto) na vazão de fundo e uma variação positiva na fração molar do destilado, o produto mais volátil. Para o caso de perturbação negativa, vale a análise contrária à que foi apresentada.

Para o treinamento das redes neurais, na parcela referente à mudança na razão de refluxo foram simuladas situações em que ocorriam desvios de –50%, -20%, +10%, +30%, +50%. Para gerar valores de confirmação ou validação do procedimento de treinamento das redes neurais, foram simuladas mais duas situações, -30% e –10%.

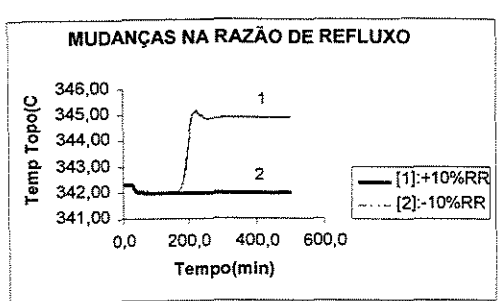


Figura 4.2: Comportamento da temperatura do topo da coluna face a mudanças na razão de refluxo.

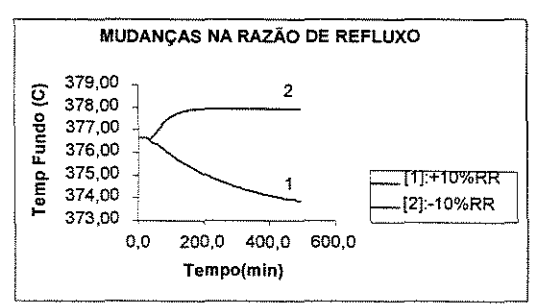


Figura 4.3 : Comportamento da temperatura do topo da coluna face a mudanças na razão de razão de refluxo.

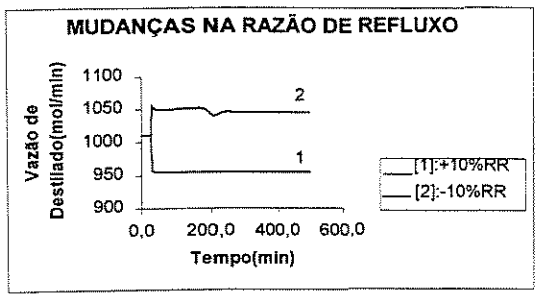


Figura 4.4 : Comportamento da vazão de produto no topo (destilado) da coluna face a mudanças na razão de refluxo.

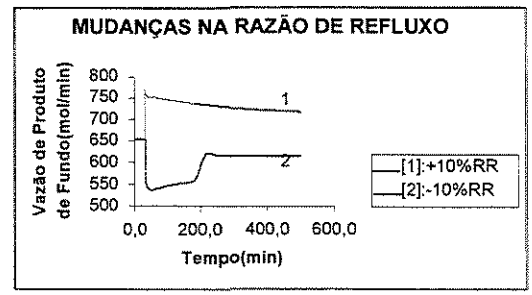


Figura 4.5: Comportamento da vazão de produto no fundo da coluna face a mudanças na razão de refluxo.

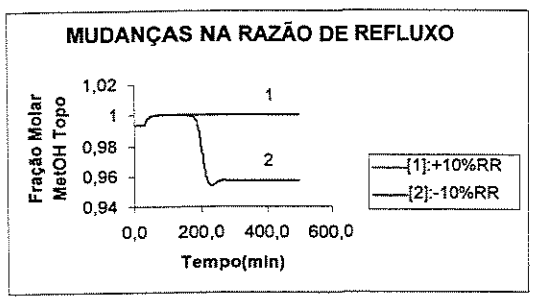


Figura 4.6 : Comportamento da fração molar de metanol no topo da coluna face a mudanças na razão de refluxo.

CASO 2: Respostas do sistema a mudanças na temperatura de alimentação da coluna

As Figuras de 4.7 a 4.11 representam o comportamento dinâmico do sistema frente a uma variação de -30% na temperatura[23] da corrente de alimentação do sistema.

Com exceção da temperatura do fundo, que apresentou uma resposta inversa[24], os demais parâmetros estão de acordo com o comportamento esperado. Corrente de entrada mais fria, diminui a temperatura e também a vazão de saída no topo da coluna,

[23] O programa de simulação permite qualquer combinação de composição vapor líquido para a alimentação.
[24] Tal fato não interfere em nada no mecanismo a ser estudado neste trabalho, pois o que importa é a tendência da variável analisada, e isto se verificou para todos os níveis estudados.

aumentando consequentemente a saída no fundo da mesma. Devido à menor quantidade de energia no sistema e à necessidade de mais energia para se volatilizar um componente mais pesado, sua concentração cai no topo da coluna. O item temperatura de alimentação por ser um dos mais sensíveis no funcionamento da coluna, foi o que se empregou as menores taxas de mudanças. Assim, para o treinamento das redes usou-se os seguintes percentuais de mudança: -30%, -15%, +10%, +20%, +30%. Para a validação dos resultados do treinamento, foram usadas dois níveis de perturbações, -20%, +15%.

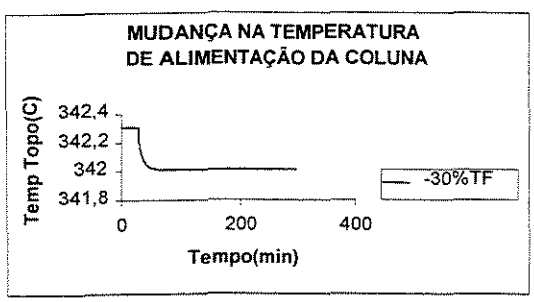


Figura 4.7 : Comportamento da temperatura do topo da coluna face a mudanças na temperatura de alimentação da coluna.

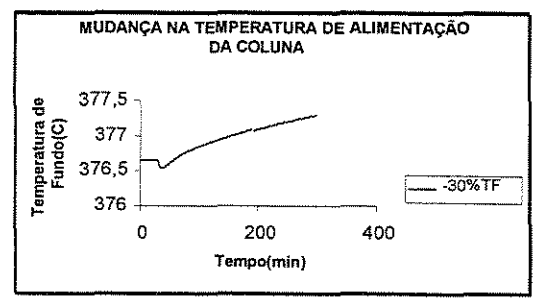


Figura 4.8: Comportamento da temperatura do fundo da coluna face a mudanças na mudança de alimentação da coluna.

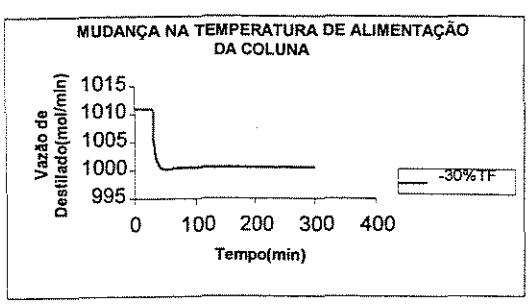


Figura 4.9: Comportamento da vazão do produto de topo (destilado) da coluna face a mudanças na temperatura de alimentação da coluna.

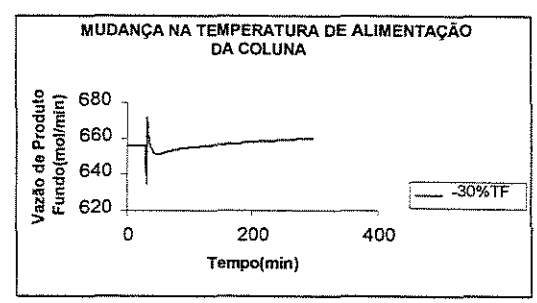


Figura 4.10: Comportamento da vazão de produto no fundo da coluna face a mudanças na temperatura de alimentação da coluna.

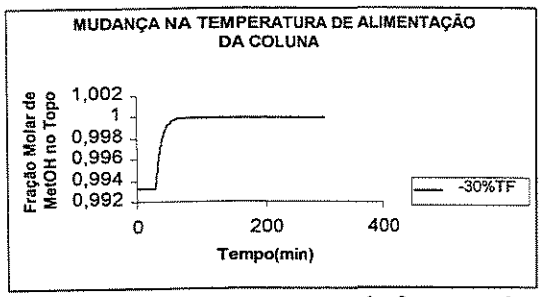


Figura 4.11 : Comportamento da fração molar de metanol no topo da coluna face a mudanças na temperatura de alimentação da coluna.

CASO 3: Respostas do sistema a mudanças na taxa de fornecimento de calor na base da coluna

As figuras de 4.12 a 4.16 mostram o comportamento dinâmico das variáveis monitoradas frente a uma variação positiva e uma variação negativa de 10% na taxa de fornecimento de calor ao refervedor.

O comportamento das variáveis estão mais uma vez dentro do esperado. Um aumento na taxa de fornecimento de calor na base da coluna, faz, obviamente, com que aumente as temperaturas do topo e da base da coluna, inversamente ao que ocorre se a taxa de calor é diminuída.

Com as vazões de destilado e de produto, dentro também do esperado, se se aumenta a vaporização do líquido dentro da coluna, como conseqüência da elevação da taxa de fornecimento de calor na base, mais destilado sairá no topo e menos produto na base. O contrário também se verifica. A quantidade percentual de metanol no topo devido à maior vaporização do produto menos volátil, com um aumento na taxa de calor no refervedor, diminui.

Para o treinamento das redes neurais, foram utilizadas simulações de aumento e diminuição na quantidade de calor fornecida ao refervedor cujos percentuais em relação ao estado normal são: -30%, -20%, -10%, +10%, +30%. Para a validação dos resultados do treinamento, foram simuladas situações com –25% e +20% de calor fornecido ao

refervedor. Aqui, como no item anterior, sobre a temperatura, é preciso se ter cuidado para que a coluna não seque.

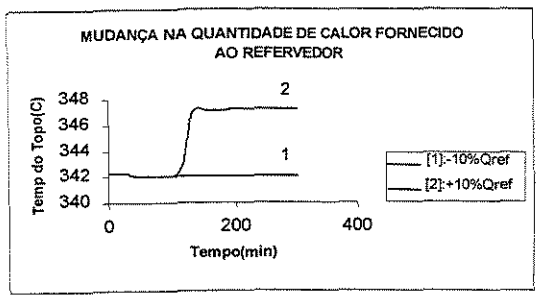


Figura 4.12 : Comportamento da temperatura do topo da coluna face a mudanças na quantidade de calor fornecido ao refervedor.

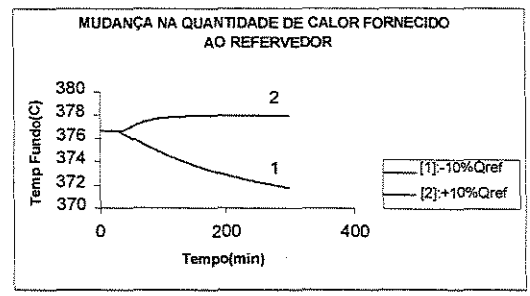


Figura 4.13 : Comportamento da temperatura do fundo da coluna face a mudanças na quantidade de calor fornecido ao refervedor.

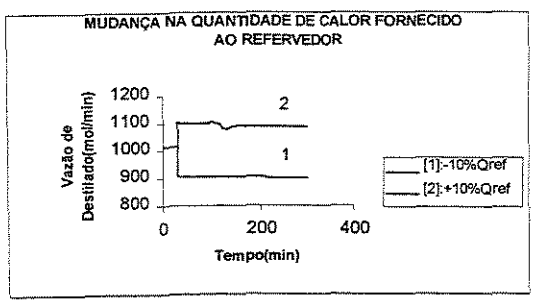


Figura 4.14 : Comportamento da vazão de produto no topo (destilado) da coluna face a mudanças na quantidade de calor fornecido ao refervedor.

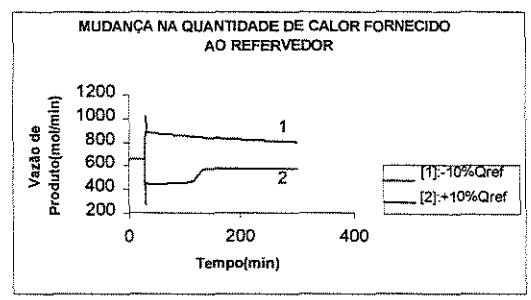


Figura 4.15 : Comportamento da vazão de produto no fundo da coluna face a mudanças na quantidade de calor fornecida ao refervedor.

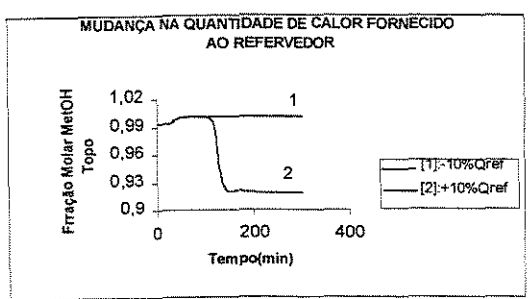


Figura 4.16: Comportamento da fração molar de metanol no topo da coluna face a mudanças na quantidade de calor fornecido ao refervedor.

CASO 4: Respostas do sistema a mudanças na composição de alimentação da coluna

A mudança positiva de 20% na composição de alimentação da coluna causou uma diminuição na temperatura do topo da mesma quando se aumentou a quantidade de água na mistura, fazendo-se subir consideravelmente quando se aumentou a quantidade de metanol na corrente de alimentação. O efeito na corrente de fundo teria, naturalmente, que ser o mesmo. O aumento na quantidade de componente menos volátil sem que se aumentasse a quantidade de calor fornecido ao refervedor, fez com que baixasse a temperatura nesta parte da coluna, também.

Quanto às vazões, de topo e de fundo, o efeito é o inverso. O aumento da composição de água faz com que menor quantidade de mistura consiga se volatilizar, diminuindo o destilado, e fazendo uma maior parte sair pelo fundo.

Proporcionalmente, quanto maior a quantidade de água na alimentação, maior a quantidade de metanol que consegue se volatilizar. Aumenta a pureza do produto de topo. No caso contrário, uma menor quantidade de água na alimentação, vaporizada com a mesma taxa de calor na base, faz com que caia a pureza do produto destilado.

Esta foi a variável à qual se aplicou as maiores taxas de mudanças, representando falhas mais graves. Para o treinamento das redes neurais, foram utilizadas mudanças de –60%, -20%, +5%, +20% e +60%. Para a validação do procedimento, foram usadas mudanças positivas e negativas de 40%. Os gráficos apresentam a simulação com variações de +20 e –60%, na variável manipulada.

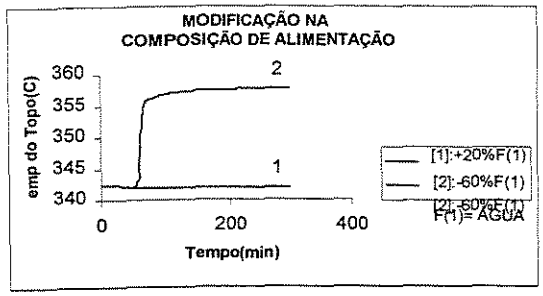


Figura 4.17: Comportamento da temperatura do topo da coluna face a mudanças na composição de alimentação da coluna.

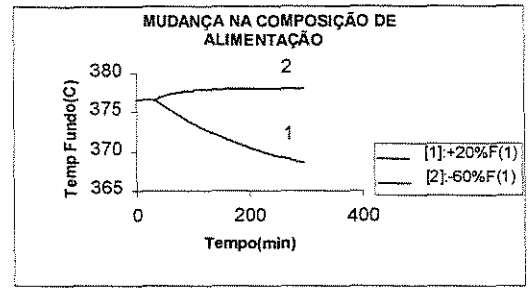


Figura 4.18 : Comportamento da temperatura do fundo da coluna face a mudanças na composição de alimentação.

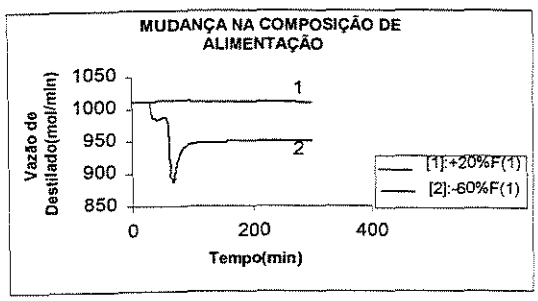


Figura 4.19 : Comportamento da vazão de produto no topo (destilado) da coluna face a mudanças na composição de alimentação da coluna.

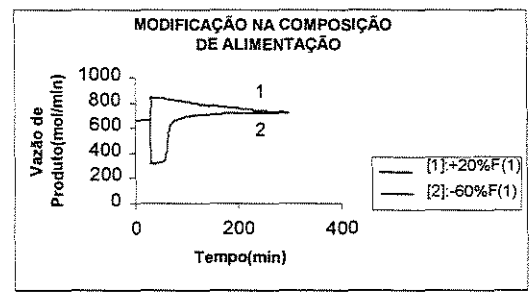


Figura 4.20 : Comportamento da vazão de produto no fundo da coluna face a mudanças na composição de alimentação.

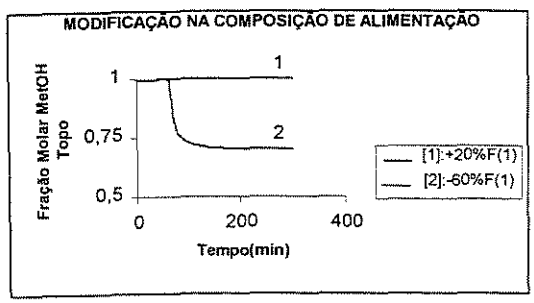


Figura 4.21: Comportamento da fração molar de metanol no topo da coluna face a mudanças na composição de alimentação da coluna.

CASO 5: Respostas do sistema a mudanças na vazão de alimentação.

A modificação na vazão de alimentação, mantendo-se as demais condições de proporcionalidade entre os componentes constantes, faz com que uma diminuição de 20% na vazão de entrada aumente a temperatura no topo da coluna, uma vez que com a mesma quantidade de calor, uma quantidade menor de alimentação vai estar mais aquecida em todos os pontos da coluna. O efeito inverso se verifica com o aumento dos mesmos 20%.

Com a temperatura do fundo tem-se o mesmo tipo de fenômeno e a explicação pode ser transposta para este caso.

A quantidade de destilado diminui em qualquer das situações. No caso da diminuição da quantidade de entrada, o efeito é menos acentuado apesar de uma maior quantidade poder ser volatilizada. A queda na quantidade de entrada não é compensada por maior volatilização.

No caso de aumento da vazão de entrada, sem aumento da energia no refervedor, necessária para que se aumentasse a taxa de volatilização na mesma proporção que no caso normal de operação original, ocorre diminuição da quantidade de destilado.

Para o produto da base, a diminuição da quantidade de entrada também faz diminuir à saída, o aumento faz aumentar a quantidade à saída.

Uma quantidade de mistura maior faz com que a energia seja menos que o suficiente para volatilizar o mais pesado. Aumenta a recuperação do metanol. No caso contrário, menor quantidade, é igual a mais volatilização, menor pureza do destilado.

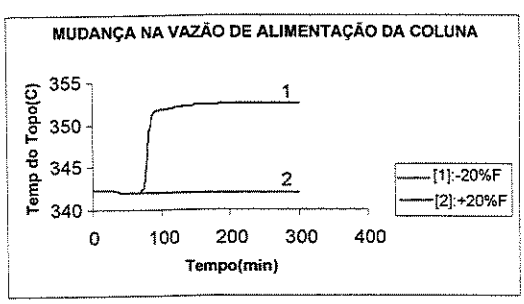


Figura 4.22 : Comportamento da temperatura do topo da coluna face a mudanças na alimentação da coluna.

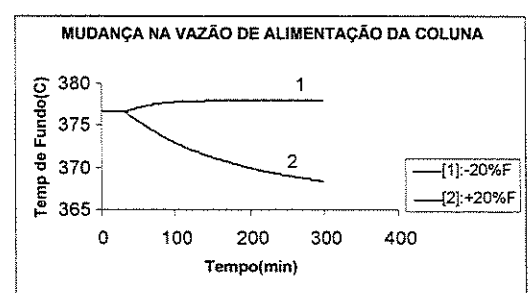


Figura 4.23 : Comportamento da temperatura do fundo da coluna face a mudanças na vazão de alimentação da coluna.

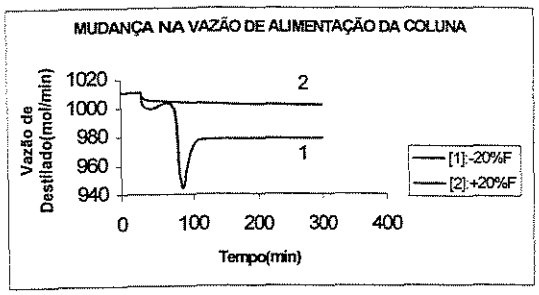


Figura 4.24 : Comportamento da vazão de produto no topo (destilado) da coluna face a mudanças na vazão de alimentação da coluna.

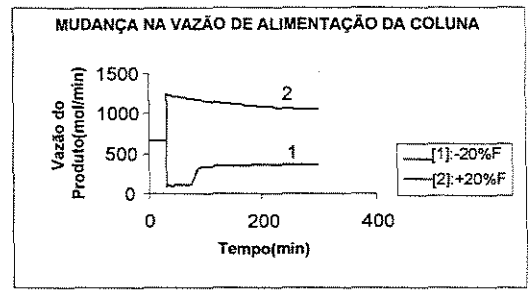


Figura 4.25 : Comportamento da vazão de produto no fundo da coluna face a mudanças na vazão de alimentação da coluna.

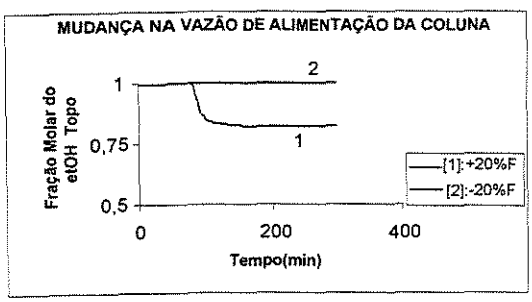


Figura 4.26: Comportamento da fração molar de metanol no topo da coluna face a mudanças na vazão de alimentação da coluna.

4.5.2 CURVAS DOS PARÂMETROS CONTROLADOS VERSUS TEMPO

As figuras de 4.27 a 4.29 mostram o comportamento da temperatura do topo da coluna quando esta é escolhida como variável controlada.

Estão representadas nas três figuras o comportamento dinâmico do sistema quando se muda a temperatura do fluido de alimentação, quando se modifica a taxa de fornecimento de calor ao refervedor e quando se muda a taxa de alimentação da coluna.

Nas três figuras, vê-se que apesar de o controlador ser capaz de reestabelecer o valor desejado para a variável controlada, o tempo de resposta é muito grande. Aqui pode-se ver um exemplo em que o procedimento de detecção e diagnóstico de falhas é útil. É

possível descobrir rapidamente a falha e diagnosticar a sua fonte, corrigindo-se ainda que 'manualmente' o problema na origem sem esperar que o sistema se estabilize automaticamente. Algumas vezes, em que ocorre desvios permanentes, este tipo de situação não ocorre, e o procedimento de detecção/diagnóstico se mostra imprescindível.

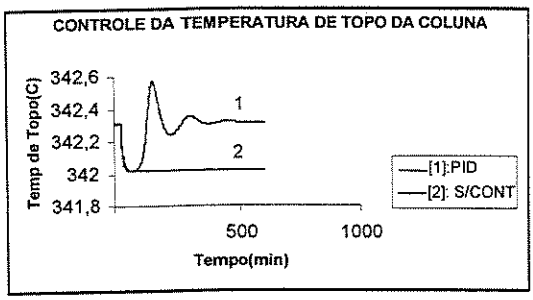


Figura 4.27: Comportamento da temperatura de topo da coluna de destilação quando se muda a temperatura do fluido que vai para a alimentação da coluna.

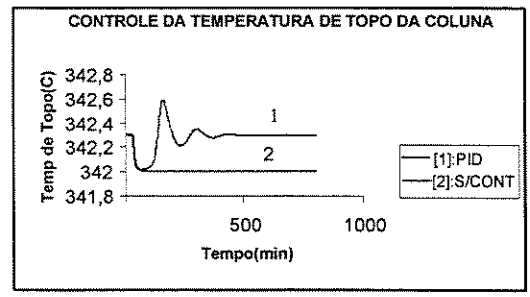


Figura 4.28: Comportamento da temperatura de topo da coluna de destilação quando se modifica a taxa de fornecimento de calor ao refervedor.

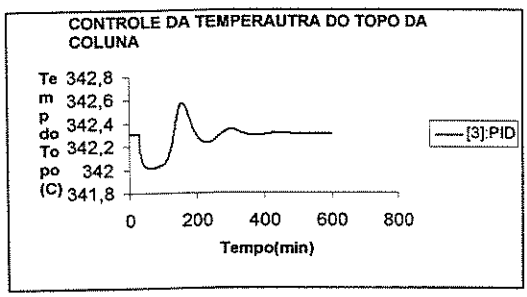


Figura 4.29: Comportamento da temperatura de topo da coluna de destilação quando se modifica a vazão de alimentação da coluna. [Obs.: este gráfico não apresenta a curva em malha aberta por causa de problemas de escala].

A Figura 4.30 mostra as três curvas anteriores sobrepostas nos mesmos ordenados.

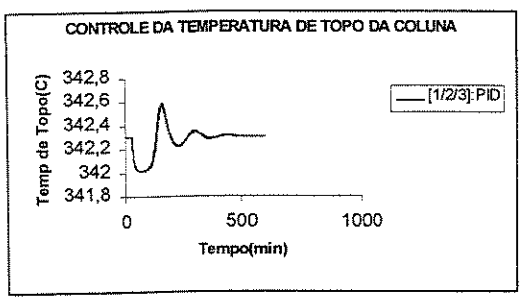


Figura 4.30: Composição das curvas de controle da temperatura de topo da coluna de destilação estudada. [As curvas quase que sobrepõem devido à pequena distância que separa um ponto do outro, só dá para perceber as diferenças observando-se o arquivo de dados, o qual foi utilizado para a confecção destas curvas].

CAPÍTULO 5

5.0 REDES NEURAIS

5.1 INTRODUÇÃO

Para resolução de problemas baseados em relações aritméticas, computadores são, usualmente, mais capazes que os humanos. Contudo, ainda existem muitas tarefas que os humanos realizam muito mais rápido e facilmente que as máquinas. Uma destas tarefas é o reconhecimento de padrões. Informações, mesmo incompletas, podem ser decifradas pelos humanos, coisa nem sempre possível com um computador.

Modelos de redes neurais artificiais, ou modelos de processamento paralelo, têm sido estudados há vários anos (cerca de quarenta anos) na esperança de se conseguir chegar a um funcionamento que reproduza o comportamento do cérebro humano, no reconhecimento de padrões e de imagens pré-estabelecidas.

Não existe uma definição consensual para as redes neurais. Cada autor, de acordo com o tipo de rede utilizada e de acordo com as características do modelo tem sua própria definição. Como exemplo, pode-se transcrever a definição de redes encontrada em NIGRIN(1993). "Uma rede neural é um circuito composto de um grande número de elementos de processamento simples os quais são 'neuralmente' baseados. Cada elemento opera somente no local da informação. Além do mais, cada elemento opera assincronamente".

A arquitetura das redes neurais artificiais é motivada, historicamente, pelo "estilo computacional" encontrado no sistema nervoso biológico, o qual tem cerca de 10^{11} neurônios, HERTZ et al. (1991). Contudo, uma rede neural eficiente não deve conter os mesmos erros 'ordinários' dos sistemas biológicos nos quais estão baseados, ROY et al.(1997). As características chaves são um largo número de unidades de processamento não-lineares relativamente simples e o alto grau de conectividade entre estas unidades. Muitos dos modelos de "treinamento" estão baseados na regra de que os pesos das conexões são ajustados por meio de dados, ou seja, as ANN aprendem a partir de exemplos, e em alguns casos, apresentam condições de generalização para dados não utilizados no

treinamento. As ANN's normalmente têm grande capacidade de processamento, uma vez que as computações dos componentes são independentes uns dos outros, e em princípio, podem ajustar qualquer tipo de função que seja mapeável[25]. Segundo WHITE(1990), redes neurais em *feedforward*[26] são estimadores estatisticamente consistentes de medidas arbitrárias. São especialmente úteis para se ajustar dados quando não se tem uma relação funcional conhecida e se dispõe de uma grande quantidade de dados para se proceder o treinamento e quando se tolera alguma imprecisão no ajuste. Redes *feedback* ou recorrentes são modelos onde as informações de entrada definem o estado de atividade do sistema. A figura (5.0) mostra um esquema dos modelos destes tipos de redes.

Uma grande limitação da utilização das redes neurais se refere à capacidade de extrapolação dos ajustes desejados. Não existe ainda um algoritmo capaz de "criar" informações a partir de um conjunto de dados utilizado para o treinamento sem que os mesmos tenham sido fornecidos durante o processo de treinamento. Daí, a principal característica da rede neural ser a de reconhecedora de padrões, melhor que a de ajustadora de funções, mesmo que para alguns casos isto seja possível.

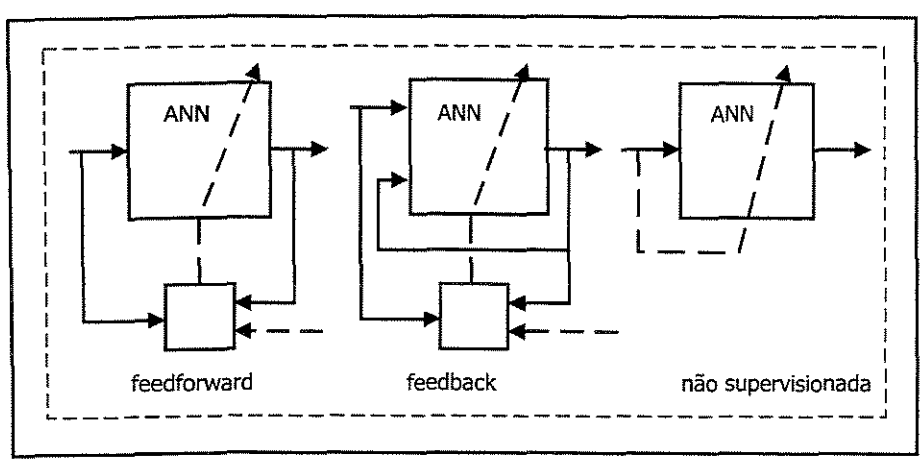


Figura 5.0: Modelos de redes neurais. As linha em negrito são as saídas desejadas da rede e o esquema de treinamento é mostrado pelas linhas pontilhadas.

[25] Para o caso de funções polinomiais existem provas matemáticas de que a aproximação por redes neurais é sempre possível, ver por exemplo RUMELHART e McCLELLAND (1986).
[26] Feedforward: os sinais são passados da entrada para a saída sem retorno de ligações, através de camadas ocultas. Pode ser uma transformação supervisionada ou não.

As duas primeiras regras de treinamento de elementos adaptativos são o *perceptron* e o LMS, ambos desenvolvidos por volta da década de 50. WIDROW (1962) e seus colaboradores desenvolveram o Madaline Rule I (MRI) a primeira e mais popular regra de aprendizagem para redes neurais com múltiplos elementos adaptativos. Outros trabalhos relacionados ao mesmo campo de desenvolvimento começaram então a aparecer. Estudos que relacionavam o funcionamento do cérebro humano aos desenvolvimentos matemáticos passaram a ter cada vez mais interfaces comuns.

Hodgin e Huxley foram os primeiros a explicarem os mecanismos envolvidos na criação e propagação dos sinais neuronais elétricos, mas é devida a McCULLOCH e PITTS(1943) a primeira modelagem de neurônios, em 1943. Na visão deles, um neurônio é uma ponte lógica com dois possíveis estados internos, ativo ou desativo. Um neurônio tem algumas entradas e saídas conectadas a outros neurônios. As entradas são somadas e o estado do neurônio é determinado pelo valor do sinal resultante com respeito a um certo *threshold*: se o sinal é maior que o *threshold* o neurônio é ativo, de outro modo é não-ativo.

Durante a década de 70, GROSSBERG(1976), desenvolveu sua Teoria de Ressonância Adaptativa (TRA), a qual é composta de novas hipóteses sobre o princípio que governa os sistemas biológicos, as quais serviram mais tarde para futuros trabalhos do mesmo autor.

Uma outra teoria de grande importância desenvolvida para auto-organização, é devida a KOHONEN(1984), com seu trabalho de "mapas de características". No inicio da década de 80, HOPFIELD(1982) também introduziu outras regras de produtos para treinamento de redes. Outros modelos significantes da década passada incluem modelos probabilísticos tais como os de HINTON et al.(1984) e HINTON e SEJNOVSKI(1984), os quais, para simplificar, é um modelo como o de Hopfield que resolve o problema utilizando *simulated anneling*, governado pela estatística de Boltzmann.

A possibilidade de aplicação em diversos campos contribuiu para que o interesse pelo desenvolvimento de novas técnicas empregasse a cada dia mais e mais pesquisadores. A primeira grande extensão das redes neurais, em *feedforward*, depois do Madaline I, ocorreu quando WERBOS(1974) desenvolveu o algoritmo de treinamento *backpropagation*, o qual ficou sem interesse por um bom período de tempo, evidenciado pela falta de trabalhos, na mesma linha, logo após o desenvolvimento inicial. Pouco depois,

RUMELHART et al.(1986), "redescobriram" a técnica e a publicaram de modo mais claro e melhor desenvolvido. Os elementos utilizados por estes autores, no desenvolvimento do algoritmo de *backpropagation* (que utiliza não-linearidades diferenciáveis na forma de funções sigmoidais, por exemplo), difere daqueles utilizados no desenvolvimento do Madaline (que utiliza quantizadores *hard-limiting*).

Os elementos computacionais são conectados por meio de pesos os quais podem ser ajustados durante o uso a fim de melhorar o seu desempenho. Os nós, ou conexões, utilizados nas redes neurais são não-lineares. A não linearidade mais comumente utilizada é representada pela função de ativação sigmoidal.

Uma rede neural é caracterizada pela sua arquitetura ou pela sua topologia, que inclui o número de camadas e o número de nós, pelas ligações entre seus nós e pelo algoritmo de treinamento e, finalmente, pela sua função de ativação. São estas regras que determinarão como os pesos, inicialmente especificados, se adaptarão para poderem representar o problema proposto.

Existem diversos modelos de redes neurais sendo que cada um tem melhor desempenho de acordo com os padrões dos dados a serem alimentados à rede. Tem-se como principal característica de diferenciação entre os diversos tipos de redes neurais, o fato de o aprendizado ser supervisionado ou não, SARLE(1997), e KOHENEN(1990). No aprendizado supervisionado, existe a utilização dos valores alvo, ou dos valores desejados ao final do treinamento. No aprendizado não supervisionado, a rede é autônoma, ou seja, se "preocupa" apenas com elementos de entrada da rede, encontrando alguma propriedade destes dados e aprendendo a refletir estas propriedades à saída. Tais propriedades e a real capacidade de aprender vai depender de rede para rede de acordo com suas particularidades de construção e funcionamento.

O campo de desenvolvimento das redes neurais é bastante ampliado e não se sabe ao certo, exatamente, a quantidade de tipos diferentes de redes neurais existentes, SARLE(1997). Não é, desta forma, possível se apresentar todos os trabalhos desenvolvidos e todas as características envolvidas em cada um deles.

Muitos tipos de redes podem ser encontradas de forma detalhada com os respectivos algoritmos em trabalhos como os de HOPFIELD(1982), GALLAGER(1968), CARPENTER e GROSSBERG(1986), MINKI e PAPERT(1990), ROSENBLAT(1959),

RUMELHART et al.(1986), RUMELHART e McCLELLAND(1986), LORENTZ(1976), POGGIO e GIROSI(1988), BISHOP(1995), KOHONEN(1984), entre vários outros citados ao longo de todo este trabalho.

5.1.1 O PAPEL DA REDE NEURAL

Uma das metas, quando se usa redes neurais, é encontrar a rede que tenha o melhor desempenho quando se apresenta dados novos para os quais não tenha sido treinada, ou seja a generalização. O caminho mais simples para a comparação de redes de diferentes arquiteturas é avaliar o erro usando dados diferentes dos utilizados no treinamento. Várias redes são treinadas pela minimização de uma função erro apropriada definida com respeito a um conjunto de dados. O desempenho da rede é então comparado avaliando-se a função erro usando um conjunto de validação independente, e a rede que apresentar o menor erro é então escolhida.

Há uma considerável coincidência entre os campos de redes neurais e a estatística. A estatística está preocupada com a análise de dados. Em terminologia de redes neurais, inferência estatística significa aprender para generalizar a partir de dados com ruídos. Algumas redes neurais não estão preocupadas com a análise de dados e desta forma têm pouca relação com a estatística. Ainda, algumas redes neurais não aprendem e também têm pouco a fazer com a estatística. Existem, entretanto, algumas redes neurais que podem aprender, com sucesso, somente a partir de dados livres de ruídos (por exemplo TRA e *Perceptron*) e deste modo, não poderiam ser utilizados métodos estatísticos. Mas muitas redes neurais que podem aprender e generalizar a partir de métodos que podem processar dados com ruídos, são similares, quando não idênticos, aos métodos estatísticos. Por exemplo:

* redes *feedforward* sem camadas ocultas são basicamente modelos lineares generalizados.

* redes *feedforward* com uma camada oculta são proximamente relacionadas a projeções de regressão de atividade.

* redes neurais probabilísticas são idênticas à análise discriminante.

* redes Kohonen por vetor de quantização adaptativa são muito similares às análises de *cluster*.

* aprendizagem de Hebbian está proximamente relacionada à análise do componente principal.

Redes em *feedforward* são um subconjunto das classes de regressão não-linear e modelos discriminantes. Os algoritmos de Levenberg-Marquardt e de Gradiente Conjugado podem ser utilizados para treinar redes com *feedforward*.

Enquanto redes neurais são discutidas em termos de seus algoritmos ou de suas implementações, métodos estatísticos são usualmente definidos em termos de seus resultados.

Por vezes, se diz que redes neurais não requerem considerações de distribuição, diferentemente dos modelos estatísticos. Na realidade, redes neurais envolvem exatamente os mesmos tipos de considerações de distribuição de modelos estatísticos, BISHOP(1995). O treinamento de mínimos quadrados envolve considerações de distribuições implícitas, tal que mínimos quadrados têm certa propriedade de optimalidade para ruídos os quais são normalmente distribuídos com igual variância para todos os casos de treinamento e independentes entre os diferentes casos. Estas propriedades de optimalidade são conseqüência do fato de que mínimos quadrados têm máxima verossimilhança sob estas condições.

A fim de se padronizar o sentido de alguns termos que são discutidos neste trabalho, é útil se ter uma indicação das fontes consultadas, uma vez que é muita vasta e diferenciada a nomenclatura utilizada. Nem todos os autores definem da mesma forma termos que aqui aparecem. A inclusão de definições detalhadas para cada um dos termos, no entanto, ocuparia espaço demasiado no trabalho. Optou-se então por uma breve definição seguida da indicação de referência da literatura aqui consultada, para maiores detalhes. Muitos termos não são fáceis de serem traduzidos para o português, pois uma tradução ao pé da letra muitas vezes não especifica exatamente o que o termo quer dizer.

Desta forma, para se evitar ter que utilizar mais que um termo cada vez que se quiser mencionar o termo, optou-se por não traduzi-lo.

5.2 DEFINIÇÕES

Algumas informações mais detalhadas ajudam a entender certos conceitos que serão desenvolvidos neste trabalho. A maior parte das definições seguem conforme proposto por SARLE(1997). Onde outros autores foram utilizados, a referência está indicada.

Backpropagation: O *backpropagation* é sem dúvida o algoritmo mais estudado no campo das redes neurais. Tem algumas vantagens em relação aos demais algoritmos, como a garantia de convergência, ainda que não necessariamente para o valor desejado (por causa da presença de pontos de mínimos locais), e a facilidade de implementação. É razoavelmente bem entendido, apesar de existirem alguns pontos não totalmente esclarecidos, como por exemplo o valor de certos parâmetros que é ajustado muito mais empiricamente, através de tentativa e erro, do que propriamente por métodos rigorosamente compreendidos.

Estritamente falando, *backpropagation* se refere ao método de computação do gradiente do erro, de trás para frente, para uma rede *feedforward*; uma forma simples mas elegante de aplicação da regra da cadeia do cálculo diferencial. De forma sintética, rede neural *backpropagation*, é uma rede com *feedforward* treinada com propagação reversa do erro.

Na verdade, o *backpropagation* é nada mais que a regra delta generalizada, GDR, desenvolvida por WERBOS(1974) e popularizada por RUMELHART et al.(1986).

Métodos de Otimização: Treinar uma rede neural é em muitos casos uma otimização numérica de uma função objetivo normalmente não-linear (função objetivo significa

qualquer função que se esteja tentando otimizar e é ligeiramente mais geral que função erro, uma vez que ela pode incluir outras quantidades tais como restrições para o decaimento dos pesos). Infelizmente, não existe um método de otimização não-linear que seja ótimo em qualquer situação. É preciso que se escolha o método de acordo com as características do problema a ser resolvido.

Para funções objetivo com derivadas segundas contínuas (entre as quais se incluem redes *feedforward* com função de ativação *sigmoidal* e as funções erro), três tipos de algoritmos gerais têm sido encontrados como sendo mais efetivos para muitas propostas práticas:

* Para pequeno número de pesos, algoritmos de Gauss-Newton, incluindo suas variantes tais como Levenberg-Marquardt e algoritmos de região de confiança, são eficientes.

* Para moderado número de pesos, alguns algoritmos quase-Newton são eficientes.

* Para um grande número de pesos, algoritmos de gradiente-conjugado[27] são mais eficientes.

Para funções objetivo que não sejam continuamente diferenciáveis, o algoritmo *simplex* de Nelder-Mead e o *simulated annealing* podem ser usados.

Todos os métodos citados acima encontram ótimos locais, contudo eles não dão garantia de encontrarem um ótimo global. Na prática, Levenberg-Marquardt, freqüentemente, encontra o melhor ótimo para uma grande variedade de problemas em comparação aos outros métodos, contudo, tem uma séria restrição quanto ao esforço computacional necessário para sua realização.

Para otimização global, também há uma grande variedade de caminhos. Pode-se simplesmente determinar qualquer dos ótimos locais a partir de numerosos pontos de partida para os pesos iniciais. Outras formas de se encontrar ótimos globais podem ser tentados utilizando algoritmos mais complexos como por exemplo *simulated annealing* e algoritmos genéticos, REEVES(1993).

---

[27] Ver Apêndice A.

Redes não Supervisionadas: Redes não supervisionadas não envolvem valores alvo. Na verdade, para muitas variedades de aprendizados sem supervisão, os alvos são os mesmos que as entradas, SARLE(1994). A forma mais atual de treinamento não supervisionado encontrado é representada pelo Mapa Auto-organizador de Kohonen, KOHONEN(1995).

Redes Supervisionadas: Aprendizagem supervisionada é aquela em ocorre através de exemplos, os padrões de treinamento. O conjunto de treinamento consiste de elementos os quais são valores dados aos pares de entrada (independente) e saída (dependente). O conjunto de treinamento no qual há n pares (indexados por i , o qual vai de 1 até N) é representado por

$$\tau = \{(\mathbf{x}_i, \hat{\mathbf{y}}_i)\}_{i=1}^{n}$$

(o circunflexo no y se justifica pelo fato de ser um valor "incerto", pois o valor de saída é admitido ser sempre corrompido por ruído, $\mathbf{y} = \hat{\mathbf{y}} + \mathbf{ruídos}$). Em alguns casos, a variável independente também pode estar corrompida com ruído.

Regressão Paramétrica e não Paramétrica: Existem duas grandes subdivisões em problemas estatísticos: paramétricos e não paramétricos. Em regressão paramétrica a forma da relação funcional entre as variáveis dependentes e independentes é conhecida e contém parâmetros os quais são possíveis de serem determinados através de um conjunto de treinamento. A característica que distingue a regressão não paramétrica é que não há (ou há muito pouco) conhecimento a respeito da forma da função verdadeira a qual está sendo estimada. Tipicamente, este último tipo de regressão envolve a utilização de muitos parâmetros os quais não têm relação 'física' com o problema. Redes neurais, incluindo redes com RBF[28], são modelos não paramétricos e seus pesos ou parâmetros não têm significado em relação aos problemas aos quais são aplicados.

---

[28] O desenvolvimento do algoritmo de redes neurais com RBF (Radial Basis Function) está descrito no Capítulo 6.

Bias e Threshold: Camadas ocultas e camadas de saída usualmente utilizam um termo de "bias" ou um *threshold* para computação da saída de cada nó da rede. O termo de *bias* pode ser tratado como o peso de uma conexão com um valor constante, por exemplo, um. Consequentemente, o *bias* pode ser visto apenas como um outro peso, e desta forma, ajustável durante o procedimento de treinamento da rede. Para uma unidade de saída linear, um termo de *bias* é equivalente a uma intersecção em um modelo de regressão linear, podendo desta forma ser considerado como um classificador ou limitador de classes.

Considere um perceptron de múlti-camada com qualquer uma das funções de ativação. Para um conjunto de treinamento com N variáveis independentes, tem-se um espaço de entrada N-dimensional. As unidades geram um hiperplano neste espaço, produzindo duas regiões onde as classificações recebem denominações contrárias.

Os pesos determinam onde este hiperplano se encontra no espaço e quanto menor for a sobreposição de características das classes, melhor será a classificação. Sem um neurônio que represente o bias, o hiperplano fatalmente passará pelo ponto de intersecção dos eixos coordenados. Para alguns problemas, isto pode não trazer nenhuma restrição, contudo, em outros, a mudança de localização do hiperplano pode ser muito útil.

O valor do bias quando muito grande implicará muita inflexibilidade de ajuste dos parâmetros, o que provoca mal ajuste dos dados. Juntamente com a variância, a qual age de modo contrário ao bias, produzindo muita flexibilidade no modelo quando tem valor grande, é um importante fator de ajuste dos parâmetros de uma rede neural, e suas implicações podem ser vistas em GEMAN et al. (1992). A forma de otimização da complexidade da rede neural passa pelo controle dos valores do bias e da variância, num processo chamado de estabilização estrutural. Todo este ajuste pode ser mais simplesmente conseguido se forem tentadas heurísticas empíricas. De outra forma, uma forma mais exata de se conseguir o ajuste da complexidade da estrutura neural é através da teoria da regularização.

Todo o problema de ajuste e classificação de variáveis pode ser melhor entendido em DUDA e HART(1973) , FUKUNAGA(1982) e BISHOP(1995), onde se mostra que para melhorar o desempenho da rede neural, deve-se ter um algoritmo capaz de ajustar a dupla *bias*-variância de tal modo que os dois valores sejam mínimos.

A necessidade do *threshold* está no fato de que a linha de separação pode ir se ajustando, ao passo que se o w_0[29] for zero, não se terá a possibilidade de se afastar do início dos eixos coordenados, seja em que dimensão for, e desta forma perde-se um 'grau de liberdade'. O problema da sobreposição de camadas se resolve porque é como uma função "e" (aditiva) em estatística, multiplica-se ou se sobrepõem os hiperplanos formando uma região convexa.

Quando se tem que o número de iterações aumenta quando não se fixa o valor do *threshold* é por causa das particularidades do problema em questão. Pode acontecer que pelo fato de se deixar todos os parâmetros 'soltos' para serem ajustados dificulte a convergência, ao passo que se se utiliza um valor fixo pode levar a convergência a ser melhor. De todo modo, cada sistema é um sistema e é preciso ter em mente que a melhor arquitetura para um determinado sistema não será, necessariamente, também para outro.

Reconhecimento de Padrões: A proposta do reconhecimento de padrão é determinar a qual categoria ou classe uma dada amostra pertence. O problema de classificação é basicamente o de dividir o espaço em regiões; uma região para cada categoria.

Para muitas aplicações de reconhecimento de padrões reais, o fato de haver sobreposições entre algumas distribuições é inevitável, mas pode ser contornado aumentando-se a complexidade da região de decisão, tornando-a mais abrangente. Deve-se ter em mente que a classificação de exemplos novos nem sempre é possível, pois a camada de separação ou o hiperplano uma vez ajustado, tem localização fixa não podendo mais ser deslocado sem que se faça um novo ajuste para as novas condições.

Função de Ativação: A função de ativação deve ser vista como uma transformação sobre os valores de entrada, seja da camada de entrada para as camadas ocultas, ou destas para a camada de saída. Pode ser entendida como uma forma de se ver a probabilidade de se atribuir a uma classe determinado vetor de entrada sendo dada a saída.

Funções de ativação são importantes para introduzir não linearidades na rede. Sem não-linearidades, as camadas ocultas não conseguiriam fazer das redes mais que o que

---

[29] Símbolo normalmente atribuído ao peso. O índice zero se refere ao peso do *bias*.

fazem simples *perceptrons* planos (os quais não têm quaisquer camadas ocultas apenas unidades de entrada e saída). A razão é que a composição das funções lineares é novamente uma função linear. Entretanto, é a não linearidade (isto é, a capacidade para representar funções não lineares) que faz as redes multicamadas tão poderosas. Quase todas as funções não lineares podem ser usadas como função de ativação, apesar de que para o aprendizado em *backpropagation* ela deve ser diferenciável, e ainda ajuda se a função for limitada[30]. As funções *sigmoidais* tais como as logísticas, a função **tanh** e a função *Gaussiana* são as escolhas mais comuns.

Função de ativação é também usada para designar a função discriminante por alguns autores, KAVURI e VENKATASUBRAMANIAN(1993), por exemplo, contudo, é bom se ter em mente que se trata de dois conceitos que se referem a etapas diferentes no processo de treinamento. Neste trabalho os termos estão em conformidade com a denominação mais normalmente divulgada.

Para as unidades de saída, deve-se escolher uma função de ativação que seja ajustada para a distribuição dos valores alvo. Funções de ativação limitadas tais como as funções logísticas são particularmente úteis quando os valores alvo tem uma faixa limitada. Mas se os valores alvo não têm uma faixa limitada, é melhor utilizar uma função de ativação não limitada, muito freqüentemente, a função identidade (o que significa sem função de ativação). Se os valores alvo são positivos mas tem um limite superior conhecido, pode-se usar uma função de ativação exponencial à saída, contudo, é preciso tomar cuidado com *overflow*.

Funções Discriminantes Lineares e superfície de Decisão: A escolha de funções discriminantes claramente não é única. Pode-se sempre multiplicar uma função discriminante por uma constante ou por um *bias* sem influenciar a decisão. Mais geralmente, se trocamos toda $g_i(x)$ por uma $f(g_i(x))$, onde f é uma função monotonicamente crescente, o resultado da classificação permanece inalterado. Esta observação pode levar a significantes simplificações analíticas e computacionais.

---

[30] Por exemplo uma função elipsoidal, esférica ou mais genericamente hiperesférica.

Uma função discriminante que é uma combinação linear dos componentes de **x** pode ser escrita como

$$\mathbf{g(x) = w^t x + w_0} \tag{5.1}$$

onde **w** é chamado o vetor peso e $\mathbf{w_0}$ é o peso *threshold.* Um classificador linear de duas categorias implementa a seguinte regra de decisão: decide ω_1 se $\mathbf{g(x) > 0}$ e ω_2 se $\mathbf{g(x) < 0}$. Assim, **x** é atribuído a ω_1 se o produto interno $\mathbf{w^t x}$ excede ao threshold - $\mathbf{w_0}$. Se $\mathbf{g(x) = 0}$, **x** pode, ordinariamente, ser atribuído a outra classe.

A equação $\mathbf{g(x) = 0}$ define a superfície de decisão que separa pontos atribuídos a ω_1 dos atribuídos a ω_2. Quando $\mathbf{g(x)}$ é linear, a superfície de decisão é um hiperplano. Se x_1 e x_2 estão ambos na superfície de decisão, então

$$\mathbf{w^t x_1 + w_0 = w^t x_2 + w_0}$$

ou

$$\mathbf{w^t (x_1 - x_2) = 0}$$

tal que **w** é normal a qualquer vetor que pertença ao hiperplano (notar que isto ocorre pelo fato de que se o produto interno ser zero, assim, os vetores **w** e ($\mathbf{x_1}$ e $\mathbf{x_2}$) devem ser ortogonais). Em geral, o hiperplano *H* divide o espaço característico em dois meio-espaços, a região $\Re_1$ para ω_1 e a região de decisão $\Re_2$ para ω_2. Desde que $\mathbf{g(x) > 0}$ se **x** está em $\Re_1$ segue que o vetor normal **w** aponta dentro de $\Re_1$. É algumas vezes dito que qualquer **x** em $\Re_1$ está no lado positivo de *H*, e qualquer **x** em $\Re_2$ está no lado negativo.

A função discriminante **g(x)** dá a medida algébrica da distância de **x** ao hiperplano. Talvez o caminho mais fácil de ver isto é expressar **x** como

$$\mathbf{x} = \mathbf{x}_p + \mathbf{r}\frac{\mathbf{w}}{\|\mathbf{w}\|},$$

onde $\mathbf{x}_p$ é a projeção normal de $\mathbf{x}$ em H, e $\mathbf{r}$ é a distância algébrica desejada, positiva se $\mathbf{x}$ está no lado positivo e negativo se $\mathbf{x}$ está no lado negativo. Então, desde que $\mathbf{g}(\mathbf{x}_p) = \mathbf{0}$, ($\mathbf{x}_p$ é ortogonal a H)

$$\mathbf{g}(\mathbf{x}) = \mathbf{w}^t\mathbf{x} + \mathbf{w}_0 = \mathbf{r}\|\mathbf{w}\|$$

ou

$$\mathbf{r} = \frac{\mathbf{g}(\mathbf{x})}{\|\mathbf{w}\|}$$

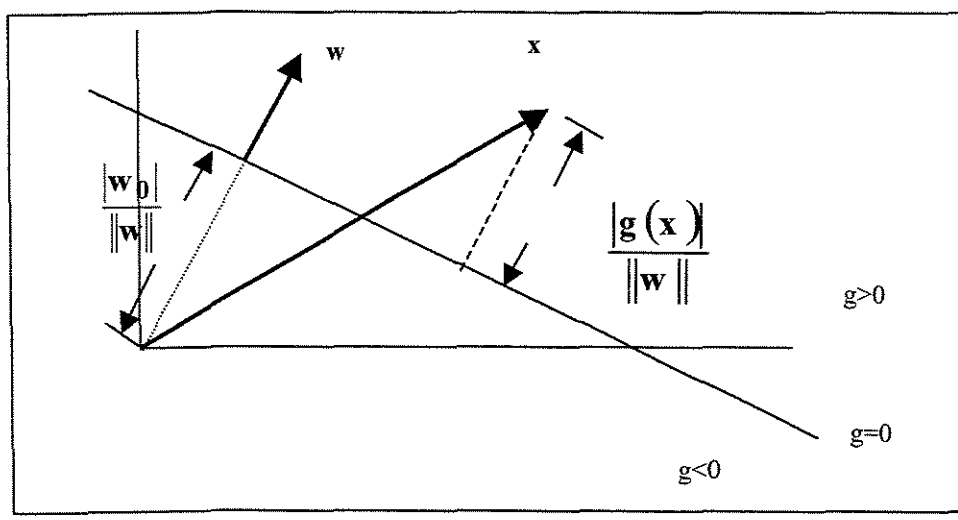


Figura (5.5): Camada de decisão linear $\mathbf{g}(\mathbf{x}) = \mathbf{w}^t\mathbf{x} + \mathbf{w}_0 = \mathbf{0}$.

Em particular, a distância a partir da origem até H é dado por $\mathbf{w_0}/\|\mathbf{w}\|$. Se $\mathbf{w_0} > \mathbf{0}$ a origem está no lado positivo de H, e se $\mathbf{w_0} < \mathbf{0}$ está no lado negativo. Se $\mathbf{w_0} = \mathbf{0}$, então $\mathbf{g(x)}$ tem a forma homogênea $\mathbf{w^tx}$, e o hiperplano passa através da origem.

Um problema com a convergência do *perceptron* é que a região de decisão pode oscilar continuamente quando as entradas não são separáveis e quando as distribuições se sobrepõem.

Para sumarizar, um discriminante linear divide o espaço através de uma superfície de decisão. A orientação da superfície é determinada pelo vetor normal **w**, e a localização da superfície é determinada pelo *threshold* $\mathbf{w_0}$. A função discriminante **g(x)** é proporcional à distância de **x** ao hiperplano, com **g(x) > 0** quando **x** está no lado positivo, e **g(x) < 0** quando **x** está no lado negativo.

Superfícies de Decisão para o Caso de Múlti Categoria: Há mais que um caminho para imaginar classificadores empregando funções discriminantes lineares. Por exemplo, reduzir o problema a c-1 problemas de duas classes, onde o i-ésimo problema possa ser resolvido por um discriminante linear que separa pontos atribuídos a ω_i a partir daquele não atribuído a ω_i. Uma atitude mais extrema seria usar c(c-1)/2 discriminantes lineares, um para cada par de classes. Como ilustrado na Figura (5.6), ambos os caminhos podem levar a regiões marcadas como as áreas sombreadas, nas quais as classificações são indefinidas. Este problema pode se evitado definindo-se c funções discriminantes lineares

$$\mathbf{g_i(x) = w_i^t x + w_{i0}}, \qquad \mathbf{i = 1,...,c}, \tag{5.2}$$

e atribuindo **x** a $\mathbf{w_i}$ se $\mathbf{g_i(x) > g_j(x)}$ para todo $\mathbf{i \neq j}$. Em caso de empate, o classificador é dito ser indeterminado. O classificador resultante é uma máquina linear. Uma máquina linear divide o espaço em c regiões de decisão, com $\mathbf{g_i(x)}$ sendo o maior discriminante se x é tomado na região $\Re_i$. Se $\Re_i$ e $\Re_j$ são contíguos, o contorno entre eles é uma porção do hiperplano H_{ij} definido por

$$\mathbf{g_i(x) = g_j(x)}$$

ou

$$(\mathbf{w}_i - \mathbf{w}_j)^t \mathbf{x} + (\mathbf{w}_{i0} - \mathbf{w}_{j0}) = 0$$

Segue que o vetor $(\mathbf{w}_i - \mathbf{w}_j)$ é normal à matriz H_{ij}, e a distância atribuída de $\mathbf{x}$ a H_{ij} é dado por $(\mathbf{g}_i - \mathbf{g}_j)/ \| \mathbf{w}_i - \mathbf{w}_j \|$. Assim, com a máquina linear não são os vetores pesos em si que são importantes mas suas diferenças. Enquanto existem c(c-1)/2 pares de regiões, elas não necessitam serem contíguas, e o número total de segmentos de hiperplanos que aparecem na região de decisão é freqüentemente menor que c(c-1)/2. Exemplos bi-dimensionais destas superfícies são dados na Figura (5.7).

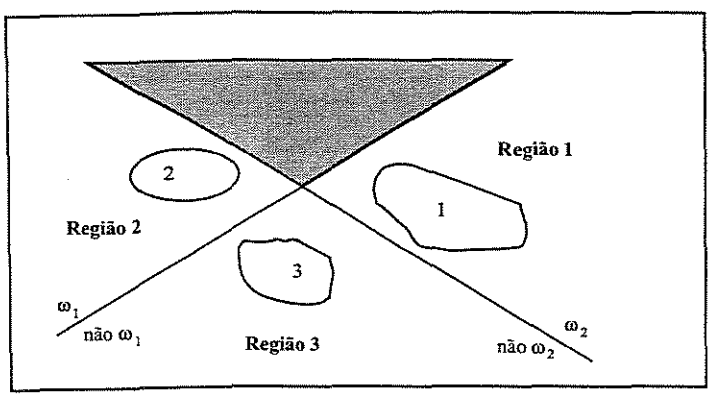


Figura (5.6): Contornos lineares para um problema de três classes.

As regiões de decisão para uma máquina linear são convexas e esta restrição definitivamente limita a flexibilidade do classificador. Em particular, toda região de decisão deve ser singularmente conectada, o que tende a fazer a máquina linear mais apropriada para problemas para os quais as densidades condicionais $\mathbf{p}(\mathbf{x}|\varpi_1)$ são unimodais. Dentro destas limitações, a máquina linear oferece razoável quantidade de flexibilidade e a virtude de ter simplicidade analítica.

Para perceptrons de multicamadas não pode ser mostrado que a convergência acontece, contudo ela é verificada em muitos problemas de interesse, LIPPMANN(1987). O número de nós deve ser grande o suficiente para formar a região de decisão requerida

pelo problema dado, mas deve ser grande até o limite que os pesos requeridos possam ser corretamente estimados a partir do conjunto de dados de treinamento disponível.

Conforme dito anteriormente, *perceptrons* com três camadas podem gerar qualquer região complexa arbitrária, o que indica que não são, portanto, necessárias redes com mais que três camadas para se fazer a aproximação de qualquer função.

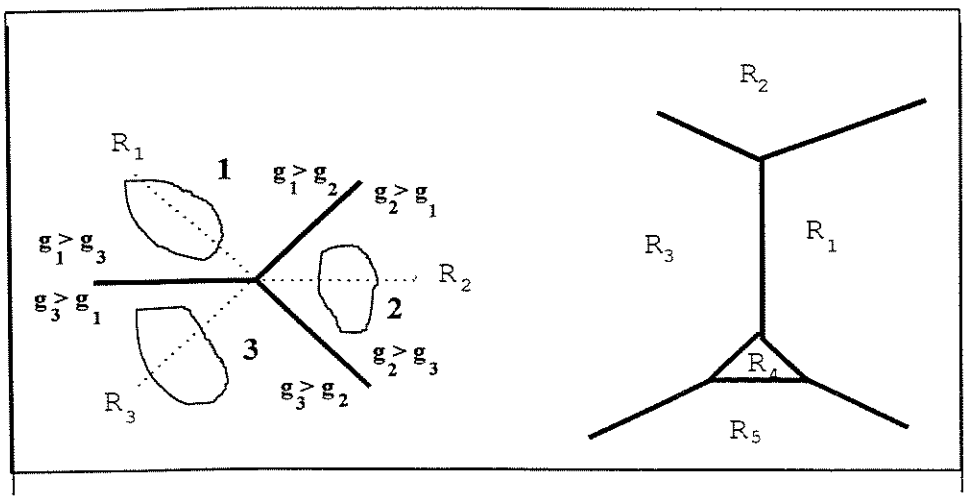


Figura (5.7): Exemplos de superfícies bidimensionais.

Funções Discriminantes Generalizadas: Foi mostrado que funções de ativação não lineares (por exemplo *sigmoidal*) agindo sobre uma combinação linear das entradas da forma:

$$\mathbf{a}_j = \sum_i \mathbf{w}_{ji}\mathbf{x}_i + \mathbf{w}_{j0} \qquad (5.3)$$

podem, em princípio, aproximar qualquer mapeamento funcional com uma precisão arbitrária, e desta forma, constitui-se um aproximador universal. Apesar disto, é interessante estudar outras formas de unidades de processamento[31]. Conforme já discutido em itens anteriores, uma rede contendo uma simples camada de unidades da forma da equação (5.3) pode somente produzir contornos de decisão que tomam a forma de

[31] No capítulo 6 faz-se o estudo de redes contendo unidades cujas ativações dependem da distância de um vetor de entrada do vetor peso.

hiperplanos no espaço de entrada. Tais redes são incapazes, contudo, de gerar regiões de decisão que sejam côncavas ou que sejam multiplamente conectadas.

Considere o espaço de entrada x ilustrado na Figura (5.8). Deseja-se encontrar uma função discriminante a qual divida o espaço em regiões de decisão $\Re_1$ e $\Re_2$ conforme mostrado. Uma função discriminante linear não é suficiente desde que a região $\Re_1$ é disjunta. Entretanto, a região de decisão requerida pode ser gerada por um discriminante quadrático da forma

$$\mathbf{y(x)} = \mathbf{w}_2\mathbf{x}^2 + \mathbf{w}_1\mathbf{x} + \mathbf{w}_0 \tag{5.4}$$

onde os peso são convenientemente determinados.

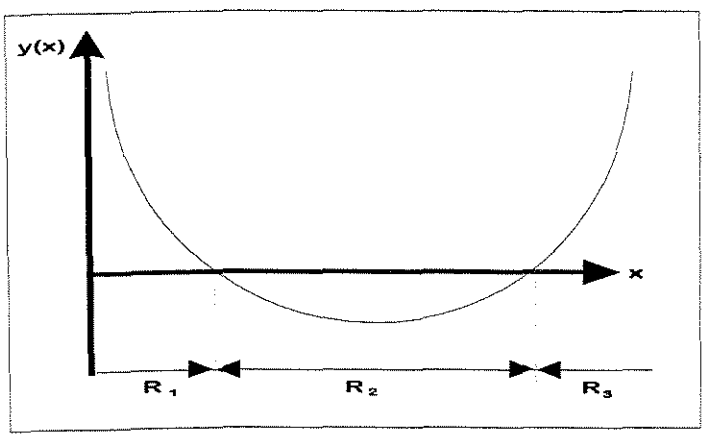


Figura (5.8) : Espaço de entrada unidimensional x com duas regiões de decisão disjuntas.

Pode-se generalizar esta idéia para ordens mais altas que apenas a quadrática, e para diversas variáveis de entrada, GILES E MAXWELL(1987) e RUMELHART et al. (1986). Para unidades de segunda ordem, a generalização de (5.3) toma a forma de

$$\mathbf{a}_j = \mathbf{w}_j^{(0)} + \sum_{i_1=1}^{d} \mathbf{w}_{ji_1}^{(1)} \mathbf{x}_{i_1} + \sum_{i_1=1}^{d} \sum_{i_2=1}^{d} \mathbf{w}_{ji_1i_2}^{(2)} \mathbf{x}_{i_1} \mathbf{x}_{i_2} \tag{5.5}$$

onde a soma ocorre sobre todas as entradas, ou unidades, as quais contribuem com ligações para j. A função quadrática geral pode ter diversas formas dependendo das condições baseadas nos coeficientes. Tais contornos podem ser elipses, hipérboles, parábolas, pares de linhas retas ou quádricas imaginárias. Como antes, esta soma é então transformada usando-se uma função de ativação não linear para dar $z_j = g(a_j)$. Notar que o somatório na equação (5.5) pode ser restringido para permitir a simetria dos termos de alta ordem. Por exemplo o termo x_1x_2 é equivalente ao termo x_2x_1 e assim, é preciso determinar apenas um destes termos no somatório. O número total de parâmetros independentes na expressão de alta ordem é então d(d-1)/2. A explosão no número de parâmetros é o principal argumento contrário ao uso de unidades com alta ordem. A compensação é que é possível arranjar para que a resposta da unidade seja invariante para várias transformações da entrada. Isto é conseguido impondo-se restrições nos pesos, o que reduz grandemente o número de parâmetros independentes, e desta forma faz com que o uso de tais unidades seja uma proposição factível. Unidades de alta ordem são geralmente utilizadas somente na primeira camada da rede, com as subsequentes camadas sendo compostas de unidades de primeira ordem convencional.

Curso De Dimensionalidade: Curso de dimensionalidade refere-se ao crescimento exponencial do hipervolume como uma função da dimensionalidade do problema. Cada unidade de entrada a mais na rede neural adiciona outra dimensão ao espaço no qual os dados dos padrões residem. Pensando desta forma, deve haver um conjunto de padrões disponíveis para ocupar, suficientemente, o espaço de entrada de tal forma a permitir à rede neural "ver" a estrutura da função que se quer mapear. O número de pontos necessários para fazer isto de modo correto cresce muito rapidamente e com eles, a dimensionalidade. Muitas formas de redes neurais (em particular MLP's) sofrem menos com a dimensionalidade que alguns outros métodos. Existem diversas técnicas para se conseguir reduzir a dimensionalidade de um problema, algumas eficientes e outras nem tanto.

Projection Pursuit Regression (PRP) e Outras Técnicas Convencionais: Existe uma grande variedade de técnicas para classificação e regressão as quais podem ser consideradas como complementares ao perceptron de múlti camadas. Uma das mais próximas às redes

neurais é a PRP, FRIEDMAN e STUELZE(1981). Para uma simples variável de saída, a PRP pode ser escrita da forma

$$\mathbf{y} = \sum_{j=1}^{M} \mathbf{w}_j \phi_j \left(\mathbf{u}_j^T \mathbf{x} + \mathbf{u}_{j0} \right) + \mathbf{w}_0 \qquad (5.6)$$

Os parâmetros $\mathbf{u_j}$ e $\mathbf{u_{j0}}$ definem a projeção do vetor[32] de entrada $\mathbf{x}$ sobre o conjunto de planos rotulados por j=1,...,M, como no *perceptron* de múlti-camadas. Estas projeções são transformadas pela 'função de ativação' não-linear ϕ_j e estas por sua vez são linearmente combinadas para formar a variável de saída $\mathbf{y}$. Otimização das funções de ativação pode ser feita por técnicas com *splines* cúbicas, PRESS et al(1992). A otimização dos pesos da primeira camada requer técnicas nãolineares. Em termos de capacidade representacional, pode-se considerar o PRP como uma generalização do *perceptron* de múlticamada, no qual a função de ativação é mais flexível.

Outra estrutura para regressão não-linear é a classe de modelos aditivos generalizados, HASTIE E TIBSHIRANI (1990) a qual é representada por

$$\mathbf{y} = \mathbf{g}\left(\sum_{i=1}^{d} \phi_i (\mathbf{x}_i) + \mathbf{w}_0 \right) \qquad (5.7)$$

onde $\phi_i(\bullet)$ são funções não lineares e $\mathbf{g}(\bullet)$ representa a função logística *sigmoidal*, BISHOP(1995). Esta é uma classe de modelos realmente muito restritiva uma vez que não permite interações entre as variáveis de entrada. Deste modo, uma função da forma $\mathbf{x_1 x_2}$ por exemplo, não pode ser modelada. Existem outras extensões para este modelo.

Teoria De Ressonância Adaptativa-TRA: Teoria de Ressonância Adaptativa foi criada por Grossberg, GROSSBERG(1976). TRA abrange uma grande variedade de redes neurais

---

[32] Ver TEIXEIRA(1997), sobre projeção matricial.

baseada explicitamente em neurofisiologia, definidas em termos de equações diferenciais detalhadas as quais se propõem a ser modelos plausíveis de neurônios biológicos. Na prática, redes ART são implementadas usando soluções ou aproximações destas equações diferenciais. A ART pode ser formulada como algoritmo supervisionado ou não-supervisionado. Conforme discutido por MOORE(1988), as ART's supervisionadas são basicamente similares a muitos algoritmos de agrupamento.

Rede Neural Probabilística-RNP: PNN significa Rede Neural Probabilística e é um termo *kernel* de análise discriminante. São unidades RBF chamadas *kernels* e são usualmente funções probabilidade como a Gaussiana. SPECHT(1990) garante que a PNN treina 100000 vezes mais rápido que o *backpropagation.*

PNN é um aproximador universal para problemas bem comportados, e assim deve ser capaz de classificar problemas quando se tem uma quantidade suficiente de dados. O principal inconveniente da PNN é que, como métodos *kernel* em geral, ela sofre com o curso da dimensionalidade. PNN não pode ignorar entradas irrelevantes sem que se faça uma modificação no algoritmo básico. Assim se se tem mais que 5 ou 6 medidas redundantes, não é a melhor escolha como classificador.

Rede Neural de Regressão Geral- RNRG: é a Rede Neural de Regressão Geral ou também Rede de Schiöler e Hartmann, SCHIÖLER e HARTMANN(1992). É uma RBF normalizada na qual há uma unidade centrada a cada caso de treinamento. É um aproximador universal de funções bem comportadas, assim deve ser capaz de aproximar bem qualquer função nestas condições a partir de um conjunto suficiente de dados. O principal problema é que também sofre dos problemas de dimensionalidade.

Generalização das Redes: Durante o treinamento, as saídas das redes supervisionadas aproximam os valores de saída de acordo com os alvos desejados. Esta habilidade pode ser útil tendo apenas um fim em si mesma. Contudo, freqüentemente, o que se deseja é generalizar, i. e., ter as saídas da rede próximas a saídas desejadas quando se usa entradas não empregadas no treinamento da rede, mas a generalização nem sempre é possível.

Há três condições que são necessárias (mas não suficientes) para uma boa generalização.

A primeira condição é que as entradas para a rede contenham informação em quantidade suficiente para se atingir os valores alvo, de tal forma que exista uma função matemática que relacione corretamente as saídas às entradas com o desejado grau de precisão. Não se pode esperar que uma rede aprenda uma função não existente. Encontrar bons tipos de dados de treinamento e em quantidade suficiente é uma tarefa que pode tomar mais tempo que o treinamento propriamente dito.

A segunda condição necessária é que a função que se está tentando treinar seja, de alguma forma, bem comportada. Em outras palavras, uma pequena mudança nas entradas, deverá, ao longo do tempo, produzir pequenas mudanças nas saídas. Para entradas e saídas contínuas, bom comportamento da função implica continuidade e restrições na primeira derivada sobre o espaço de entrada. Algumas redes neurais pode aprender descontinuidades contanto que a função consista de um número finito de pedaços contínuos. Muitas funções não muito bem comportadas tais como aquelas produzidas por geradores de números pseudo-randômicos e algoritmos de criptografia não podem ser generalizados por redes neurais. Freqüentemente, uma transformação não linear das variáveis de entrada pode melhorar o comportamento da função e melhorar a generalização. Para classificação, se não é necessário estimar probabilidades, então esta condição não é, teoricamente, necessária. Em particular, redes *feedforward* treinadas com uma camada oculta pela minimização da taxa de erro são classificadores consistentes se o número de unidades ocultas cresce a uma taxa relativa conveniente ao número de casos treinados. Para funções *booleanas*, o conceito de comportamento é menos explícito.

A terceira condição necessária para uma boa generalização é que os casos de treinamento sejam um subconjunto suficientemente grande e representativo (amostra em termos estatísticos) do conjunto de todos os casos que se quer generalizar (a população, em termos estatísticos). A importância desta condição está relacionada ao fato de que há, estritamente falando, dois tipos diferentes de generalização: interpolação e extrapolação. Interpolação aplica-se a casos que são mais ou menos "vizinhos" dos casos utilizados para treinamento, qualquer caso diferente deste pode ser considerado extrapolação. Em particular, casos que estejam fora do conjunto de treinamento, requer extrapolação. Casos de dados dentro de grandes lacunas no conjunto de dados de treinamento também podem

requerer extrapolação. Interpolação pode ser quase sempre feita de forma garantida, mas extrapolação é notoriamente sem garantia.

Assim, para uma função entrada-saída que seja convenientemente bem comportada, um caso teste que esteja próximo a algum caso usado para treinamento, terá uma saída correta. Se se tem uma amostra adequada para o conjunto de treinamento, todo caso na população estará bem próximo de um dos utilizados no treinamento. Consequentemente, sob estas condições, e com um treinamento adequado, a rede poderá generalizar convenientemente para a população.

Se se tem mais informação sobre a função, por exemplo, que a saídas deveria ser linearmente relacionadas à entradas, pode-se freqüentemente tirar vantagem desta informação colocando-se restrições na rede ou ajustando-se por um modelo específico, tal como um modelo linear, para melhorar a generalização. Extrapolação é muito mais confiável em modelos lineares que em modelos não lineares flexíveis, apesar de que não de modo tão seguro quanto a interpolação.

Ruídos e a Generalização: Ruídos em dados reais nunca são uma boa coisa, uma vez que eles limitam a precisão de generalização que pode ser conseguido sem problemas como extensão do conjunto de treinamento. Por outro lado, adicionar-se ruídos artificiais (*jitter*) nos valores de entrada durante o treinamento é um dos diversos caminhos para se melhorar a generalização para funções bem comportadas quando se tem pequenos conjuntos de treinamento.

Certas considerações sobre os ruídos são necessárias para resultados teóricos. Usualmente, a distribuição de ruídos é assumida como sendo de média zero e variância finita. O ruído em diferentes casos é assumido ser independente ou seguir algum modelo estocástico conhecido, tal como processo auto-regressivo, McCULLAGH e NELDER(1989).

Se se tem ruídos nos valores alvo, o erro quadrado médio da generalização pode nunca ser menor que a variância do ruído, não importando qual a quantidade de dados de treinamento que se tenha. Mas pode-se estimar a média dos valores alvo, condicional em um dado conjunto de valores de entrada, para qualquer desejado grau de precisão obtendo-se um conjunto de treinamento suficientemente grande e representativo, assumindo-se que a

função que se está querendo ajustar é do tipo apropriado para a rede que se está usando, e assumindo-se que a complexidade da rede é apropriada. Ruído nos valores alvo é exacerbado pelo *overfitting*.

Ruídos nos valores de entrada também limita a precisão da generalização, mas de uma maneira mais complicada que ruídos nos valores alvo. Em uma região do espaço de entrada onde a função que está sendo aprendida é razoavelmente monótona, ruídos à entrada terá pouco efeito. Em regiões onde a função é em degrau, ruídos podem degradar de forma irreversível a generalização.

Além do mais, se a função alvo é $\mathbf{Y = f(x)}$, mas se observa entradas com ruído, x+d, pode-se obter uma precisão arbitrariamente estimada de **f(x)**, dada x+d não importando o tamanho do conjunto de treinamento que se use. A rede não aprenderá **f(x)**, mas aprenderá, ao invés disto, a convolução de **f(x)** com a distribuição de ruído d.

Overfitting: Novamente, a questão crítica no desenvolvimento de uma rede neural é a generalização: como a rede neural poderá fazer bem as predições para os quais não foi ajustada? Redes neurais , como outros métodos de estimação não lineares tais como regressão *kernel* e *splines*, pode sofrer de subajuste (*underfitting*) ou de sobreajuste (overfitting). Uma rede que não seja suficientemente complexa, ou seja, tem menos parâmetros ajustáveis que o necessário, pode falhar para detectar totalmente um sinal em um conjunto de dados complicado, se a rede for treinada subajustadamente. Uma rede que seja muito complexa, com mais parâmetros ajustáveis que o necessário, pode ajustar o ruído, não apenas o sinal, levando a um sobreajuste. O sobreajuste é especialmente perigoso porque pode facilmente levar a predições que estão longe da faixa para as quais os dados foram treinados com muitas das redes neurais existentes.

Dada uma quantidade de padrões de treinamento, há no mínimo cinco considerações a serem feitas para se evitar ou o sobre ou o subajustamento, e consequentemente melhorar a generalização:

* **Seleção do modelo**

* ***Jittering***

* **Decaimento do peso**

* **Parada antecipada**

* **Aprendizagem Bayesiana**

A complexidade de uma rede neural está relacionada a ambos, o número de pesos e ao tamanho dos pesos. A seleção do modelo está preocupada com o número de pesos, e consequentemente o número de unidades ocultas e camadas. Quanto mais pesos há, relativamente ao número de casos de treinamento, mais o overfitting amplifica os ruídos nos alvos. Os outros itens listados estão preocupados, diretamente ou indiretamente, com o tamanho dos pesos. Todos estes itens podem ser encontrados detalhadamente nos trabalhos de AN(1996), NELSON e ILLIGWORWT(1991), BISHOP(1995), NEAL(1993a e b) e BUNTINE e WEIGEND(1991).

Número de Camadas Ocultas Necessárias: Pode ser que nenhuma camada intermediária seja necessária. Modelos lineares e modelos lineares generalizados são úteis em uma larga variedade de aplicações. Se a função que se quer treinar é medianamente não linear, pode-se conseguir melhor generalização com um modelo linear simples que com um modelo não linear complicado caso haja poucos dados ou muito ruído para estimar as não linearidades de maneira precisa.

Em MLP's com funções de ativação *Step/Threshold/Headviside*, necessita-se duas camadas ocultas para generalização completa SONTAG(1992) e BISHOP(1995).

Em MLP's com qualquer das grandes variações das funções de ativação contínuas não lineares , uma camada oculta com um número arbitrariamente grande de unidades é suficiente para uma aproximação universal de forma própria. Mas ainda se está buscando uma teoria que mostre, de forma analítica, quantas camadas ocultas são necessárias para aproximar uma determinada função.

Segundo SARLE(1997), se se tem somente uma entrada, não parece ser vantajoso usar mais que uma camada oculta. Mas as coisas se tornam muito mais complicadas quando se tem duas ou mais entradas.

Número de Unidades Ocultas Necessárias: Não há uma fórmula para se determinar uma boa arquitetura de rede a partir apenas do número de entradas e saídas. Esta tarefa depende do número de casos de treinamento, da quantidade de ruídos, e da complexidade da função ou da classificação que se está desejando fazer. Há problemas com milhares de entradas e milhares de saídas que requer milhares de unidades ocultas, e problemas com milhares de entradas e saídas e que requer somente uma unidade oculta, ou nenhuma.

Existem regras que se relacionam ao número de casos disponíveis dizendo para se usar o número de camadas ocultas tal que o número de pesos na rede vezes 10 seja menor que o número de casos. Tais regras estão somente preocupadas com sobreajustamento e parecem que não podem ser estendidas como regra geral confiável.

Uma escolha conveniente do número de camadas ocultas depende do fato de que se se está usando parada antecipada ou alguma outra forma de regularização. Se não, deve-se simplesmente tentar muitas redes com diferentes números de unidades ocultas, estimar a generalização do erro para cada uma, e escolher a rede com o mínimo erro generalizado estimado.

Usando algoritmos de otimização convencionais (ML, Conjugados), há pouca razão para se tentar usar uma rede com mais pesos que casos de treinamento, desde que tais redes provavelmente sobreajustarão. Mas LAWRENCE , GILLES e TSOI (1996), mostraram que o *backpropagation* padrão pode ter uma grande dificuldade para reduzir o erro de treinamento para um nível próximo a um valor ótimo, de forma que, usando-se sobreajuste, as redes podem reduzir ambos o erro de treinamento e o erro de generalização.

Quando se está usando parada antecipada, é essencial usar grandes lotes de unidades ocultas a fim de se evitar mínimos locais, SARLE(1995). O autor mostra que não parece haver um limite superior para o número de unidades ocultas, que não esteja relacionado ao tempo de processamento e à capacidade de armazenamento computacional, o que por outro lado, não quer dizer que haja muita vantagem em se usar mais unidades ocultas que o número de casos de treinamento, desde que o mínimo local não ocorre com muitas unidades ocultas.

Se se está usando decaimento de peso ou estimação Bayesiana pode-se usar muitas unidades ocultas NEAL(1995). Entretanto, não é estritamente necessário fazer assim, porque outros métodos estão disponíveis para se evitar mínimos locais, tais como múltiplas

partidas randômicas e *simulated annealing*, (SARLE(1995) diz que tais métodos não são seguros para se usar com a parada antecipada). Pode-se usar redes neurais com muitas unidades ocultas, ou então tentar diferentes redes com diferentes números de unidades ocultas, e escolher com base no erro de generalização de cada uma delas. Com decaimento de peso ou estimador MAP Bayesiano, deve-se conservar o número de pesos menor que a metade do número de treinamento.

Deve-se manter em mente que com duas ou mais entradas, uma MLP com uma camada oculta contendo somente algumas poucas unidades, pode ajustar somente uma limitada variedade de funções alvo. Mesmo superfícies simples, bem comportadas tais como Gausianas irregulares em duas dimensões podem requerer de 20 a 50 unidades ocultas para uma aproximação útil. ANN com um pequeno número de unidades ocultas freqüentemente produz falsos cumes e vales na superfície de saída. Treinar uma rede (típica) com 20 unidades ocultas requererá cerca de 150 a 2500 casos de treinamento se não se usar parada antecipada ou regularização.

Há vários resultados teóricos de como a dificuldade de aproximação de erros decresce quando o número de unidades aumenta, mas as conclusões são bastante sensíveis às considerações e é preciso se levar em conta a função que se está tentando aproximar. De acordo com o resultado de BARRON(1993), em uma rede com I entradas e H unidades em uma camada oculta simples, a raiz quadrada integrada do erro (RQIE) decrescerá, no mínimo, tão rápido quanto $H^{0.5}$ sob algumas considerações complicadas de 'alisamento' ou de aproximação.

Estimativa da Generalização do Erro: Há muitos métodos para se estimar a generalização do erro. Estatística de amostra simples: AIC,SBC, FPE, Mallows'C_p, etc. Em modelos lineares, a teoria estatística provê vários estimadores simples da generalização do erro sob várias considerações de amostragem. Estes estimadores ajustam o erro de treinamento para o número de erros que está sendo estimado, e em alguns casos para a variância do ruídos se a mesma é conhecida.

Estas estatísticas podem ser usadas como estimativas da generalização do erro em modelos não lineares quando se tem um grande conjunto de dados de treinamento. Corrigir estas estatísticas para não linearidades requer substancialmente mais computação, e a

teoria nem sempre é garantida para redes neurais devido às violações das condições de regularidade.

Entre os estimadores de generalização simplificada estão aqueles que não requerem que a variância do ruído seja conhecida, *Schwar's Bayesian Criterion* (SBC e BIC), RAFTERY(1995); e freqüentemente funciona bem para redes neurais, SARLE(1995). AIC e FPE tendem a sobreajustar com redes neurais.

EFRON(1986), considerando regressão logística, diz que para problemas de classificação, as fórmulas não são tão simples quanto para regressão com ruídos normais.

Validação dos Dados de Treinamento: O método mais comumente usado para estimar a generalização do erro em redes neurais é reservar parte dos dados como um conjunto de teste (divisão de amostras). O conjunto teste deve ser uma amostra representativa dos casos que se quer generalizar. Após o treinamento, utiliza-se o conjunto teste para fazer a generalização do erro. A desvantagem deste teste é que reduz a quantidade de dados que será utilizada no treinamento.

Validação cruzada e *bootstrapping* são, ambos, métodos para estimar a generalização do erro baseado na "re-amostragem", e *jackkniffing*, é o método o qual é usado para estimar o *bias* de uma estatística, EFRON(1983), MASTERS(1995), TIBSHIRANI(1996) e SARLE(1995). As estimativas resultantes de generalização de erro são freqüentemente usados para escolher dentre os vários modelos, redes de diferentes arquiteturas.

A forma de se escolher dentre os métodos de validação cruzada podem ser vistos em SHAO(1993) e SARLE(1997).

Pré-Processamento Extração de Característica: Em muitas aplicações práticas a escolha e o uso do pré-processamento será um dos fatores mais significantivos na determinação do desempenho final do sistema.

No caso mais simples, o pré-processamento pode tomar a forma de uma transformação linear dos dados de entrada, e possivelmente, também dos dados de saída (onde passa a se chamar pós-processamento). Pré-processamentos mais complexos podem

envolver a redução da dimensão dos dados de entrada. O fato de que tal redução pode levar à melhora no desempenho pode em princípio parecer paradoxal, uma vez que não se pode aumentar a quantidade de informações fornecida à rede, que se quer treinar, retirando-se padrões de treinamento.

Um outro caminho através do qual a rede neural pode ter seu desempenho melhorado, por vezes até de forma considerável, é através da incorporação do conhecimento prévio, o qual se refere a informações relevantes. O conhecimento prévio pode ser ou incorporado à estrutura da rede ou utilizado no pré ou pós-processamento. Pode também ser usado para modificar o processo de treinamento através do uso da regularização.

Um aspecto final da preparação dos dados surge do fato de que os dados reais freqüentemente sofrem de um número de deficiências tais como falta de valores de entrada ou valores alvo incorretos.

O pré-processamento é, para muitas aplicações práticas, o estágio mais importante no desenvolvimento da solução, e a escolha da forma de pré-processamento pode ter significantivo efeito na generalização da rede treinada. E dentre as formas de pré-processamento, uma das mais importantes é a redução da dimensionalidade dos dados de entrada. De forma mais simples pode envolver apenas o descarte de alguns valores de treinamento, caso estes apresentem informações redundantes, de pouca utilidade para a resolução do problema em questão, ou caso a mesma informação seja repetida em mais que uma variável. De forma mais elaborada, pode-se processar combinações lineares ou não das variáveis originais a fim de se formar novas entradas para a rede neural a ser treinada. Este último procedimento é o que se pode, numa tradução literal, considerar como extração de característica do conjunto de dados.

Em algumas situações, a redução da dimensionalidade, resultará em perda de informações, e desta forma, uma das metas principais das estratégias de pré-processamento é assegurar que a maioria das informações relevantes seja retida pelo conjunto a ser utilizado para o treinamento das redes, atentando-se sempre para não sobrecarregar o procedimento de resolução com procedimentos computacionais exaustivos. Isto leva à necessidade de, em alguns casos, ser preciso reter maior número de características para se assegurar que não se está descartando muitas informações úteis. Para todos estes

procedimentos vale ter em mente que, aqui, como em todo o conjunto de soluções utilizando redes neurais, cada sistema é um sistema, não se podendo portanto indicar uma fórmula definitiva capaz de solucionar todos os casos. O teste da validação sempre pode ser empregado para se ter idéia das transformações ocorridas nos resultados segundo cada uma desta modificações no conjunto de padrões de entrada anunciadas até aqui.

Uma das formas mais comuns de pré-processamento consiste de um re-escalonamento linear dos dados de tal forma a deixá-los todos em uma mesma ordem de grandeza, o que deixa todos os dados com a mesma importância absoluta. Por exemplo, se se tem dados de temperatura e de frações molares, é possível tomá-los como variando de 0 a 1, sendo que o menor valor é zero e o maior 1. Desta forma, tem-se um conjunto de dados onde todos os dados se encontra num faixa bem definida e coerente de valores. Este procedimento possibilita a utilização de pesos iniciais randômicos, o que de outra forma não seria conveniente, necessitando-se ter um procedimento de iniciação dos pesos de forma marcadamente diferente. Outra forma de linearização mais sofisticada também pode ser utilizada, conforme a apresentada em FUKUNAGA(1990).

Logicamente, depois de treinada, é preciso desfazer a linearização dos dados utilizando-se a mesma regra inicial.

Ao contrário do excesso de padrões, o que pode ocorrer é de estarem faltando dados para o modelo de treinamento da rede neural. Também para este caso, existem heurísticas para contornar o problema. Por exemplo pode-se recolocar o valor que esteja faltando por um valor médio dos valores na região do dado que falta. De forma mais elaborada, pode-se fazer uma regressão utilizando os dados disponíveis na região que apresenta falhas de preenchimento. Estatísticamente falando, a utilização do algoritmo EM (de expectativa máxima do valor) pode oferecer melhores resultados, GHAHRAMANI e JORDAN(1994).

Os assuntos descritos neste capítulo fazem parte do conjunto principal de conceitos que envolvem a teoria a respeito de redes neurais. A maior parte dos temas aqui abordados estão implementados no programa computacional desenvolvido neste trabalho ou fornecem subsídios para um melhor entendimento do funcionamento das redes neurais. Quando mais que uma opção para o mesmo conceito podia ser feita, testou-se ambas a fim de se determinar qual a mais conveniente. A descrição do algoritmo bem como as

considerações adotadas para estudo e testes pode ser conferida no próximo capítulo, onde se encontra o detalhamento e a descrição das equações que regulam o modelo adotado.

CAPÍTULO 6

6.0 DESCRIÇÃO DO MODELO DA REDE NEURAL IMPLEMENTADA

A meta na construção de dispositivos capazes de aprender utilizando redes neurais é projetar sistemas que sejam robustos, confiáveis e poderosos. Não há interesse em se criar instrumentos de aprendizagem que sejam fracos, e de baixa eficiência. A generalização é uma das características mais desejadas quando se projeta uma rede neural, e isto implica poder reconhecer, determinar ou classificar corretamente um determinado padrão sem que se tenha que conhecer suas características na hora do treinamento.

Seja implementado em um *hardware* paralelo ou simulado em um *software*, toda rede neural consiste de uma coleção de elementos ou funções básicas que trabalham juntas para resolver problemas.

Neste capítulo estão descritas as bases teóricas para implementação do programa de treinamento de redes neurais desenvolvido neste trabalho. A forma como o mesmo está organizado está detalhada no próximo capítulo, o de resultados. Anexo, no Apêndice C, está a listagem contendo o código fonte do programa de treinamento de redes neurais.

6.1 BACKPROPAGATION E SUAS VARIANTES

Backpropagation foi um dos primeiros algoritmos de treinamento de rede que apareceram. Depois de estudos ainda na década de 40, McCULLOCH e PITTS(1943), e de trabalhos como os de COVER(1960) e depois de BAUM(1988), WERBOS(1974), foi desenvolvido o algoritmo de treinamento com *backpropagation*. Mas foi Rumelhart e seus colaboradores quem 'redescobriu' a técnica que havia ficado esquecida por algum tempo, RUMELHART et al.(1986).

O algoritmo conforme foi criado, quase não tem utilidade prática atualmente devido à sua baixa eficiência em termos de capacidade de treinamento. O conhecimento de

como foi desenvolvido pode, no entanto, ser um bom começo para que se possa entender a teoria por trás desta técnica de treinamento de redes neurais.

Existem muitos trabalhos em que se encontram detalhadas as características e melhoramentos possíveis para as redes neurais. Neste trabalho implementou-se um algoritmo básico de treinamento utilizando o *backpropagation* e foi-se acrescentando os diversos melhoramentos surgidos durante o tempo. Como o programa que utilizou este algoritmo não foi o mais eficiente, e também pelo fato de se encontrar na literatura uma vasta quantidade de material acerca deste tipo de algoritmo, não será aqui detalhado o seu modelo. Dentre os trabalhos consultados, alguns trazem significativa descrição de modelos e modificações úteis. Pode-se citar dentre eles, além dos já citados, os trabalhos de COURANT e HIBERT(1962), McCLELLAND(1986), LIPPMANN(1987), FAHLMAN(1988), POGGIO e GIROSI(1989), DARKEN e MOODY(1992), LeCUN et al.(1993), RIEDMILER e BAUM(1993), RIEDMILER(1994), BERTSEKAS e TSITSIKLIS(1996), KUSHNER e YIN(1997) e ORR e LEEN (1997).

Para o tipo de aprendizagem que seja possível de se implementar, pode ser visto no trabalho de RIEDMILLER(1994), a forma de aprendizagem por cada padrão em comparação a uma forma de aprendizagem por conjunto de padrões. Para a inicialização dos pesos da rede neural, podem ser consultados os trabalhos de KOLEN e POLLACK(1990), o trabalho de THIMM e FIESLER(1997) e também o trabalho de PLATT(1991), sendo que este último é específico para inicialização de pesos de rede com algoritmo RBF.

6.2 FUNÇÕES BASE RADIAIS (RBF)

Apesar de, como no caso do algoritmo utilizando *backpropagation*, existir na literatura bastante material acerca das redes neurais utilizando o algoritmo com RBF, o mesmo será aqui descrito devido à quantidade de modificações que foi introduzida no algoritmo base, sugerida em trabalhos citados ao longo do capítulo.

Redes com funções base radiais pertencem ao grupo de funções que utilizam funções *kernel*, distribuídas em diferentes vizinhanças do espaço de entrada, cujas respostas são essencialmente de natureza local, ROY et al. (1997). Foram inicialmente propostas por MOODY e DARKEN(1989). Eles se inspiraram em neurônios localmente sintonizados ou em neurônios seletivos os quais têm uma resposta para alguma faixa de variáveis de entrada.

Radial Basis Function, RBF, é simplesmente uma classe de funções simétricas definidas em $\Re^n \rightarrow \Re$ para o caso de funções monovariáveis ou $\Re^n \rightarrow \Re^{n'}$, para funções múltivariáveis geralmente descritas como

$$\phi(\mathbf{x}) = \phi(\|\mathbf{x} - \mathbf{c}\|); \quad \mathbf{x}, \mathbf{c} \in \Re^n; \ \mathbf{x} \geq \mathbf{0};$$

Em princípio, podem ser empregadas em qualquer tipo de modelo (linear e não linear) e em qualquer tipo de rede (camada simples ou múlticamada), Figura (6.1). Esta é uma outra classe de redes neurais na qual a ativação das unidades ocultas é feita pela distância entre o vetor de entrada e o vetor protótipo, ou núcleo. Seu uso é possível do ponto de vista de aproximação de funções, interpolação com ruídos, teoria da classificação ótima e funções potenciais.

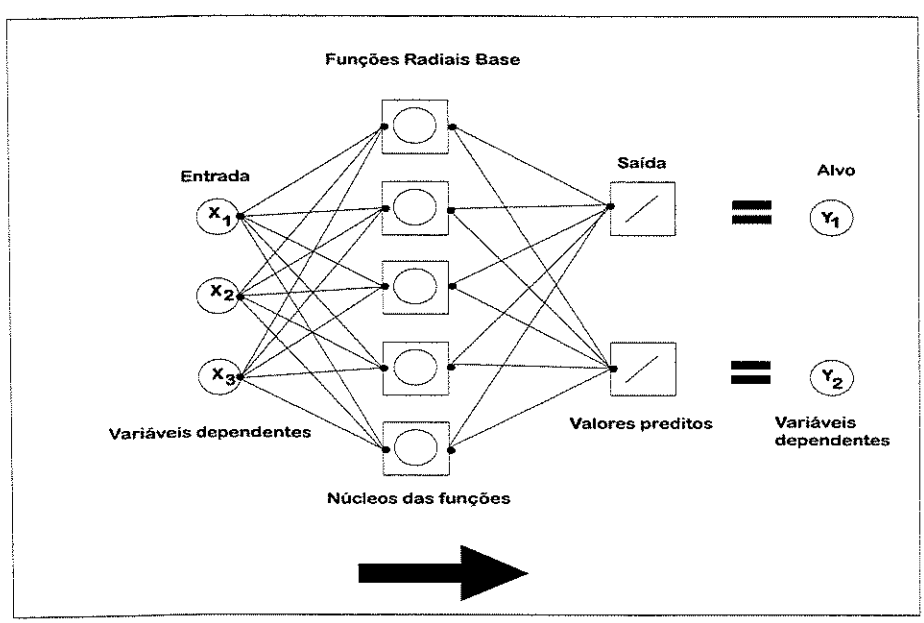


Figura (6.1): Representação de uma rede neural com RBF.

Segundo BISHOP(1995), uma propriedade interessante e importante destas redes neurais com RBF é que ela forma uma ligação entre um grande número de conceitos díspares, o que como conseqüência possibilita procedimentos para treinamento de redes neurais (com RBF) os quais podem ser muito mais rápidos que os métodos usados para treinar *perceptrons* de múlti-camadas. Isto segue da interpretação que pode ser dada para as representações internas formadas pelas unidades ocultas, e leva a um procedimento em dois estágios. No primeiro estágio, os parâmetros que governam as funções base (correspondentes às unidades ocultas) são determinados, usando métodos não-supervisionados (métodos que usam apenas os dados de entrada e não os de saída, capítulo 5) relativamente rápidos. O segundo estágio de treinamento envolve a determinação dos pesos da camada final, o qual requer a solução de um problema linear, que é mais rápido.

RBF tem suas origens nas técnicas para interpolação exata de um conjunto de dados em um espaço múlti-dimensional, POWEL(1987). Sua característica é que a resposta decresce (ou cresce) monotonicamente com a distância do ponto central, ou kernel. O centro, a escala da distância, e a forma precisa da função radial são parâmetros do modelo, todos fixos se for linear. A interpolação exata requer que todo vetor de entrada seja mapeado exatamente sobre o vetor de saída, e isto forma o ponto de partida para a discussão das redes RBF.

Segundo FRITZKE(1994), a largura da função base é o mesmo que o desvio padrão e para o caso de a mesma ser Gausiana, a largura para cada unidade é ajustada pelo comprimento médio de todas as pontes que saem dela. A idéia por trás disto é dar a cada unidade um "campo receptivo" no espaço do vetor de entrada com sobreposições limitadas a uma determinada extensão em relação às vizinhanças.

Segundo KAMINSKI e STRUMILLO(1997), um dos maiores problemas em se projetar uma rede neural com RBF é escolher de forma apropriada o número de funções radiais. Se se escolhe muito poucos centros pode-se ter uma aproximação pobre ao passo que um conjunto muito grande de centros pode causar sobreajustamento o que também é ruim para os propósitos de generalização.

6.2.1 CASO DE SAÍDA UNIDIMENSIONAL

Considere-se o mapeamento de um vetor **d**-dimensional **x** em um vetor unidimensional **t**. O conjunto de dados consiste de N vetores de entrada $\mathbf{x}^n$, juntos com os correspondentes alvos $\mathbf{t}^n$. A meta é encontrar uma função **h(x)** tal que

$$\mathbf{h}(\mathbf{x}^n) = \mathbf{t}^n,\ n=1,...,N \tag{6.1}$$

esta restrição indica que a função ajustada deve passar exatamente pelos pontos dados. Existem outros critérios os quais podem ser impostos a **h(x)** de tal forma a restringir a forma da função, FRANKE(1982).

O método RBF, POWEL(1987), introduz um conjunto de N funções base, uma para cada padrão de entrada, o qual toma a forma $\phi(\|\mathbf{x}-\mathbf{x}^n\|)$ onde $\phi(\bullet)$ é alguma função não linear, chamada de função base em analogia ao conceito de vetores sendo compostos por uma combinação de vetores base. Deve ser preferencialmente bem comportada (diferenciável). Assim, essa n-ésima função depende da distância $\|\mathbf{x}-\mathbf{x}^n\|$, usualmente tomada como sendo Euclidiana, entre o valor **x** e $\mathbf{x}^n$. A saída do mapeamento é tomada como sendo uma combinação linear das funções base

$$\mathbf{h}(\mathbf{x}) = \sum_{n=1}^{N} \mathbf{w}_n \phi(\|\mathbf{x}-\mathbf{x}^n\|) \tag{6.2}$$

A flexibilidade de **h(x)**, sua habilidade para ajustar muitos diferentes tipos de funções, deriva da liberdade de escolha dos diferentes valores dos pesos. As funções base e quaisquer parâmetros que as mesmas possam conter, são fixos, para o caso linear e se podem variar durante o processo, o modelo é não linear. Quanto à flexibilidade, é preciso que se diga que um modelo, se for muito flexível poderá se tornar inconveniente se ajustar

qualquer tipo de ruído; por outro lado, quanto mais inflexível pior será para se atingir os valores desejados de saída.

As condições de interpolação podem ser escritas então na forma de matriz como

$$\Phi \mathbf{w} = \mathbf{t} \tag{6.3}$$

onde $\mathbf{t} = (\mathbf{t}^n)$, $\mathbf{w} = (\mathbf{w}_n)$, e a matriz quadrada Φ tem elementos $\Phi_{nn'} = \phi\left(\left\|\mathbf{x}^n - \mathbf{x}^{n'}\right\|\right)$.

Existindo a inversa Φ^{-1}, se for quadrada e não singular e definida estritamente positiva, MICCHELLI(1986), pode-se resolver (6.3) para dar

$$\mathbf{w} = \Phi^{-1}\mathbf{t} \tag{6.4}$$

MICHELLI(1986) mostrou que para um grande número de classes de funções $\phi(\bullet)$, a matriz Φ é realmente não singular levando-se em conta que os pontos são distintos. A teoria de MICHELLI se aplica a muitas escolhas de Φ, incluindo-se as funções lineares. Se $\Phi(\mathbf{x}) = \mathbf{x}^2$, então a função (6.2) é necessariamente uma função quadrática em **x**, tal que Φ é singular se **d** excede a dimensão do espaço das polinomiais quadráticas, a qual é 0.5(d+1)(d+2). Quando os pesos em (6.2) são ajustados pelos valores dados em (6.4), a função **h(x)** representa uma superfície contínua e diferenciável a qual passa exatamente através de cada ponto dado.

Existem diversas formas de modelos lineares cujas funções base são polinomiais ou variações das mesmas. Combinações de ondas *sinusoidais* (séries de Fourier)

$$\phi(\mathbf{x}) = \mathbf{sen}\left(\frac{2\pi(\mathbf{x}-\theta)}{\mathbf{M}}\right)$$

são freqüentemente usadas em processamento de sinais. Funções logísticas, do tipo

$$\phi(\mathbf{x}) = \frac{\mathbf{1}}{\mathbf{1} + \mathbf{exp}\left(\mathbf{b}^{T}\mathbf{x} - \mathbf{b}_{0}\right)},$$

são utilizadas em redes neurais com *perceptrons* de múlti camadas.

Das diversas formas de funções base já consideradas, a mais comum é a função Gaussiana, BROOMHEAD e LOWE(1988), MOODY e DARKEN(1989)

$$\phi(\mathbf{x}) = \mathbf{exp}\left(-\frac{\mathbf{x}^{2}}{2\sigma^{2}}\right). \tag{6.5}$$

Conforme descrito anteriormente, PLATT(1991) utiliza uma aproximação para inicialização dos pesos da primeira camada e diz que o desempenho da rede melhora. No programa implementado neste trabalho pode-se comprovar que ainda que não seja de forma tão acentuada, a melhora pode realmente ser observada. Deste modo, a modificação proposta pelo autor é

$$\phi(\mathbf{x}) = \begin{cases} \left(\mathbf{1} - \left(\mathbf{x}/\mathbf{q}\sigma^{2}\right)\right)^{2} & \textbf{se } \mathbf{x} < \mathbf{q}\sigma^{2} \\ \mathbf{0} & \textbf{de outra forma.} \end{cases} \tag{6.5a}$$

onde **q=2,67** foi escolhido empiricamente para se fazer o melhor ajuste Gaussiano.

A Figura(6.2a) mostra a função Gaussiana, a qual é uma função *kernel* bastante popular, com três diferentes tamanhos de σ, onde o parâmetro σ controla as propriedades de "alisamento" da função de interpolação. A Gaussiana (6.5) é uma função base localizada[33] com a propriedade de que $\phi \to \mathbf{0}$ quando $|\mathbf{x}| \to \infty$. Desta forma, tem-se que a

[33] O termo se refere ao fato de que a função Gaussiana ser uma função que dá uma resposta significativa somente nas vizinhanças do centro.

função Gaussiana RBF decresce monotonicamente conforme se distancia do centro. Uma outra função base com a mesma propriedade é

$$\phi(\mathbf{x}) = \left(\mathbf{x}^2 + \sigma^2\right)^{-\alpha}, \ \alpha > \mathbf{0} \tag{6.6}$$

a qual é uma equação múltiquádrica (MQE) inversa, (Fig6.2 c).

Não é, entretanto, necessário que a função base seja localizada de forma que outras possíveis escolhas são as funções *spline thin-plate* (TPS), devidas a DUCHON(1976) e MEINGUEST(1979), as quais são funções derivadas de métodos variacionais, Figura(6.2d)

$$\phi(\mathbf{x}) = \mathbf{x}^2 \log(\mathbf{x}). \tag{6.7}$$

A função

$$\phi(\mathbf{x}) = \left(\mathbf{x}^2 + \sigma^2\right)^{\beta}, \ \mathbf{0} < \beta < \mathbf{1} \tag{6.8}$$

para a qual $\beta = \mathbf{0,5}$ é conhecida como função múlti-quadrática;

A pseudo-cúbica (PC) tem a forma

$$\phi(\mathbf{x}) = \mathbf{x}^3, \tag{6.9}$$

mostrada na Figura(6.2d) e foi desenvolvida por DUCHON(1976).

E a função "linear"

$$\phi(\mathbf{x}) = \mathbf{x} \tag{6.10}$$

a qual têm a propriedade de que $\phi \to \infty$, quando $\mathbf{x} \to \infty$.

Notar que, na função (6.10), $\mathbf{x} = \left\| \mathbf{x} - \mathbf{x}^n \right\|$ e por isso ainda é não-linear em relação aos componentes de **x**. Em uma dimensão, representa a forma mais simples de interpolação, e não é suficientemente flexível para ser utilizada em aprendizagem supervisionada.

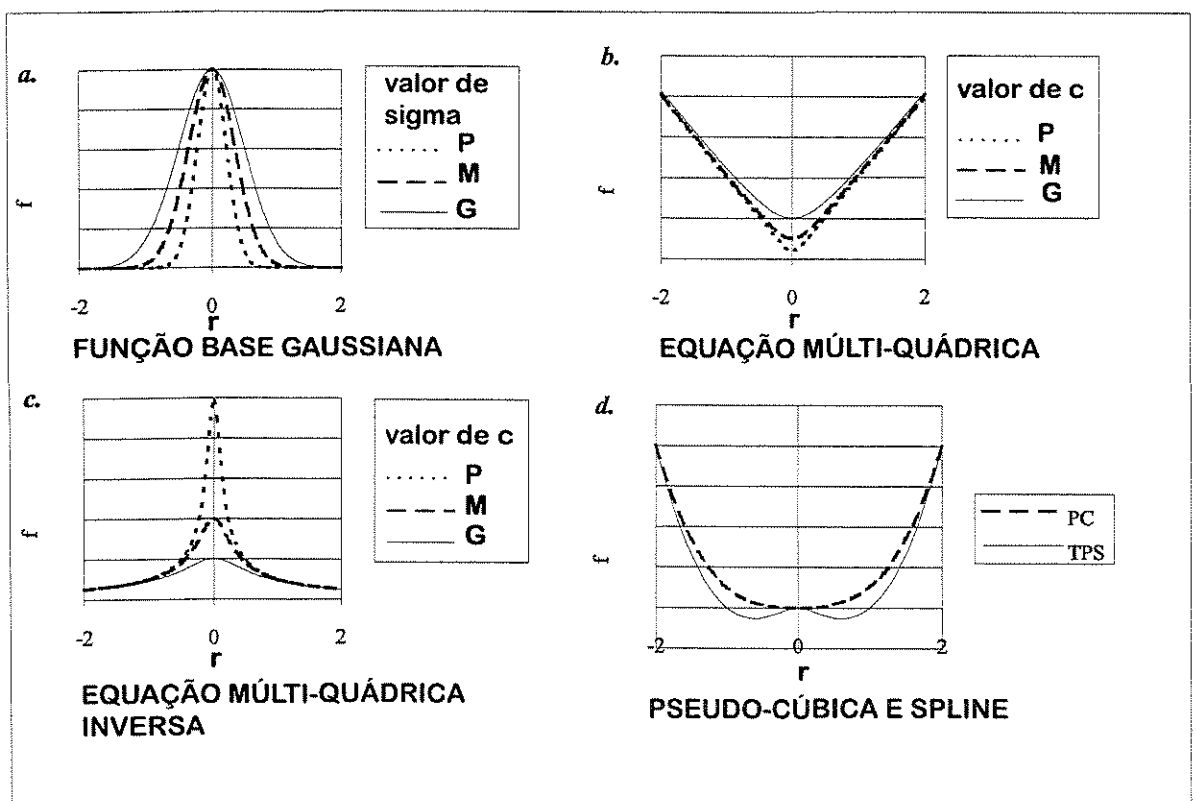


Figura (6.2): Formas das cinco RBF mais comumente usadas.

Segundo CHEN et al.(1991), no contexto de redes neurais, existem razões para se considerar funções base localizadas. As funções Gaussianas serão priorizadas aqui porque além de serem localizadas, têm um grande número de propriedades analíticas úteis[34].

[34] Funções bem comportadas, diferenciáveis.

6.2.2 CASO DE SAÍDA MÚLTI-DIMENSIONAL

A generalização para diversas variáveis é direta. Cada vetor de entrada $\mathbf{x}^n$ deve ser mapeado exatamente sobre o vetor de saída $\mathbf{t}^n$ o qual tem componentes $\mathbf{t}_k^n$ tal que a função (6.1) se torna

$$\mathbf{h}_k(\mathbf{x}^n) = \mathbf{t}_k^n \text{ , } n = 1, \ldots, N \tag{6.11}$$

onde $\mathbf{h}_k(\mathbf{x})$ é obtida por superposição das mesmas N funções base como para o usado no caso unidimensional,

$$\mathbf{h}_k(\mathbf{x}) = \sum_{n=1}^{N} \mathbf{w}_{kn} \phi\left(\left\|\mathbf{x} - \mathbf{x}^n\right\|\right) \tag{6.12}$$

(notar que neste caso existe um conjunto de pesos para cada ajuste). Os pesos são, por analogia, obtidos como em (6.4)

$$\mathbf{w}_{kn} = \sum_{n} \left(\Phi^{-1}\right)_{nn'} \mathbf{t}_k^{n'} . \tag{6.13}$$

Notar que em (6.13) a mesma matriz Φ^{-1} é usada para cada uma das funções de saída.

A partir das definições acima, onde se mostra a utilização das funções base para o mapeamento de uma função qualquer, pode-se depreender que o método da função base radial constrói a solução para o problema da interpolação como uma combinação linear das funções base lineares.

6.2.3 REDES NEURAIS COM RBF

Redes com RBF são redes neurais com camadas de neurônios ou nós em *feedforward* (Figura(6.1)). Os nós da camada de entrada simplesmente propagam os valores para a camada oculta , onde estão totalmente conectados. As conexões entre a camada de entrada e a camada oculta especifica o conjunto dos centros das funções. Os centros das funções podem ser fixados antes do treinamento da rede ou então modificados durante o treinamento da mesma. Para cada nó oculto determina-se uma distância entre eles e o vetor de entrada, denotado pela norma Euclidiana, e então realiza-se uma transformação nesta distância calculada. Cada nó oculto é totalmente conectado aos nós de saída. Os pesos entre a camada oculta e a camada de saída são determinados durante o treinamento da rede. Cada nó de saída da rede determina uma soma ponderada das saídas dos nós da camada oculta.

Muitas das vantagens dos modelos de redes neurais adaptativos estão contidas nos aspectos não lineares das unidades ocultas tal que as relações não lineares de entrada-saída possam ser modeladas. Infelizmente, uma vez que o critério de erro depende da resposta de elementos não lineares, o problema de encontrar uma solução ótima global torna-se um problema de minimização quadrada, os quais podem ser resolvidos, usualmente, por técnicas iterativas. Por exemplo, por propagação reversa do erro, a qual sofre do defeito de se poder encontrar antes da solução global, uma solução local, o que muitas vezes nem mesmo com um bom conjunto de partida pode ser contornado.

Uma rede com RBF é muito similar em arquitetura às redes com MLP treinadas com algoritmo *backpropagation*. A maior diferença é a função de ativação na camada oculta. Redes com MLP usam como função de ativação a função *sigmoidal*, redes com RBF usam a função base radial como função de ativação. Outra notável diferença entre os dois tipos de redes é o número de nós da camada oculta. Se ambas são utilizadas para resolver o mesmo problema, a rede com RBF terá muito mais nós que a rede com MLP. Especificar uma rede com RBF para resolução de um problema é equivalente a especificar o número de nós em cada uma das camadas. O número de nós de entrada para as camadas 'externas' é fácil de se determinar. Por exemplo, quando se quer fazer uma interpolação de dados, o número de nós de entrada é igual ao número de variáveis independentes, e o número de nós de saída é igual ao número de variáveis dependentes. Para problemas de classificação, o

número de nós de entrada é usualmente igual ao número de classes. A seleção do número de nós das camadas ocultas e os centros das funções é muito mais flexível e muito mais difícil por depender de alguns fatores externos como por exemplo a capacidade de processamento que se dispõe para efetuar os cálculos.

Os mapeamentos discutidos para o RBF proporciona uma função de interpolação a qual passa exatamente através de todos os pontos dados, exceto os desvios de aproximação, BISHOP(1995). Uma limitação para o procedimento de interpolação exata é o número de funções base, que neste caso, é igual ao número de padrões no conjunto de dados, e assim para grandes conjuntos de dados, a função de mapeamento se torna muito difícil de ser determinada.

Introduzindo-se um número de modificações no procedimento exato de interpolação obtém-se a rede neural com RBF dada por BROOMHEAD e LOWE(1988) e também por MOODY e DARKEN(1989)). Estes trabalhos descrevem uma interpolação mais "suave" na qual o número de funções base é determinado pela complexidade do mapeamento a ser processado ao invés do tamanho do conjunto de dados; perde-se assim um pouco da flexibilidade do método mas ganha-se em tempo de processamento e diminuição das dificuldades computacionais, a interpolação, desta forma deixa de ser exata.

As modificações requeridas são:

1. O número M de funções base não necessita ser igual ao número de N pontos dados, e tipicamente é muito menor que N.
2. Os centros das funções base são restringidos para serem dados pelos padrões do vetor de entrada. Desta forma, a determinação dos centros convenientes se torna parte do processo.
3. Em lugar de ter uma largura comum para o parâmetro σ, a cada função base é dado o seu próprio valor de σ_j o qual é também determinado durante o treinamento.
4. Parâmetros de *bias* são incluídos na soma linear. Eles compensam a diferença entre o valor médio dos dados da função de ativação base e os valores médios dos alvos correspondentes.

O problema da interpolação pode ser colocado quando se tem **m** vetores distintos, $\{\mathbf{x}_i ; \mathbf{i} = 1,2,...,\mathbf{N}\}$ em $\Re^n$ e **N** números reais $\{\mathbf{y}_i ; \mathbf{i} = 1,2,...,\mathbf{N}\}$ na forma $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N}$, como o de encontrar uma função $\mathbf{Y} : \Re^n \rightarrow \Re$ a qual satisfaz as condições de interpolação

$$\mathbf{Y}(\mathbf{x}_i) = \mathbf{y}_i ; \qquad \mathbf{i} = 1,2,...,\mathbf{N}$$

a função **Y** é definida de tal forma que passa através de todos os pontos.

Quando estas mudanças são feitas na fórmula de interpolação exata (6.12), chega-se à seguinte forma para o mapeamento da rede neural com RBF

$$\mathbf{y}_k(\mathbf{x}) = \sum_{j=1}^{M} \mathbf{w}_{kj} \phi_j(\mathbf{x}) + \mathbf{w}_{k0} \tag{6.14}$$

Para o caso da função base ser Gaussiana, tem-se

$$\phi_j(\mathbf{x}) = \mathbf{exp}\left(-\frac{\|\mathbf{x} - \mu_j\|^2}{2\sigma_j^2} \right) \tag{6.15}$$

onde **x** é o vetor de entrada **d**-dimensional com elementos $\mathbf{x}_i$ e μ_j é o vetor que determina o centro da função base ϕ_j e tem elementos μ_{ji}. Notar que as Gaussianas em (6.15) não são normalizadas. Este mapeamento pode ser representado como uma rede neural mostrado na Figura (6.1).

Um esquema de aproximação tem a propriedade de, em um conjunto de funções aproximação (i.e. o conjunto de funções correspondentes a todas às escolhas dos

parâmetros ajustáveis) há uma função a qual tem um erro de aproximação mínimo para qualquer função dada ser aproximada.

A função RBF Gaussiana considerada acima pode ser generalizada para permitir matrizes covariância arbitrárias Σ_j. Assim tomando-se as funções base a forma será

$$\phi_j(\mathbf{x}) = \exp\left\{-\frac{1}{2}(\mathbf{x}-\mu_j)^T \Sigma_j^{-1}(\mathbf{x}-\mu_j)\right\} \tag{6.16}$$

Uma vez que as matrizes covariância são simétricas, isto significa que cada função base tem **d(d+3)/2** parâmetros independentes ajustáveis (onde d é a dimensão do espaço de entrada), e pode ser comparada aos **(d+1)** parâmetros independentes para as funções base em (6.15). Na prática há um *trade off* a ser considerado entre o uso de um pequeno número de bases com muitos parâmetros ajustáveis e um grande número de funções menos flexíveis.

Um aspecto chave das redes neurais com RBF é a distinção entre os papéis dos pesos da primeira e da segunda camadas. Como será visto, as funções base podem ser interpretadas de uma forma que permite aos pesos da primeira camada (i.e. parâmetros que governam as funções base) serem determinados através de técnicas de treinamento não supervisionado. Isto leva a um procedimento, em dois estágios, para as redes neurais com RBF. No primeiro estágio o conjunto de dados de entrada $\{\mathbf{x}^n\}$, sozinho, é usado para determinar os parâmetros das funções base (por exemplo μ_j e σ_j para a função Gaussiana esférica considerada acima). As funções base são então conservadas fixas enquanto os pesos da segunda camada são encontrados na segunda fase do treinamento. Técnicas para otimizar os parâmetros das funções base podem ser vistas em BISHOP(1995). Para efeitos de demonstração da metodologia, será assumido que os parâmetros já tenham sido escolhidos, e será discutido o problema de otimização dos pesos da segunda camada. Notar que se existe menor número de funções base que os pontos dados, então em geral, não será possível encontrar um conjunto de valores para os quais a função de mapeamento ajusta os dados exatamente (interpolação não exata).

Começa-se considerando-se o mapeamento de rede neural com RBF dado em (6.14) e considerando o *bias* incluído nos pesos dados, tem-se

$$\mathbf{y}_k(\mathbf{x}) = \sum_{j=0}^{M} \mathbf{w}_{kj}\phi_j(\mathbf{x}) \tag{6.17}$$

onde ϕ_0 é uma função base 'extra' com valor de ativação fixado em 1, $\phi_0 = \mathbf{1}$. Isto pode se escrito em notação matricial como

$$\mathbf{y}(\mathbf{x}) = \mathbf{W}\phi \tag{6.18}$$

$$\begin{pmatrix} \mathbf{y_1} \\ \mathbf{y_2} \\ \mathrm{M} \\ \mathbf{y_m} \end{pmatrix} = \begin{pmatrix} \mathbf{a_{11}} & \mathbf{a_{12}} & \Lambda & \mathbf{a_{1m}} \\ \mathbf{a_{21}} & \mathbf{a_{22}} & \Lambda & \mathbf{a_{2m}} \\ \mathrm{M} & \mathrm{M} & \mathrm{O} & \mathrm{M} \\ \mathbf{a_{m1}} & \mathbf{a_{2m}} & \Lambda & \mathbf{a_{mm}} \end{pmatrix} \begin{pmatrix} \mathbf{w_1} \\ \mathbf{w_2} \\ \mathrm{M} \\ \mathbf{w_m} \end{pmatrix}$$

onde

$$\mathbf{a_{ij}} = \Phi\left(\left\|\mathbf{x_i} - \mathbf{c_j}\right\|\right) \qquad \mathbf{i, j = 1,2,...,m}$$

e

$\mathbf{W} = (\mathbf{w}_{kj})$ e $\phi = (\phi_j)$.

Desde que as funções base sejam conservadas fixas, a rede é equivalente à rede de camada simples utilizando discriminante linear. Se as funções base puderem se mover ou mudar de tamanho ou se há mais que uma camada oculta, a rede com RBF é não linear. Pode-se otimizar os pesos minimizando-se através de uma função erro conveniente. Uma escolha particularmente conveniente é considerar a soma dos quadrados do erro dada por

$$\mathbf{E} = \frac{1}{2}\sum_{n}\sum_{k}\left\{\mathbf{y}_k\left(\mathbf{x}^n\right) - \mathbf{t}_k^n\right\}^2 \tag{6.19}$$

onde $\mathbf{t}_k^n$ é o valor alvo para a unidade de saída k quando se alimenta a rede com os valores do vetor de entrada $\mathbf{x}^n$. Desde que a função erro é uma função quadrática dos pesos, seu mínimo pode ser encontrado em termos da solução de um conjunto de equações lineares, BISHOP(1995)

$$\Phi^T\Phi\mathbf{W}^T = \Phi^T\mathbf{T} \tag{6.20}$$

onde $(\mathbf{T})_{nk} = \mathbf{t}_k^n$

e

$(\Phi) = \phi_j\left(\mathbf{x}^n\right)$.

A solução formal para os pesos é dado por

$$\mathbf{W}^T = \Phi^{\perp}\mathbf{T} \tag{6.21}$$

onde a notação $\Phi^{\perp}$ denota a pseudo inversa de Φ. Na prática, as equações (6.20) são resolvidas usando-se SVD[35], para evitar problemas devido ao possível mau condicionamento da matriz Φ. Assim, pode-se ver que os pesos da segunda camada podem ser encontrados, rapidamente, através de técnicas de inversão de matrizes.

Na maior parte serão consideradas redes com RBF nas quais a dependência da função das redes na segunda camada serão lineares, e a função erro é dada pela soma dos quadrados. É possível considerar o uso de funções de ativação não lineares aplicados às

---

[35] Ver TEIXEIRA(1997).

unidades de saída, ou outras escolhas para a função erro. Entretanto, a determinação dos pesos da segunda camada não é um problema linear, e consequentemente é necessária uma técnica de otimização não linear. Uma das maiores vantagens das redes com RBF é o fato de ser possível evitar a necessidade de tais otimizações das redes durante o treinamento.

Uma vantagem da rede com RBF em relação às MLP[36] é que os parâmetros do modelo têm alguma interpretação em relação ao processo modelado. Os pesos entre a camada de entrada e a camada oculta representam ou os pontos dados no espaço de entrada ou posições selecionadas no espaço de entrada. Os pesos entre a camada oculta e a camada de saída controlam as contribuições de cada nó oculto. Para MLP com uma camada oculta, selecionar o número próprio de nós na camada oculta é o único mecanismo para regular a complexidade do modelo. Com o acréscimo do número de nós ocultos a complexidade aumenta. Em redes com RBF, o fator de alisamento (também chamada a largura dos campos) provê um mecanismo alternativo para regular a complexidade do modelo. Otimizar o fator de alisamento é usualmente mais fácil e mais rápido que o número de nós da rede. Sabe-se que redes RBF podem ser treinadas muito mais rapidamente que redes com MLP, e o mínimo global é garantido na superfície de erro. O projeto e treinamento de redes com RBF consiste em:

1. determinar quantas funções *kernel* utilizar,
2. encontrar seus centros e largura, e
3. encontrar os pesos que os conecte aos nós de saída.

6.2.4 RBF E A TEORIA DA APROXIMAÇÃO

A propriedade de interpolação da rede com RBF, assegurada pelo teorema de Michelli, MICHELLI(1986), não é suficiente para garantir bons resultados. Uma questão muito importante que surge a partir da teoria da aproximação é se a solução pode ser

[36] Veja tópico de comparação de desempenho dos dois métodos neste capítulo.

impedida de ser alcançada devido às oscilações entre os pontos de interpolação. Isto, assegura POGGIO e GIROSI(1989), não acontece quando se usa funções *spline*, devido às restrições de alisamento que são impostas a este tipo de funções, contudo, pode acontecer quando se usa outras funções radiais.

JACKSON(1988) obteve um importante resultado a respeito desta questão. Segundo sua questão geral: dada uma função multivariada **h(x)** e um conjunto de **N** padrões $\{\mathbf{x}_i | \mathbf{i} = \mathbf{1},...,\mathbf{N}\}$, existe uma seqüência de funções $\mathbf{H}_N(\mathbf{x})$ com

$$\mathbf{H}_N(\mathbf{x}) = \sum_{i=1}^{N} \mathbf{w}_i^N \Phi(\|\mathbf{x} - \mathbf{c}_i\|) + \sum_{j=1}^{k} \alpha_j^N \mathbf{p}_j(\mathbf{x})$$

em algum domínio limitado e aberto no qual

$$|\mathbf{H}_N(\mathbf{x}) - \mathbf{h}(\mathbf{x})| \to \mathbf{0} \text{ quando } \mathbf{N} \to \infty \ ?$$

Segundo a proposta de JACKSON, esta pergunta pode ser respondida afirmativamente por causa das considerações feitas. Em particular, ele considerou que a função $\Phi(\mathbf{r}) = \mathbf{r}$ e mostrou que estas condições são satisfeitas em $\Re^{2n+1}$ mas não em $\Re^{2n}$.

Ainda com respeito à teoria da aproximação, existe o problema com os ruídos nos dados. Em casos como este, fica sem sentido se referir a interpolação pois o erro devido aos ruídos pode ser maior que erro devido à interpolação propriamente dita, e uma modificação conforme a sugerida por TIKHONOV e ARSENIN(1977) pode ser introduzida para se chegar a uma solução para a equação (6.18). A solução proposta consiste em trocar a matriz ϕ por $\phi + \lambda \mathbf{I}$ onde **I** é a matriz identidade, e λ é um parâmetro 'pequeno', cuja magnitude é proporcional à quantidade de ruído nos padrões apresentados. Deste modo, os coeficientes das funções radiais expandidas ficam

$$\mathbf{y}(\mathbf{x}) = \mathbf{W}[\phi + \lambda \mathbf{I}]. \qquad (6.18a)$$

Esta equação dá uma aproximação para a RBF expandida, e a forma original pode ser recuperada se se fixar λ igual a zero.

6.2.5 TEORIA DE REGULARIZAÇÃO

Uma motivação alternativa para expansões RBF vem da teoria da regularização (POGGIO e GIROSSI(1990a e 1990b). A técnica de regularização envolve adicionar à função erro um termo extra o qual é projetado para penalizar mapeamentos os quais não são bem comportados, podendo assim controlar a complexidade do modelo. Isto ocorre porque quando se deseja fazer a aproximação de uma função, algumas considerações a respeito da função são necessárias devido ao fato de que num conjunto de dados $\mathbf{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_1^N$, com ruídos, existirem 'infinitas' respostas possíveis, o que torna o sistema mal colocado. Uma maneira de consideração é fazer a função ser aproximada por alguma forma $\mathbf{y}(\mathbf{w},\mathbf{x})$ as quais dependem somente dos parâmetros $\mathbf{w}$ desconhecidos. A solução do problema colocado desta maneira depende muito fortemente de uma consideração (forma da função) correta. Se com a forma funcional assumida é possível apenas uma representação não muito fiel da verdadeira função buscada, nenhuma técnica de regressão possibilitará um ajuste melhor, por melhor que seja tal técnica. Por outro lado, ao invés de se escolher a forma funcional, a forma mais comum é assumir que a função de aproximação é uma função bem comportada no sentido de que duas entradas similares correspondem a duas saídas similares.

Para simplicidade de notação sejam consideradas redes que tenham apenas uma saída simples $\mathbf{y}$, tal que com a soma dos quadrados dos erros, a função erro total a ser minimizada torna-se

$$\mathbf{E} = \frac{1}{2}\sum_{\mathbf{n}} \left\{\mathbf{y}\left(\mathbf{x}^{\mathbf{n}}\right) - \mathbf{t}^{\mathbf{n}}\right\}^2 + \frac{\upsilon}{2}\int \left|\mathbf{P}\mathbf{y}\right|^2 \mathbf{dx} \qquad (6.22)$$

onde **P** é algum operador diferencial, como por exemplo o regularizador de Tikhonov, e υ é chamado o fator de regularização, e pode ser visto como um fator de suficiência do conjunto de dados para especificar a função. Em particular, quando o valor do fator de limitação é zero, implica que o problema é não restringido e então a solução da função pode ser encontrada através dos exemplos. No outro extremo, quando o fator tende ao infinito, implica que uma restrição de bom comportamento é *per se* suficiente para especificar a solução. O primeiro termo da equação se refere à proximidade da função y em relação aos dados e o segundo termo se refere ao comportamento da função, presença ou não de oscilações. Funções de mapeamento de rede **y(x)** as quais têm uma grande curvatura darão surgimento a grandes valores de $|\mathbf{Py}|^2$ e conseqüentemente a grandes penalidades na função erro total. O valor de υ controla a importância relativa do termo de regularização, e conseqüentemente o grau de 'alisamento' da função **y(x)**.

Pode-se resolver o problema de mínimos quadrados de (6.22) usando cálculo das variações, BISHOP(1995), como segue. Ajustando-se a derivada funcional de (6.22) com respeito a **y(x)** para zero obtém-se

$$\sum_n \left\{\mathbf{y}\left(\mathbf{x}^n\right)-\mathbf{t}^n\right\}\delta\left(\mathbf{x}-\mathbf{x}^n\right)+\upsilon\hat{\mathbf{P}}\mathbf{P}\mathbf{y}(\mathbf{x})=\mathbf{0} \tag{6.23}$$

onde $\hat{\mathbf{P}}$ é o operador diferencial adjunto para **P** e $\delta(\mathbf{x})$ é a função Delta de Dirac. As equações (6.23) são as equações Euler-Lagrange correspondentes a (6.22). Uma solução formal para estas equações pode ser escrita em termos do teorema de Green do operador $\hat{\mathbf{P}}\mathbf{P}$, as quais são funções **G(x,x')** que satisfazem

$$\hat{\mathbf{P}}\mathbf{P}\mathbf{G}(\mathbf{x},\mathbf{x}')=\delta(\mathbf{x}-\mathbf{x}'). \tag{6.24}$$

Se o operador **P** é translacionalmente e rotacionalmente invariável, as funções de Green dependem somente da distância $\|\mathbf{x}-\mathbf{x}'\|$, e conseqüentemente são funções radiais. A solução formal para (6.23) pode ser escrita como

$$\mathbf{y}(\mathbf{x}) = \sum_{n} \mathbf{w}_n \mathbf{G}\left(\left\|\mathbf{x} - \mathbf{x}^n\right\|\right) \tag{6.25}$$

a qual tem a forma de uma expansão linear em funções base radial. Substituindo (6.25) em (6.23) e usando (6.24) obtém-se

$$\sum_{n} \left\{\mathbf{y}(\mathbf{x}^n) - \mathbf{t}^n\right\}\delta(\mathbf{x} - \mathbf{x}') + \upsilon \sum_{n} \mathbf{w}_n \delta(\mathbf{x} - \mathbf{x}') = \mathbf{0}. \tag{6.26}$$

Integrando-se sobre toda a região em volta de $\mathbf{x}^n$ mostra que os coeficientes de $\mathbf{w}_n$ satisfaz

$$\mathbf{y}(\mathbf{x}^n) - \mathbf{t}^n + \upsilon \mathbf{w}_n = \mathbf{0}. \tag{6.27}$$

Valores para os coeficientes $\mathbf{w}_n$ podem ser encontrados resolvendo-se (6.25) nos valores dos dados de treinamento $\mathbf{x}^n$ e substituindo-se em (6.27). Isto dá os valores de $\mathbf{w}_n$ como as soluções da equação linear

$$(\mathbf{G} + \upsilon \mathbf{I})\mathbf{w} = \mathbf{t} \tag{6.28}$$

onde $(\mathbf{G})_{nn'} = \mathbf{G}\left(\left\|\mathbf{x}^{n'} - \mathbf{x}^n\right\|\right)$, $(\mathbf{w})_n = \mathbf{w}_n$, $(\mathbf{t})_n = \mathbf{t}^n$ e $\mathbf{I}$ denota a matriz unitária.

Se o operador $\mathbf{P}$ é escolhido como tendo a forma particular

$$\int |\mathbf{P}\mathbf{y}|^2 \, d\mathbf{x} = \sum_{\lambda=0}^{\infty} \frac{\sigma^{2\lambda}}{\lambda! 2^{\lambda}} \int \left|\mathbf{D}^{\lambda}\mathbf{y}(\mathbf{x})\right|^2 d\mathbf{x} \tag{6.29}$$

onde $\mathbf{D}^{2\lambda} = \left(\nabla^2\right)^{\lambda}$ e $\mathbf{D}^{2\lambda+1} = \nabla\left(\nabla^2\right)^{\lambda}$, com ∇ e ∇^2 denotando os operadores gradiente e Laplaciano respectivamente, então as funções de Green são Gaussianas com largura dos parâmetros igual a σ.

Existe muita similaridade entre esta forma de função base e aquela discutida no contexto da interpolação exata. Aqui, as funções de Green $\mathbf{G}\left(\left\|\mathbf{x}-\mathbf{x}^n\right\|\right)$ correspondem às funções base $\phi\left(\left\|\mathbf{x}-\mathbf{x}^n\right\|\right)$, e há uma função deste tipo centrada em cada ponto dado no conjunto de treinamento. Também vê-se que (6.28) reduz o resultado da interpolação exata (6.3) quando o parâmetro de regularização σ é zero. Quando o parâmetro de regularização é maior que zero, entretanto, não se tem uma interpolação exata. O efeito do termo de regularização é forçar um mapeamento mais 'alisado' da função da rede.

Na prática, a regularização também pode ser aplicada a redes com RBF nas quais as funções não sejam restringidas a ser centradas nos pontos de dados, nos padrões, e nos quais o número de funções base não necessitam igualar o número de pontos dados. Também, termos de regularização podem ser considerados para funções cujas bases não sejam necessariamente funções de Green.

Levando-se em conta que o termo de regularização é uma função quadrática do mapeamento da rede, os pesos da segunda camada podem novamente ser encontrados pela solução de um conjunto de equações lineares as quais minimizam a soma dos erros quadrados. Por exemplo, o regularizador

$$\frac{\upsilon}{2}\sum_n\sum_k\sum_i\left(\frac{\partial^2 \mathbf{y}_k^n}{\partial \mathbf{x}_i^2}\right) \tag{6.30}$$

penaliza os mapeamentos os quais tenham grande curvatura, BISHOP(1991). Este regularizador leva aos pesos da segunda camada os quais são encontrados pela solução de

$$\mathbf{MW} = \Phi^{\mathrm{T}}\mathbf{T} \tag{6.31}$$

onde

$$(\mathbf{M})_{jj'} = \sum_{n} \left\{ \phi_j^n \phi_{j'}^n + \upsilon \sum_i \left(\frac{\partial^2 \phi_j^n}{\partial \mathbf{x}_i^2} \frac{\partial^2 \phi_{j'}^n}{\partial^2 \mathbf{x}_i^2} \right) \right\} \tag{6.32}$$

e $\Phi = \left(\phi_n^j\right)$, como antes. Quando $\upsilon = \mathbf{0}$ (6.31) reduz-se aos resultados prévios (6.20). A inclusão do termo de regularização adiciona pequenas porções ao custo computacional, desde que a maior parte do tempo é gasta resolvendo-se as equações lineares acopladas (6.31).

Mais motivação para o uso de RBF para aproximação de funções vem da teoria de regressão kernel. A meta do aprendizado é encontrar um mapeamento de **x** para **y** o qual capture os aspectos sistemáticos dos dados, sem ajustar os ruídos dos dados. As bases para o entendimento desta técnica estão descritas no trabalho de GHAHRAMANI e JORDAN(1994). Para a utilização de redes com RBF para classificação, pode-se consultar o trabalho de DUDA e HART(1973) ou o de BISHOP(1995), onde existe um completo desenvolvimento da teoria.

6.2.6 SUBCONJUNTOS DE DADOS

Um procedimento simples para a seleção de centro de base de funções μ_j é ajustá-los iguais ao subconjunto randômico dos vetores de entrada a partir do conjunto de treinamento. É claro que isto não é um procedimento ótimo e está longe de estar próximo à estimação da densidade, e isto pode levar ao uso de um número desnecessário de funções base a fim de conseguir desempenho adequado no treinamento de dados. Este método é freqüentemente usado, entretanto, para proporcionar um conjunto de valores iniciais para muitos dos procedimentos adaptativos.

Outro caminho é começar com todos os dados como centros de funções base e então seletivamente remover centros nos quais se tenha perturbações mínimas no desempenho do sistema.

Estas técnicas apenas ajustam os centros das funções base, e as larguras dos parâmetros σ_j devem ser escolhidos usando algum outro procedimento. Uma heurística apropriada é escolher todos os σ_j como sendo iguais e dados por algum múltiplo da distância média entre os centros das funções base. Isto assegura que as funções base sobreponham em algum grau o que, conseqüentemente, dá uma representação relativamente boa da distribuição dos dados de treinamento. Deve-se também reconhecer que a largura ótima pode ser diferente para as funções base em diferentes regiões do espaço de entrada. Por exemplo, as larguras podem ser determinadas a partir da distância média de cada função base aos seus L mais próximos vizinhos de cada função base, onde L é tipicamente 'pequeno'. Tais procedimentos de escolher os parâmetros da função base são muito rápidos, e permite que a rede com RBF seja ajustada rapidamente, contudo, são procedimentos que não podem ser considerados como ótimos.

6.2.7 MÍNIMO QUADRADO ORTOGONAL

O principal caminho para selecionar um subconjunto de dados como centros de função base é inspirado na técnica dos mínimos quadrados ortogonais. O procedimento para seleção de funções base inicia-se considerando-se uma rede com apenas uma função base. Para cada dado ajusta-se o centro da função base para o vetor de entrada para aquele dado, e então ajusta-se os pesos da segunda camada pela técnica da pseudo-inversa usando o conjunto de N dados. Os centros das funções base os quais dão os menores erros residuais são conservados. Nos passos subseqüentes do algoritmo, o número de funções base é então aumentado incrementalmente. Se a este ponto no algoritmo, λdos pontos dados tiver sido selecionado como centro das funções base, então N-λredes são treinadas e cada um dos restantes N-λdados, por sua vez, é selecionado como centro das funções base adicionais. As funções base extra, as quais dão o menor valor para a soma dos quadrados dos erros residuais, são então conservadas, e o algoritmo prossegue para o próximo estágio.

Tal caminho é computacionalmente intensivo desde que a cada passo é necessário obter uma solução completa da pseudo-inversa para cada possível escolha das funções base.

Um procedimento muito mais eficiente para conseguir o mesmo resultado é o do mínimo quadrado ortogonal, CHEN et al.(1989,1991). Em linhas gerais, o algoritmo envolve uma adição seqüencial das novas funções base, cada uma centrada nos dados, como descrito acima. Isto é feito construindo-se um conjunto de vetores no espaço S ocupado pelos vetores das unidades de ativação ocultas para cada padrão no conjunto de treinamento. É então possível calcular diretamente quais os dados deveriam ser escolhidos como os próximos centros das funções base a fim de produzir a maior redução na soma dos quadrados do erros residuais. Valores para os pesos da segunda camada são também determinados ao mesmo tempo. Se o algoritmo é aplicado continuamente, então todos os pontos serão selecionados, e o erro residual será zero. A fim de conseguir boa generalização, o algoritmo deve ser parado antes que isto ocorra. Este é o problema da seleção da ordem do modelo.

6.2.8 APRENDIZAGEM EM REDES COM RBF

Quando se usa grandes conjuntos de dados para o treinamento de redes com RBF o número de nós na camada oculta e a localização dos centros terá profundos efeitos no desempenho da solução da rede. A escolha mais natural, como dito antes, é fazer o número de nós da camada oculta como sendo igual ao número de padrões disponíveis para o treinamento da rede. Este procedimento leva à mais básica forma de rede com RBF. As principais vantagens desta escolha são: 1) é simples; 2) a escolha dos centros das funções é unicamente determinado; 3) utiliza totalmente o conjunto de dados apresentados; e 4) é altamente desejável no caso de entradas de alta dimensão e de dados de treinamento altamente esparsos. Entretanto, existem muitos problemas associados à essa solução simples. Por exemplo, custo de computação, problemas com o mau-condicionamento, problemas com sobreajustamento, ou seja, mais dados que o necessário. Uma análise de cada um dos problemas pode ser feita como a seguir.

Custo de computação: quando se tem um número muito grande de dados, o processamento destes dados pode ser problemático, tomando um alto tempo, pois a complexidade cresce

polinomialmente com ordem N, uma vez que uma matriz desta ordem precisará ser invertida.

Mau-condicionamento: na operação de inversão da matriz, o número da condição é uma medida do efeito sobre uma matriz inversa aproximada. É um número definido como a razão do maior autovalor pelo menor autovalor. Se o número da condição é 'próximo' a N, a matriz é dita ser bem-condicionada. Se é muito maior que N, é dita ser mal-condicionada, e a solução aproximada obtida para a equação linear $\mathbf{Ax} = \mathbf{b}$ será muito sensível ao valor dos coeficientes.

Sobreajustamento: escolhendo-se uma função base para cada padrão, a rede tem no mínimo N+n+1 parâmetros livres (N pesos, n *bias* para os nós de saída, 1 largura da função base). O número de graus de liberdade na rede é maior que o número de dados. A rede pode passar exatamente através de cada dado quando a largura da função base é escolhida suficientemente pequena. Se os dados têm um comportamento regular mas estão contaminados por ruídos, a rede reterá todas as características dos pontos individuais. Este fenômeno é então chamado de sobreajustamento. O modelo ajusta muito bem o conjunto de dados apresentados, mas normalmente é péssimo para generalização do treinamento. As oscilações do ajuste podem ser muito intensas, e isto caracteriza o fenômeno.

A fim de se contornar estes problemas acima apresentados, associados às redes com RBF, inúmeros métodos têm sido propostos para reduzir o número de centros das funções e para selecionar os centros das mesmas.

6.2.9 ALGORITMOS DE CLUSTERING (OU ALGORITMOS DE AGRUPAMENTO)

Por causa do curso da dimensionalidade que atormenta muitos procedimentos de reconhecimento de padrões, uma grande variedade de redutores de dimensão têm sido

propostos. Os procedimentos estatísticos clássicos são 'análise do componente principal' e 'análise de fator', sendo que ambos reduzem a dimensionalidade através da combinação de características.

Em sentido amplo, um esquema de classificação pode simplesmente representar um método conveniente para organizar um grande conjunto de dados tal que a recuperação de informações seja feita de forma mais eficiente. Descrever padrões de similaridade e diferenças entre os objetos sob investigação por meio dos rótulos de suas classes pode prover um resumo apropriado de dados, EVERITT(1993). A solução geralmente buscada é a partição dos **n** objetos dados em um conjunto de *clusters* onde um objeto pertence a somente um núcleo, e o conjunto completo de tais núcleos contém todos os objetos.

Os dados básicos para análise de *clusters* é, e geral, uma matriz **X**, a qual dá os valores das variáveis para cada um dos objetos ou indivíduos sob investigação. Desta forma

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_{11} & \mathbf{x}_{12} & \Lambda & \mathbf{x}_{1p} \\ \mathbf{x}_{21} & \mathbf{x}_{22} & \Lambda & \mathbf{x}_{2p} \\ \mathrm{M} & \mathrm{M} & \mathrm{O} & \mathrm{M} \\ \mathbf{x}_{n1} & \mathbf{x}_{n2} & \Lambda & \mathbf{x}_{np} \end{pmatrix}$$

onde as variáveis podem ser de diferentes tipos.

Como um simples melhoramento da escolha do subconjunto de dados como os centros das funções base, pode-se usar as técnicas de *clustering* para encontrar um conjunto de centros os quais mais precisamente reflete a distribuição dos dados. MOODY e DARKEN(1989) usam o algoritmo *K-means clustering*, no qual o número **K** de centros deve ser decido de antemão. O algoritmo envolve um procedimento de re-estimação.

6.2.10 SELEÇÃO UNIFORME OU RANDÔMICA DOS CENTROS

Seleção randômica dos centros: a fim de se reduzir o número de centros das funções (i.e, o número de nós ocultos), o método mais fácil e o menos intensivo computacionalmente é o método de seleção randômica: selecionando-se **m** padrões randomicamente dentro do universo N de padrões (**m<N**). Este método se baseia no fato de que se o conjunto de N padrões do conjunto de treinamento forma um conjunto representativo, um subconjunto retirado randomicamente deste conjunto inicial, dará um bom subconjunto representativo de dados. Esta técnica dá resultados de pouca confiabilidade para conjuntos de dados pouco representativos.

Seleção uniforme dos centros: colocando-se os centros em uma grade uniforme assegura-se que os centros estejam uniformemente distribuídos sobre o espaço de entrada. Entretanto, esta técnica produz bons resultados somente se os próprios padrões também estiverem também uniformemente distribuídos pelo espaço de entrada.

6.2.11 ALGORITMO K-MEANS CLUSTERING

Ao invés de colocar os centros das funções em cada um dos padrões dados ou em um subconjunto de dados randomicamente selecionados, um método mais comumente usado é o que emprega o algoritmo *k-means clustering* para encontrar os centros das funções, DUDA e HART(1973).

Em geral, tais algoritmos são usados para agrupar objetos definidos por um conjunto de propriedades numéricas de tal forma que os objetos dentro do grupo, são mais similares que os objetos em grupos diferentes. Portanto, a um algoritmo *clustering* é preciso que se dê critérios para se medir a similaridade dos objetos, e de como se agrupar dentro do conjunto. O algoritmo *k-means* usa a distância Euclidiana para medir as similaridades

entre os objetos. Existem dois tipos de procedimentos para o algoritmo *k-means*, o adaptativo e o padrão. O algoritmo *K-means clustering* precisa assumir que o número de grupos (*clusters*) é conhecido previamente.

O algoritmo padrão foi desenvolvido em 1957, e está descrito em LLOYD(1982), e em MaCQUEEN(1967) e é um algoritmo de agrupamento (*clustering*) geral para agrupar N objetos em **m** grupos de um dado número **m**. O método minimiza a distância Euclidiana quadrada **E** da forma:

$$\mathbf{E} = \sum_{i=1}^{N}\sum_{j=1}^{m}\mathbf{M}_{ij}\left\|\mathbf{x}_i - \mathbf{c}_j\right\|$$

onde c_j; j=1,2,...,m são os centros dos *clusters*, $\mathbf{x}_i$; i=1,2,...,N são os N objetos. $\mathbf{M}_{ij}$ é a função membro do *cluster* a qual é uma matriz **M** x **m** de 0's e 1's com exatamente um **1** por linha o qual identifica os grupos aos quais os objetos pertencem. Neste algoritmo, a similaridade entre os objetos é definida pela distância Euclidiana: a menor distância entre dois objetos representa que os dois objetos são mais similares.

O algoritmo *k-means*

1. Escolher inicialmente um número **m** de *clusters*. Randomicamente, escolher os **m** padrões de treinamento e atribuí-los aos centros iniciais dos *clusters*.
2. Para cada padrão $\mathbf{x}_i$, atribua $\mathbf{x}_i$ ao grupo j*, onde $\left\|x_i - c_{j^*}\right\| = \min_j \left\|x_i - c_j\right\|$.
3. Para cada grupo c_j, $\mathbf{c}_j = \frac{1}{\mathbf{m}_j}\sum_{x_i \in \text{grupo } j}\mathbf{x}_i$, onde $\mathbf{m}_j$ é o número de padrões no grupo j.
4. Repita o passo 2, até que não ocorra mudanças no número de *clustering* ou tenha atingido algum critério de distância pré-estabelecido.

6.2.12 MAPAS DE CARACTERÍSTICAS AUTO-ORGANIZÁVEIS DE KOHONEN

O algoritmo de Kohonen pode ser usado para determinar os centros do agrupamentos devido à sua similaridade com o algoritmo *k-means*.

O algoritmo de Kohonen é uma rede de duas camadas, entrada e saída. Os nós da camada de saída são, usualmente, organizados em uma matriz de duas dimensões. A conectividade dos nós na camada de saída define a forma dos mapas de características. Cada nó de entrada é totalmente conectado a cada nó de saída no mapa. Os pesos entre a camada de entrada e a camada de saída são modificados durante o treinamento.

Este algoritmo é viável quando o número de centros de *clusters* pode ser especificado antes de se começar o procedimento de treinamento e a quantidade de centros desejado é alto. É deste ponto de vista, similar ao algoritmo *k-means*.

6.2.13 ALGORITMO DE DISTÂNCIA MaxMin

O algoritmo *k-means* somente pode ser usado quando se conhece, antes de começar o processamento dos dados, o número de centros ou *clusters*. Caso este valor não seja conhecido nestas condições, um outro algoritmo deve ser empregado. É neste contexto que aparece o algoritmo de distância mínima e máxima. É um algoritmo simples devido a BATCHELOR e WILKINS(1969) os quais desenvolveram uma forma de agrupar os dados em um número desconhecido de centros ou *clusters*.

O algoritmo tenta alocar os centros de forma tão uniforme quanto possível no espaço dos padrões apresentados. O número de centros de funções encontrados pelo algoritmo depende da distância mínima especificada (ou pela distância absoluta).

Algoritmo de MaxMin distância

1. Iniciar o primeiro centro das funções. Ajuste o centro inicial $\mathbf{c}_1$ como sendo o primeiro padrão de treinamento ou qualquer outro padrão randômico escolhido.

2. Calcular a distância mínima de cada padrão. Para cada padrão $\mathbf{x}_i$, $\mathbf{d}_i \min_j \left\| \mathbf{x}_i - \mathbf{c}_j \right\|$.

3. Selecionar o padrão com a máxima distância mínima. $\mathbf{D} = \mathbf{d}_{j^*} = \max_i \{\mathbf{d}_i\}$.

Se $\mathbf{D} > \frac{1}{2}$ do valor médio de $\{\mathbf{D}_{\text{prévio}}$ ou $\mathbf{D} > \mathbf{D}_{\min}$ (onde $\mathbf{D}_{\min}$ é uma distância mínima especificada) então adicionar o padrão $\mathbf{x}_j$ ao conjunto de centros, $\mathbf{c}_j = \mathbf{x}_j$.

4. Repita o passo 2 enquanto um novo centro for gerado.

Este foi o algoritmo de *clustering* implementado no programa apresentado neste trabalho.

6.2.14 REDES COM RBF COMO APROXIMADORES UNIVERSAIS

Redes com RBF de λ entradas, **m** nós ocultos e **n** saídas processa um mapeamento $\mathbf{y} : \Re^{\lambda} \rightarrow \Re^{n}$. Antes de considerar quaisquer problemas práticos (por exemplo como escolher o tamanho da rede, ou como os pesos serão determinados durante o treinamento) de obtenção da solução da rede com RBF para o problema dado, é preciso responder a uma questão fundamental a respeito da habilidade de representação das redes com RBF. A questão é: qual o tipo de classes de funções podem ser bem aproximadas pela rede com RBF, $\mathbf{y}(\mathbf{x}, \mathbf{W}_{\lambda-m-n})$? Aqui, $\mathbf{W}_{\lambda-m-n}$ representa os parâmetros livres da rede.

A propriedade de aproximador universal: redes com RBF podem ser vistas como esquemas de aproximação. Um esquema de aproximação é um instrumento de um conjunto $\Im$ de funções 'baseadas' no espaço $\Re^{\lambda}$. Usualmente, este instrumento consiste de um número de componentes (por exemplo o número de nós ocultos **m** na rede), tal que o conjunto $\Im$ pode ser caracterizado por este número (**m**); i.e., pode-se denotá-lo por $\Im_m$. Deste modo pode-se escrever $\Im_U = Y_{m=1}^{\infty} \Im_m$. Então o modelo da função é dito ter a propriedade de aproximador universal, HORNIK et al.(1989), se $\Im_U$ é densa no espaço das funções contínuas **C[U]** definidas em algum domínio **U** de $\Re^{\lambda}$; ou em outras palavras, para qualquer função contínua **y(x)** baseada em **U** existe uma função específica $\hat{\mathbf{y}}_m \in \Im_m$ tal que $\hat{\mathbf{y}}_m(\mathbf{x})$ pode aproximar **y(x)** arbitrariamente bem.

HARTMAN e KELLER(1990), mostraram que a rede com RBF com nós ocultos com funções Gaussianas é um aproximador universal para mapeamentos definidos em espaço convexo, $\Re^{\lambda}$, isto está provado no teorema de Stone-Weierstrass, no mesmo artigo.

GIROSI e POGGIO(1990) provaram que redes com RBF definidas pela equação

$$\mathbf{y}(\mathbf{x}) = \sum_{j=1}^{m} \mathbf{w}_j \phi_j(\mathbf{x})$$

onde os $\mathbf{w}_j$ são desconhecidos, **m** é o número de nós da camada oculta, e $\phi_j = \phi\left(\left\|\mathbf{x} - \mathbf{x}^j\right\|\right)$ é a função base radial centrada no ponto $\mathbf{x}^j$, tem a propriedade da melhor aproximação.

Um esquema de aproximação tem a propriedade de melhor aproximação se no conjunto $\Im$ das funções de aproximação, há uma função a qual tem erro de aproximação mínimo para qualquer função a ser aproximada a partir de um dados conjunto de funções.

6.3 COMPARAÇÃO DAS REDES RBF COM PERCEPTRON DE MÚLTI-CAMADA

Resultados teóricos têm demonstrado que diversos tipos de redes neurais tem a propriedade universal da aproximação, isto é, têm a habilidade para representar uma função arbitrária com um grau de precisão que é função do número de unidades intermediárias. Entretanto, considerações práticas tais como as relativas às vantagens de diferentes redes neurais para a aproximação de funções usando uma quantidade moderada de unidades ocultas, não são completamente entendidas, RUSSEL e FAUSETT(1996).

Redes neurais com RBF e *perceptrons* de múltiplas camadas têm papéis semelhantes uma vez que ambos provêem técnicas para aproximação de funções não lineares, mapeando-as em espaços múlti-dimensionais. Em ambos os casos, o mapeamento é expresso em termos de composições parametrizadas de funções de variáveis simples. Entretanto, as estruturas particulares das duas redes são muito diferentes e isso é interessante de ser comparado em mais detalhes. Algumas das importantes diferenças entre *perceptrons* de múltiplas camadas e redes com RBF são:

1. As representações das unidades ocultas do *perceptron* de múltiplas camadas depende dos somatórios ponderados (pelos pesos) dos valores de entrada, transformados por funções de ativação monotônicas. Assim, a ativação da unidade oculta em *perceptrons* são superfícies constantes. Em contraste, as unidades ocultas em redes com RBF usam a distância para um vetor protótipo seguido pela transformação com uma (usualmente) função localizada. A ativação de uma função base é desta forma constante em hiperesferas concêntricas de dimensão **d-1** (ou mais geralmente, hiperelipsóides de dimensão **d-1**).

2. Um *perceptron* de múltiplas camadas pode ser considerado como formador de representação distribuída no espaço dos valores de ativação para as unidades desde que, para um dado vetor de entrada, muitas unidades ocultas contribuirão para a determinação do valor de

saída. Durante o treinamento, as funções representadas pelas unidades ocultas devem ser tais que, quando linearmente combinadas pelos pesos da camada final, eles geram saídas corretas para uma faixa de valores de entrada possível. A interferência e cruzamento de pares entre as unidades ocultas, os quais isto requer, faz com que o processo de treinamento seja altamente não linear, com problemas de mínimo local, ou próximo a regiões monótonas na função erro resultando em quase cancelamento nos efeitos dos diferentes pesos. Isto pode levar a convergência muito lenta do procedimento de treinamento mesmo com avançadas estratégias de otimização. Em contraste, a rede com RBF, com funções base localizadas, forma uma representação no espaço de unidades ocultas a qual é local com respeito ao espaço de entrada porque, para um dado vetor de entrada, somente poucas unidades terão ativação significativa, e devido a tal, tem um procedimento de aprendizagem garantido, BROOMHEAD e LOWE(1988).

3. Em contraste às MLP's, as redes com RBF são capazes de representar transformações não lineares arbitrárias representadas por um conjunto de padrões entrada-saída, desde que conforme mostrado por MICHELLI(1986), o aprendizado com RBF pode produzir interpolações que passam exatamente pelos padrões dados. É bem verdade, no entanto, que esta interpolação exata pode produzir resultados tão ruins de generalização quanto aqueles obtidos pela interpolação com MLP.

4. Um *perceptron* de múlticamada freqüentemente tem muitos pesos, e um padrão de conectividade complexo, tal que nem todos os pesos possíveis em uma dada camada estão presentes. Uma variedade de diferentes funções de ativação pode ser usada dentro da mesma rede. Uma rede com RBF, entretanto, geralmente tem uma arquitetura simples consistindo de duas camadas de pesos, na qual a primeira camada contém os parâmetros das funções base, e a segunda camada forma as combinações das ativações das funções base para gerar as saídas.

5. Todos os parâmetros no *perceptron* são normalmente determinados ao mesmo tempo como parte da estratégia de treinamento global envolvendo treinamento supervisionado.

Uma rede com RBF, entretanto, é tipicamente treinada em dois estágios, com as funções base sendo determinadas primeiro por técnicas não supervisionadas usando somente os dados de entrada, e os pesos da segunda camada subseqüentemente sendo encontrados por rápidos métodos lineares supervisionados.

CAPÍTULO 7

7 RESULTADOS: DESENVOLVIMENTO DA METODOLOGIA DE ANÁLISE DE FALHAS E DESCRIÇÃO DE CONSTRUÇÃO E DE USO DO SOFTWARE PARA TREINAMENTO DAS REDES NEURAIS

7.1 DESENVOLVIMENTO DA METODOLOGIA DE ANÁLISE DE FALHAS

O desenvolvimento da metodologia para o detecção/diagnóstico utilizando as redes neurais através do programa desenvolvido neste trabalho encontra-se descrito no corpo deste capítulo, a partir do item 7.4, de forma não agrupada. Optou-se por tal procedimento a fim de colocar de forma mais próxima, a metodologia e o exemplo correspondente de tal forma que facilitasse a compreensão do leitor. A cada uma das proposições feitas, existe um esquema em forma de figura que descreve o mecanismo proposto bem como o seu funcionamento.

7.2 O SOFTWARE -descrição

Conforme as propostas iniciais descritas no capítulo de objetivos deste trabalho, implementou-se um programa computacional em linguagem C (compatível com Visual C++ 5.0, anexo no Apêndice C), com várias possibilidades de escolha de algoritmos de

atualização de pesos, seja em *backpropagation*, seja com as atualizações pelos algoritmos de Marquardt-Levenberg, por um algoritmo quase-Newton ou usando as funções base radiais, o qual possibilita determinar os conjuntos de pesos para redes com uma camada oculta e com possibilidade de utilização de diferentes números de neurônios na camada intermediária.

Na camada inicial, o número de neurônios é, de certa forma, fixo pois está diretamente relacionado à quantidade de variáveis do processo que serão utilizadas para monitoramento da planta e que servirão como parâmetros independentes no treinamento das redes neurais. O fato de se dizer que seja fixo o número de neurônios não implica dizer que o programa seja particular, pois pode-se usar a quantidade de neurônios que for necessária, bastando para tanto mudar a quantidade de neurônios na camada inicial para o treinamento da rede, no arquivo de entrada de dados, descrito mais à frente.

Na camada final, o número de neurônios é mais flexível e dependerá do esquema de determinação das falhas. Poderá, desta forma, ter no mínimo uma saída para indicar que alguma condição de falha ocorreu; ou uma quantidade que seja o número de falhas mais uma condição sem falhas. No caso de utilizar-se apenas uma saída, atribui-se um valor de saída para a condição que se deseja associar (falha ou não-falha), e considera-se como certas as respostas dentro de um certo intervalo ao redor do valor considerado. Dentro deste último tipo de determinação, o melhor é utilizar dois valores extremos para indicar situações também extremas, podendo-se assim, para uma saída binária associar, por exemplo, **1** à condição de falha e **0** caso contrário.

O programa em *backpropagation* 'puro' ou mesmo o que utiliza dois possíveis tipos de atualizadores de pesos, não apresentou resultados muito animadores, quando se tinha em mente a utilização do procedimento para problemas em escala bem maior que a apresentada na literatura pesquisada. No programa aqui desenvolvido, a forma de utilização é indicada por uma variável (**met**) que pode ser modificada no começo do programa e que identifica o tipo de algoritmo a ser utilizado. O número **0** indica uma rede com algoritmo modificado segundo o esquema de Marquardt-Levenberg; o número **2** está associado à modificação quase-Newton e o número **1,** o qual inicialmente indicava uma rede com *backpropagation* 'pura' passou a designar a rede com as funções base radiais, uma vez que é impossível treinar redes em tempo razoável e com grandes conjuntos de dados ao se utilizar o *backpropagation.*

Os resultados, de acordo com o que pode ser conferido de modo comparativo nas tabelas de resultados ao longo deste capítulo, mostra que o programa com RBF apresenta resultados muito melhores, e possibilita o treinamento de redes de maior porte, permitindo conseguir atingir os propósitos deste trabalho, qual seja, detectar e diagnosticar falhas utilizando dados dinâmicos do processo.

No primeiro caso do primeiro algoritmo, foi utilizado o *backpropagation* tendo os pesos atualizados pelo gradiente descendente (GDR) o qual é o mais comum e mais utilizado pela grande maioria dos autores consultados. É um método robusto e que consegue ajustar os padrões para grandes conjuntos de dados o que é sem dúvida uma grande vantagem. Por outro lado, a sua lentidão de convergência se faz notar de modo muito acentuado conforme se vai aumentando a dimensão do conjunto de dados a serem treinados. Desta forma, para conjuntos de dados de sistemas complexos, sua utilização prática pode ficar comprometida, sendo deste modo um procedimento de baixa versatilidade.

No segundo caso do primeiro algoritmo, utilizou-se uma variação do modelo descrito anteriormente o qual é também um método em *backpropagation* que utiliza como atualizador dos pesos, uma forma baseada no método de Marquardt-Levenberg, descrito em HAGAN e MENHAJ (1994). Este método, conforme explicitado por vários dos autores citados neste trabalho, possibilita convergências rápidas, contudo, a utilização do método das derivadas para determinação da direção de atualização dos pesos, tem como agravante o grande espaço requerido na memória do computador para se treinar redes com pequena dimensão. No caso de redes de grande dimensão, o método é impraticável devido ao grande esforço computacional necessário para inverter a matriz jacobiana, e além do mais, mesmo que se consiga processar os cálculos, o tempo requerido não é de forma alguma compensador.

O terceiro caso, ainda do primeiro algoritmo, é um método quase-Newton e como no caso anterior, tem como desvantagem o grande esforço computacional requerido para treinar as redes de dimensões média e grande.

Devido às características dos métodos aqui apresentados e às exigências do trabalho, fica evidente que um novo algoritmo deveria ser empregado, visando a contornar os problemas encontrados. Este novo algoritmo, como já citado nos parágrafos anteriores,

se encontra detalhado no capítulo 6 deste trabalho e emprega funções base radiais, as quais se mostraram muito mais eficientes para o treinamento das redes neurais aqui utilizadas.

O algoritmo com RBF, é um procedimento que utiliza duas formas de determinação dos pesos durante o processo de treinamento das redes neurais. A primeira fase do treinamento utiliza um algoritmo baseado na distância dos padrões dados a um certo centro ou *kernel*, e é basicamente um procedimento não-supervisionado e não iterativo, sendo portanto rápido. A segunda fase, mais demorada, se comparada à primeira, se desenvolve de modo muito mais eficiente que o algoritmo tentado anteriormente, devido à nova maneira de se ativar os neurônios da camada intermediária.

No programa aqui implementado, fazendo-se testes empíricos, optou-se por restringir o número de centros em 100, devido ao bom comportamento das curvas que representavam a resposta dinâmica do sistema utilizado para estudo e devido à grande quantidade de dados utilizados para o treinamento das redes, o que aumentaria em demasia o trabalho computacional sem grandes melhorias de aproximação dos valores requeridos. Tal procedimento 'suaviza' o ajuste, uma vez que não se terá, neste caso, a 'função' ajustada passando exatamente por todos os padrões dados, o que diminuindo a quantidade de iterações necessárias, diminui o tempo para o treinamento da rede.

Os valores para os parâmetros dos *bias* são incorporados à soma linear de determinação dos pesos, de forma que esse termo é ajustado como se fosse simplesmente mais um peso da estrutura da rede neural.

O algoritmo utilizado para se fazer a limitação do número de centros é o de distância MinMax que pode ser conferido no item 6.2.13. Optou-se, ao invés de impor uma quantidade de centros, o que possibilitaria usar o algoritmo *k-means*, por delimitar a sua quantidade máxima. O programa tem, desta forma um limitador (**max_centros**) que faz a comparação entre as distâncias entre os centros escolhidos, automaticamente, e os aloca de forma tão uniforme quanto possível no espaço dos padrões apresentados à entrada.

Caso seja o número de padrões menor que 100, o número de centros dependerá da distância mínima especificada. O algoritmo calculará as distâncias euclidianas entre cada centro, começando obviamente com um único centro, e todos os demais padrões e determinará o número de centros necessários. O critério de parada, aqui utilizado, segue exemplos encontrados na literatura os quais dizem que um valor apropriado para cessar a

atribuição dos centros se encontra através da comparação entre a máxima distância (euclidiana) mínima e uma fração do seu valor anterior. Neste tipo de critério, garante-se sempre que a maior distância entre os centros tem um valor determinado. Desta forma, fixa-se um percentual máximo da penúltima distância em comparação com a última e pára-se de alocar centros quando este valor for atingido. Aqui utilizou-se um critério onde a distância atual não poderia ser maior que 0,0125 da anterior.

A função base utilizada foi a função gaussiana, por ser a mais comum em diversos trabalhos consultados e ter a característica de ser bem comportada e localizada. À esta função gaussiana, aplicou-se a modificação proposta por PLATT(1991), que utiliza uma aproximação para inicialização dos pesos da primeira camada. Conforme descrito no trabalho citado, o desempenho da rede, ainda que não de forma tão acentuada, realmente melhora, o que pode ser comprovado pela necessidade de um menor número de iterações para o treinamento da rede. Outros tipos de funções base não apresentariam, entretanto, dificuldades de serem implementadas. As formas destas funções podem ser conferidas no Capítulo 6.

O valor de sigma que aparece tanto na função gaussiana 'pura' quanto na função modificada pelo algoritmo de Platt, é que controla o 'alisamento' da função de interpolação, e é aqui utilizado com valor único para todas as funções base e não conforme o detalhado na equação (6.15), o que impossibilitaria de usar o algoritmo de Platt. Desta forma, valores de sigma que não sejam convenientes para determinado conjunto de padrões, pode dificultar ou até mesmo inviabilizar o treinamento das redes com a precisão requerida.

A influência da variação dos valores de sigma pôde ser comprovada de forma muito clara, neste trabalho. Uma pequena mudança nos seus valores acarreta, às vezes, uma mudança brusca na quantidade de iterações necessárias para se atingir determinado nível de diferença entre os valores desejados e os valores inferenciados pela rede. Diz-se, "às vezes", pelo fato de que parece haver um determinado limite, tanto superior quanto inferior para este tipo de influência. Não se pode, dentro das condições utilizadas para sua variação neste trabalho, afirmar isto de forma categórica pelo fato de se ter utilizado poucos valores de sigma diferentes, em relação ao que seria necessário para se chegar a uma conclusão afirmativa.

O programa está construído de forma a determinar em primeiro lugar os pesos da rede que se situam entre a primeira camada ou de entrada e a segunda camada ou oculta. Este procedimento de determinação é não supervisionado e rápido uma vez que não envolve os valores alvo ou valores de saída. Para a determinação dos pesos da segunda camada é preciso se inverter uma matriz não quadrada, o que é feito aqui utilizando-se uma técnica chamada SVD, a qual decompõe a matriz em partes de forma a poder resolvê-la. Esta técnica está detalhada no trabalho de TEIXEIRA(1997).

Não se levou em conta, no programa, a presença de ruídos nos valores dos padrões devido ao fato de se estar trabalhando com dados de simulação e não com dados reais. Na prática porém, ruídos podem ser causa de erros de ajuste. Sem se levar em conta tais ruídos, o ajuste pode não corresponder à realidade do sistema de processos.

A utilização de padrões para treinamento em uma rede neural com estrutura de ajuste muito rígida, onde cada padrão corresponda a um centro, pode inviabilizar a determinação de padrões não usados no treinamento, principalmente se houver o ajuste de ruídos. Isto é o chamado sobreajustamento.

A teoria da regularização é muito importante em conjuntos de dados cujos padrões de treinamento das redes neurais não tenham um comportamento muito fácil de modelar. O fato de redes com funções radiais base poder fazer com que o ajuste passe exatamente em cada um dos padrões apresentados, pode dificultar a localização de uma saída correta quando o padrão apresentado não tiver sido usado para o treinamento. Isto ocorre porque 'entre' dois padrões dados quaisquer, a forma do ajuste pode ser a mais esdrúxula possível, uma vez que fora dos padrões apresentados, não se pode garantir, para um conjunto genérico e mal distribuído de padrões, que o ajuste guarde proximidade com os valores corretos.

O método de atualização dos pesos é por conjunto de padrões, onde são lidos todos os padrões para depois se fazer a atualização dos mesmos, ao invés de se fazer a leitura de um padrão e em seguida se atualizar os pesos.

Para a inicialização aleatória dos pesos, se utilizou, depois de testes empíricos, uma limitação para que os valores dos pesos não fossem menores que –0,5 e nem maiores que 0,5. Antes desta limitação, os valores iniciais dos pesos atingiam valores muito díspares

o que dificultava alcançar o nível de erro entre o valor desejado e o valor calculado, necessário à saída.

A fim de se evitar trabalhar com variáveis cujas faixas de valores tenham dimensões muito diferentes, como por exemplo vazões e frações molares, o programa normaliza todos os dados de entrada e saída, de forma que ao maior valor é atribuído o valor **1** e ao menor é atribuído o valor **0**.

A determinação da norma euclidiana, a distribuição dos centros e a atualização dos pesos está conforme o exposto no capítulo 6 e na literatura citada. Os algoritmos de determinação dos inversos das matrizes, quadradas e não quadradas foram tiradas do trabalho, já citado de TEIXEIRA(1997).

O critério de parada ou de convergência aqui utilizado leva em conta a soma dos erros quadráticos de cada par entrada-saída e também o erro máximo conseguido entre todos os valores de entrada e saída para cada par. Este último método tem por vantagem, garantir que dentro do intervalo especificado, todos os pesos determinados permitem encontrar a saída com no máximo, o valor de desvio apresentado. Para os EXEMPLOS de 1 a 3, a título de comparação, são apresentados nas tabelas dos resultados numéricos, dois valores para o critério de parada quadrático, o primeiro se refere ao valor alcançado quando se trabalhou apenas com os resultados devidos ao algoritmo 1[37], e o segundo quando se trabalhou com os resultados do algoritmo 2. O erro absoluto máximo de ajuste alcançado, em quase todos os casos aqui apresentados, foi sempre maior que o erro quadrático mas em nenhum caso ficou maior que 0,01% do valor em que o mesmo ocorria. Para os dois últimos EXEMPLOS, foram utilizados apenas o segundo algoritmo devido ao fato de o conjunto de dados ser bem maior tornando impraticável a utilização do *backpropagation* quer 'puro', quer modificado para o treinamento das redes neurais.

---

[37] A partir daqui fica subentendido que o algoritmo 1 é o algoritmo em *backpropagation* com o procedimento de Marquardt-Levenberg e que algoritmo 2 é o procedimento com Funções Radiais Base

7.3 O SOFTWARE: arquivos de entrada/saída

Apesar de não apresentar uma interface muito elaborada a qual é feita apenas para funcionar em DOS, o programa é extremamente simples de operar, não necessitando muitos dados para poder usá-lo.

Na primeira tela que aparece quando se aciona o arquivo executável, existe uma lista de opções as quais apresentam um guia simples de instruções dos próximos passos a serem tomados. A primeira coisa a se fazer, quando nenhuma outra instrução foi dada, é ler um arquivo de dados onde deverão estar os valores dos padrões a serem utilizados para o treinamento da rede.

Teclando-se então a letra (**L**), o programa perguntará qual o nome do arquivo de dados em que deverá buscar os dados mencionados acima. Digita-se o nome do arquivo correspondente, que obviamente, deverá estar no mesmo diretório que o arquivo executável do programa.

Lido o arquivo de entrada de dados, digita-se obrigatoriamente a letra (**T**), a qual está associada à subrotina de treinamento da rede. Feito isto, o programa vai perguntar o nome do arquivo de saída, que é onde ficarão armazenados os valores dos pesos da rede treinada.

Como saída de tela do programa, vai sempre aparecendo os valores dos erros, tanto o quadrático quanto o erro máximo bem como a quantidade de iterações necessárias até aquele ponto. Ao final, ao alcançar a tolerância indicada, o programa indica qual foi o erro quadrático alcançado, o erro máximo absoluto e os respectivos valores percentuais; depois, apertando-se a tecla *enter*, vão aparecendo, utilizando os valores dos pesos calculados, os valores à saída para cada padrão utilizado para o treinamento mostrando, com um asterisco, onde está o padrão em que ocorreu a maior diferença entre o valor desejado e o valor inferenciado.

O inferenciamento acima citado é automático, não precisando ser solicitado ao programa. Para se fazer o teste do programa com dados para os quais a rede não tenha sido treinada, existe o comando **determinar**. Na mesma tela inicial existe uma opção que identifica esta opção com a letra (**D**). Contudo, antes de se escolher esta opção é necessário

se ler o arquivo em que estão os dados os quais serão utilizados para o teste. Aciona-se, desta forma a tecla (**I**), a qual está associada a uma subrotina de leitura de dados para inferenciamento. Teclando (**I**), surgirá uma pergunta na tela, a respeito do nome do arquivo que contém os valores dos pesos da rede neural treinada. O nome do arquivo de entrada, aqui, será o nome do arquivo de saída do treinamento.

Lido o arquivo de dados com os valores dos pesos da rede treinada, aciona-se então a letra (**D**). Feito isto, surgirá na tela uma pergunta acerca do nome do arquivo o qual contém os dados a serem inferenciados, ou ajustados. A resposta indicará se os valores estão de acordo com o esperado (aqui a saída está ajustada para comparar saídas binárias, zeros e uns).

7.3.1 arquivo de entrada

O arquivo de entrada para o treinamento deve conter os valores dos seguintes parâmetros:

nai: número de neurônios de entrada da rede neural,

nad: número de neurônios de saída da rede neural,

nev: número de elementos do vetor de entrada, ou seja o número de padrões utilizado para o treinamento das redes,

tol: é o valor da tolerância, ou o valor do erro quadrático determinado pelo programa.

I(i,j): valores dos dados de entrada, sendo que são lidos todos os padrões referentes ao primeiro neurônio para depois se mudar a leitura para o segundo, terceiro e assim sucessivamente,

D(i,j): valores de saída. São lidos da mesma forma que os valores de entrada, de forma de devem estar dispostos no arquivo de entrada da mesma maneira que o anterior.

Para a utilização da subrotina de inferenciamento, basta se construir um arquivo de dados com os valores a serem inferenciados. Nenhuma outra informação seria necessária pois todos os dados necessários ao inferenciamento serão passados pelo programa principal para o arquivo de saída. Atentar, no entanto, para o fato de que uma rede só pode inferenciar valores se o vetor de entrada tiver o mesmo tamanho do vetor de entrada da rede usada para o treinamento.

RESULTADOS E METODOLOGIAS PARA A DETECÇÃO E O DIAGNÓSTICO DE FALHAS

Existem na literatura alguns exemplos de sistemas em que se procura determinar a ocorrência de possíveis falhas entre os valores de seus parâmetros. Na maioria das vezes, utiliza-se sistemas constituídos de apenas um ou no máximo dois equipamentos. Apenas um exemplo encontrado, utilizava um sistema complexo.

Utilizando o programa de treinamento de redes aqui desenvolvido, treinou-se as redes para cinco EXEMPLOS diferentes a fim de comprovar a aplicabilidade do método proposto. Dos cinco EXEMPLOS aqui descritos, quatro são apresentados na literatura e servem para mostrar o desempenho do algoritmo aqui desenvolvido em comparação com os apresentados pelos respectivos autores. O último é o mais importante deste trabalho pois utiliza os dados do comportamento dinâmico de uma coluna de destilação multicomponente, que é justamente o tema central desta tese e é onde estão descritos os métodos propostos para detecção e diagnóstico de falhas. Para tanto, uma série de investigações de formas de treinamento foram propostas e implementadas a fim de se mostrar a viabilidade de utilização de redes neurais na determinação de ocorrência de falhas em sistemas de processos.

7.4 EXEMPLO 1: SISTEMA DE TRÊS REATORES TANQUE EM SÉRIE

(Ex.: HOSKINS et al. (1988))

O problema consiste em diagnosticar falhas em um sistema simples composto por três reatores-tanques em série. Cada peça do equipamento é operada isotermicamente e com taxa de fluxo constante(sem acúmulo). As variáveis que são usadas para a determinação de falhas são a taxa de fluxo, a temperatura e a concentração do componente (A) e (B) nas correntes 1 e 4. Todas as seis variáveis de estado são monitoradas e seus sensores são assumidos sem erro. Todas as medidas são tomadas em estado estacionário.

A tabela 7.1 lista seis falhas selecionadas, denominadas de A-F, sendo que cada uma afeta a operação do processo considerada neste estudo.

A	Concentração de entrada do componente A	Baixa
B	Concentração de entrada do componente A	Alta
C	Taxa de fluxo de entrada	Baixa
D	Taxa de fluxo de entrada	Alta
E	Temperatura	Baixa
F	Temperatura	Alta

Tabela 7.1: Lista de falhas selecionadas.

Doze medidas foram utilizadas para treinar a rede, sendo onze com erro e uma em condição de normalidade. Os valores das medidas foram conseguidos utilizando o CMS (*Combined Model Simulator*).

O autor não detalha exatamente como faz a simulação (os parâmetros não são especificados) e como faz a contagem do número de iterações, que neste caso não segue o modelo convencional. Tudo isto é de importância secundária pois o que se quer aqui é apenas utilizar os valores do sistema a diferentes estados estacionários a fim de se treinar a rede com o programa desenvolvido.

A tabela 7.2 lista os valores utilizados no treinamento da rede.

FR	Falha	T	C^1_A	C^4_A	C^1_B	C^4_B
18	A	190	0.3	3.18	0.2275	3.252
18	A	190	0.6	2.88	0.3755	3.104
18	B	190	1.3	2.18	0.5990	2.881
18	B	190	1.6	1.88	0.6681	2.812
13	C	190	1.0	2.48	0.4475	3.033
15	C	190	1.0	2.48	0.4777	3.002
22	D	190	1.0	2.48	0.5600	2.920
26	D	190	1.0	2.48	0.5958	2.884
18	E	150	1.0	2.48	0.8960	2.584
18	F	210	1.0	2.48	0.3102	3.170
18	F	230	1.0	2.48	0.1703	3.310
18	Normal	190	1.0	2.48	0.3500	2.990

Tabela 7.2: Dados de processo para o exemplo1.

Sendo que, FR = taxa de fluxo, (ft^3min^{-1}), T = temperatura (^oF), C = concentração ($lbmol^1ft^{-3}$).

Os resultados do treinamento das redes, podem ser acompanhados na Tabela 7.3 a e b, onde estão detalhados a arquitetura da rede e o número de iterações necessárias para se atingir o desvio desejado.

Por causa dos critérios de parada restritivos utilizados, e por causa da saída que temos ser 0 ou 1, para o caso de não-falha ou de falha, respectivamente, todas as redes treinadas conseguem determinar os conjuntos de dados com a respectiva falha em 100% dos casos. Utiliza-se, como critério de generalização, 10% do valor máximo, ou seja se a rede treinada, quando submetida a um conjunto de entrada devolver como saída valores até 0.1, estes serão considerados 0 (zero); e no outro extremo, se devolver até 0.9, serão considerados como sendo 1(um).

A rede treinada tem, como parâmetros fixos, 06 neurônios de entrada e 06 neurônios de saída. Foram utilizadas 12 medidas para treinamento, ou amostras.

MDA(máxima diferença absoluta) (1) = 0,01%

DQ(Diferença quadrática) (2) = 10^{-2} para o primeiro algoritmo e 10^{-6} para o segundo algoritmo com RBF

Os parâmetros variáveis estão resumidos na tabela 7.3 a e b.

N	ETA	ALFA	NNI	NITER1	NITER2	NITER2
1	0.25	0.50	05	7930	NC	-
2	025	0.50	08	9785	NC	-
3	0.25	0.50	10	7446	NC	-
4	0.25	0.50	15	7877	222^{**}	$1613^{(1)**}$
5	0.25	0.50	20	8501	222^{**}	$1613^{(1)**}$
6	0.25	0.50	30	8101	222^{**}	$1613^{(1)**}$
7	0.50	0.50	05	3145	NC	-
8	0.50	0.50	08	5878	NC	-
9	0.50	0.50	10	2253	NC	-
10	0.50	0.50	15	4654	112^{**}	$803^{(1)**}$
11	0.50	0.50	20	5229	112^{**}	$803^{(1)**}$
12	0.50	0.50	30	4984	112^{**}	$803^{(1)**}$
13	0.80	0.50	10	1093	NC	-
14	0.90	0.50	10	1034	NC	-
15	1.00	0.50	10	NC	NC	-
16	0.95	0.50	10	1169	NC	-

Continuação

N	ETA	ALFA	NNI	NITER1	NITER2	NITER2
17	0.92	0.50	10	1065	NC	-
18	0.91	0.50	10	1046	NC	-
19	0.91	0.60	10	1188	NC	-
20	0.91	0.80	10	2129	NC	-
21	0.91	0.90	10	196	NC	-
22	0.91	0.90	05	218	NC	-
23	0.91	0.90	08	204	NC	-
24	0.91	0.90	15	282	11**	81(1)**
25	0.91	0.90	20	NC	11**	81(1)**

Tabela 7.3a: resultados do treinamento de redes neurais a diversos parâmetros.

Onde,

NC, indica que com os parâmetros dados, o erro máximo permitido não foi alcançado para o algoritmo 1,

NNI é o número de neurônios da camada intermediária,

NITER é o número de iterações, sendo que 1 e 2 indicam os algoritmos anterior, *backpropagation*, com atualização dos pesos segundo o algoritmo de Marquardt-Levenberg e atual, com RBF. E dentre os de número 2, a primeira coluna sempre indica o número de iterações necessárias para a convergência do algoritmo com as mesmas condições que o algoritmo 1, são portanto os de comparação mais direta. Os NITER2 da segunda coluna se referem ao critério que considera erros quadráticos menores,

ALFA e ETA são parâmetros do GDR.

** indica que a rede está com o número máximo de centros,

ou (2) como índice indica qual dos dois critérios de parada foi o pior. (1) MDA, (2) DQ.

SIGMA = 0,0125 para a rede com RBF (neste exemplo).

Para o caso de utilização do algoritmo 1, a melhor combinação de parâmetros ocorre no conjunto apresentado pelo número 21, o qual representa a rede treinada utilizando o menor número de iterações para o critério de convergência desejado ($DQ=10^{-2}$). Para o caso de utilização do algoritmo 2, os melhores resultados estão resumidos na linha 24 da mesma tabela. Veja-se que para o mesmo critério de parada anterior ($DQ=10^{-2}$) foram necessárias apenas 11 iterações para se chegar ao mesmo resultado (contra 196). Mesmo no caso em que o critério de parada adotado foi muito mais rigoroso, o número de iterações ainda continuou abaixo do valor conseguido quando da utilização do primeiro algoritmo. 100% das redes treinadas apresentaram a resposta correta quando solicitadas com os conjuntos de entrada conforme os utilizados para o treinamento. A pequena quantidade de dados não permitiu maiores variações na fase de inferenciamento, como por exemplo, separar uma parte do conjunto de padrões a fim de usá-lo para validação do treinamento.

A Tabela 7.3b, resume alguns outros resultados para otimização dos parâmetros quando se modificam os valores de ALFA, SIGMA e ETA.

SIGMA	ETA	ALFA	NITER(10^{-6})	NITER(10^{-2})
0.0125	0.99	0.90	$73^{(1)**}$	9^{**}
0.0125	0.99	0.95	$51^{(1)**}$	17^{**}
0.0125	0.99	0.97	$137^{(1)**}$	94^{**}
0.0125	0.99	0.94	$52^{(1)**}$	5^{**}
0.0125	0.99	0.93	$49^{(1)**}$	5^{**}
0.00125	0.99	0.93	$49^{(1)**}$	5^{**}
0.000125	0.99	0.93	$49^{(1)**}$	5^{**}
0.125	0.99	0.93	$117^{(1)**}$	5^{**}

Tabela 7.3b: Outros resultados de otimização dos parâmetros de treinamento de uma rede neural com RBF. Os valores de SIGMA foram escolhidos empiricamente partindo-se do valor inicial 10 e a cada tentativa, dividindo-se por dois. A partir de 1,25, foi-se dividindo por 10.

7.5 EXEMPLO 2: SISTEMA DE UM REATOR CONTROLADO (PI)

Considere um processo com um controlador PI em *feedback* no qual heptano é convertido cataliticamente a tolueno, operando próximo às condições de estado estacionário. Nenhum evento catastrófico ocorre porque qualquer falha no processo é aliviada pelo controlador. Deste modo, nenhuma mudança de temperatura ocorre no reator devido a uma falha. A taxa da reação é controlada pela temperatura do reator. É assumido que o controlador tem capacidade de se auto-monitorar podendo portanto considerá-lo sem falhas para os propósitos deste trabalho.

As possíveis falhas podem ser detalhadas conforme a tabela 7.4

Falha 1	Deterioração do desempenho do catalisador, queda do valor de k_0
Falha 2	Incrustação na parede do reator, queda do valor do coef. de troca térmica, h
Falha 3	Incrustação no trocador de calor, queda do valor do coef. global, h
Falha 4	Vazamento na linha de alim. do reator, queda na taxa de alim volum., q
Falha 5	Vazamento na linha de alim. do reciclo, queda na taxa de alim. Reator, q

Tabela 7.4 Falhas e suas possíveis conseqüências.

Estas falhas são detectadas através do monitoramento da concentração de saída do heptano, da temperatura de saída do aquecedor e do sinal de saída do controlador.

O conhecimento das relações entre as medidas e as causas de falhas pode muitas vezes ser conseguida através do acompanhamento diário do processo.

A tabela 7.5 mostra as relações entre os valores das três medidas (normalizadas pela divisão do valor da medida pelo valor normal da mesma). Os valores listados para as três medidas e os valores normais das variáveis foram obtidos tomando-se uma média de 1000 amostras geradas por um simulador. As falhas foram "geradas" decrescendo-se os valores dos parâmetros k_0, h, h', q e q'.

N	Causa	S_n^*	T_h^*	C_{c7h8}^*
1	0.90 k_0	0.98	0.99	0.95
2	0.90 h	1.07	1.02	1.00
3	0.90 h	1.11	1.00	1.00
4	0.90 q	0.95	0.99	1.05
5	0.90 q	0.90	1.00	1.00

Tabela: 7.5 Conseqüências das falhas nas variáveis medidas em estado quase-estacionário.

A rede utilizada neste exemplo tinha como parâmetros fixos, 03 nós de entrada e 05 nós de saída sendo que os primeiros correspondem aos valores das variáveis mudadas e os últimos correspondem às falhas detectadas. Aqui, novamente o critério de falha ou não-falha é representado por algarismos binários 1 e 0, respectivamente. Em todos os casos utilizados para o treinamento, foi possível recuperar o diagnóstico conforme o previsto pelo treinamento da rede. (Funcionou para 100% dos casos). As tabelas 7.6a e 7.6b, resumem os principais resultados do treinamento.

N	ETA	ALFA	NNI	NITER1	NITER2	NITER2
1	0.25	0.50	05	11375	169	1600
2	025	0.50	08	5747	169	1600
3	0.25	0.50	10	5268	169	1600
4	0.25	0.50	15	4134	169	1600
5	0.25	0.50	20	3497	169	1600
6	0.25	0.50	30	2795	169	1600
7	0.25	0.50	40	2466	169	1600
8	0.50	0.50	05	5018	85	1600
9	0.50	0.50	08	3162	85	798
10	0.50	0.50	10	2657	85	798
11	0.50	0.50	15	2035	85	798
12	0.50	0.50	20	1667	85	798
13	0.50	0.50	30	1335	85	798
14	0.50	0.50	40	1216	85	798

Continuação

N	ETA	ALFA	NNI	NITER1	NITER2	NITER2
15	0.75	0.50	05	3286	57	532
16	0.75	0.50	08	2298	57	532
17	0.75	0.50	10	1670	57	532
18	0.75	0.50	15	1304	57	532
19	0.75	0.50	20	1140	57	532
20	0.75	0.50	30	902	57	532
21	0.75	0.50	40	836	57	532
22	0.90	0.50	30	779	48	443
23	0.95	0.50	30	748	45	419
24	0.98	0.50	30	727	44	407
25	0.98	0.75	30	885	22	201
26	0.98	0.85	30	NC	12	114
27	0.98	0.85	30	NC	-	-
28	-	-	-	-	11	112
29	-	-	-	-	6	61
30					36	90

Tabela 7.6a: Resultados do treinamento da redes para o exemplo 2. Para este caso, o número de centros sempre foi máximo e o MDA foi sempre o pior resultado logo foi sempre o critério de parada do treinamento da rede.

Para outros valores dos parâmetros de treinamento ALFA, SIGMA e ETA tem-se os seguintes resultdos para o treinamento das redes neurais:

SIGMA	ETA	ALFA	NITER1(10^{-2})	NITER2(10^{-6})
0.0125	0.99	0.93	65	16
0.00125	0.99	0.90	61	6
0.000125	0.99	0.90	61	6
0.125	0.99	0.90	61	6
1.25	0.99	0.90	NC	NC

Continuação				
SIGMA	**ETA**	**ALFA**	**NITER1(10^{-2})**	**NITER2(10^{-6})**
0.0125	0.99	0.89	7	7

Tabela 7.6b: Resultados do estudo de otimização dos parâmetros de treinamento da rede com RBF. O número de centros sempre foi máximo e o MDA foi sempre o pior resultado logo foi sempre o critério de parada do treinamento da rede.

Como no exemplo anterior, o algoritmo 2 (com RBF) mostrou-se mais eficiente que o algoritmo 1. Uma quantidade muito menor de iterações foi necessária para o treinamento da rede neural.

7.6 EXEMPLO 3: REATOR TANQUE EM SÉRIE COM COLUNA DE DESTILAÇÃO

O processo envolvido neste exemplo é constituído por um reator CSTR encamisado, onde ocorre uma reação irreversível, exotérmica de primeira ordem e uma coluna de destilação. O reator dispõe de três malhas de controle, os quais controla a temperatura de saída o *holdup* do reator e a concentração de saída dos produtos.

A corrente efluente do reator contém uma mistura binária de A e B e alimenta a coluna de destilação onde é separada numa corrente destilada contendo 98% de A e numa corrente que sai pela base contendo 98% de B. A coluna tem um controlador PI que controla as composições do topo e da base manipulando a taxa de refluxo e a taxa de vapor. A tabela 7.7 apresenta as disfunções selecionadas que podem ocorrer na planta.

FALHA	**Símbolo**
Alta taxa de entrada do fluxo no reator	**F1**
Baixa taxa de entrada do fluxo no reator	**F2**

Continuação

FALHA	Símbolo
Alta concentração de A à entrada do reator	**F3**
Baixa concentração de A à entrada do reator	**F4**
Alta temperatura de entrada do fluxo no reator	**F5**
Baixa temperatura de entrada do fluxo no reator	**F6**
Falha no controlador da base	**F7**
Falha no controlador do destilado	**F8**

Tabela 7.7: Conjunto de possíveis falhas analisadas na detecção de falhas.

A rede foi treinada utilizando-se treze conjuntos entrada-saída para a rede. Os dados utilizados foram resumidos na tabela 7.8.

N	Falha	C	T	V	F	T_j	F_j
1	F1(+15%)	0.2575	600.66	48.6	49.0	595.0	52.5
2	F2(-15%)	0.2307	599.17	47.4	34.0	594.2	49.6
3	F3(+15%)	0.2520	602.82	47.6	37.3	597.1	61.2
4	F4(-15%)	0.2315	597.74	48.4	43.7	592.9	37.9
5	F5(+15%)	0.2020	608.44	48.0	40.0	598.9	83.7
6	F6(-15%)	0.2991	589.73	48.0	40.0	588.9	8.8
7	F1(+5%)	0.2494	600.24	48.2	42.0	594.8	50.8
8	F2(-5%)	0.2405	599.73	47.8	38.0	594.5	48.8
9	F3(+5%)	0.2480	600.99	47.9	38.8	595.2	53.9
10	F4(-5%)	0.2414	598.94	48.1	41.2	594.1	45.7
11	F5(+5%)	0.2296	602.96	48.0	40.0	597.2	61.8
12	F6(-5%)	0.2617	597.83	48.0	40.0	592.9	37.2
13	Normal	0.2450	600.00	48.0	40.0	594.6	49.9

Tabela 7.8: Dados de entrada para o treinamento da rede.

O treinamento desta rede, utilizando os algoritmos iniciais apenas com o *backpropagation* com o algoritmo de Marquardt-Levenberg demandou um grande esforço computacional. Foram testadas diversas composições de parâmetros mas em nenhum caso o treinamento ocorreu com um número de iterações menor que 150.000. O autor do trabalho apresenta dados de número de iterações que confirmam a dificuldade de treinamento de redes como esta. Segundo o autor, utilizando 10 camadas ocultas, a rede precisou de mais de 70.000 para alcançar a convergência. Como o critério aqui adotado foi mais restritivo, o número de iterações só poderia mesmo ser maior. Aqui, o fato de não se encontrar um conjunto ótimo de parâmetros para a rede neural impediu de se chegar a resultados razoáveis.

Utilizando-se, contudo, o segundo algoritmo com as funções radiais, foi perfeitamente possível se chegar a números de iterações muito menores e alcançar valores de erros quadráticos e diferenças absolutas máximas da mesma ordem de grandeza dos expressos para os demais exemplos anteriores. A tabela 7.9 mostra os resultados do treinamento a diversos valores de parâmetros da rede neural.

Este exemplo mostra bem os problemas encontrados para se determinar os parâmetros de uma rede neural. Com algumas combinações de dados pode não ser possível mapeá-las, quando isto é possível, consegue-se a custa de um grande esforço computacional.

SIGMA	ETA	ALFA	NITER(10^{-2})	NITER(10^{-6})
0.0125	0.90	0.90	11	82
0.0125	0.95	0.90	10	77
0.0125	0.98	0.90	10	75
0.0125	0.99	0.90	10	74
0.0125	0.99	0.95	15	51
0.0125	0.99	0.97	82	126
0.0125	0.99	0.96	30	65
0.00125	0.99	0.95	15	51
0.000125	0.99	0.95	15	51
0.125	0.99	0.95	31	462

Continuação

SIGMA	ETA	ALFA	NITER(10^{-2})	NITER(10^{-6})
1.25	0.99	0.95	NC	NC

Tabela 7.9: Resultados do treinamento de uma rede neural com RBF. O número de centros sempre foi máximo e o MDA foi sempre o pior resultado logo foi sempre o critério de parada do treinamento da rede.

7.7 EXEMPLO 4: DADOS DA SIMULAÇÃO DE UMA COLUNA DE DESTILAÇÃO MULTICOMPONENTE EM BATELADA

Os dados utilizados para se testar a rede com grandes conjuntos de dados foi retirada do trabalho de PEDROSA(1998), onde se simula o comportamento uma coluna de destilação em batelada e se mede o comportamento dinâmico de algumas variáveis a fim de se fazer o controle da mesma. No sistema são medidas cinco variáveis: as temperaturas de topo e fundo da coluna, a razão de refluxo interna (R/R+1), e as composições de topo de dois componentes (hexano e ciclohexano).

Foram utilizados cerca de 3800 padrões[38] ou amostras para o treinamento das redes e devido a isto, é impossível apresentar aqui os dados utilizados para treinamento. Apenas os resultados são sumarizados na Tabela 7.10. Todos os casos deste exemplo foram estudados utilizando-se a rede com RBF[39]. Dois procedimentos diferentes de inicialização de pesos foram considerados. São os procedimentos de inicialização de pesos de forma randômica e de inicialização segundo o procedimento de PLATT(1991) o qual está apresentado na equação (6.50a).

---

[38] Estes 3800 padrões foram arranjados de forma a se conseguir treinar redes com diversas arquiteturas conforme o mostrado na tabela 7.10.

[39] Apenas os resultados otimizados (ou seja, os melhores resultados) estão sendo apresentados nesta tabela.

NCENT	NITE_R1	NITE_R2	SIGMA	ETA	ALFA	DQ	ARQUIT
100	121	81	0.0125	0.99	0.89	0.000001	10/1/1000
100	94	70	0.0125	0.99	0.89	0.000001	15/5/1000
100	397	297	0.0125	0.99	0.92	0.000091	3/2/200
100	699	417	0.0125	0.99	0.91	0.00091	3/2/1200
100	42	36	0.0125	0.99	0.05	0.000001	10/5/1500

Tabela 7.10: Resultados do treinamento de redes com RBF com dados de um simulador de uma coluna de destilação em batelada.

Onde NCENT é igual ao número de centros da rede neural, o qual foi imposto o valor máximo 100, por causa das dificuldades de treinamento; NITE_R1 é o número de iterações usando o procedimento de inicialização dos pesos de forma randômica, NITE_R2 é o número de iterações usando o procedimento de inicialização dos pesos pelo procedimento proposto por PLATT(1991); DQ é a diferença quadrática e ARQUIT é a arquitetura da rede com número de neurônios de entrada/neurônios de saída/ total de padrões utilizados no treinamento. A grande quantidade de dados neste exemplo possibilitou separar um conjunto de dados os quais não foram utilizados no treinamento, para funcionar como conjunto de prova para a 'extrapolação'. Foram separados de forma mais ou menos equidistantes, 100 valores dentre todos os valores do conjunto de dados. Conforme descrito no capítulo 2, o fato de a distribuição dos padrões ser mais ou menos contínua, possibilitou inferenciar de forma correta as saídas para os 100 valores apresentados.

Como pode ser observado por estes 4 EXEMPLOS, o objetivo de se implementar um algoritmo confiável e de grande versatilidade foi alcançado. Em todos os casos foram conseguidas melhoras significativas sendo que o caso exposto no exemplo 3 deve ser olhado com especial atenção, uma vez que com o procedimento anterior o treinamento nem mesmo foi possível. Finalmente para o exemplo 4, pode-se notar que em sistemas com um conjunto mais elevado de padrões também foi possível se atingir os resultados buscados.

7.8 EXEMPLO 5: DADOS DA SIMULAÇÃO DE UMA COLUNA DE DESTILAÇÃO EM ESTADO DINÂMICO

É neste exemplo que estão os principais resultados deste trabalho. Conforme está descrito no Capítulo 4 foi desenvolvido um programa computacional que simula o comportamento dinâmico de uma coluna de destilação multicomponente. O objetivo aqui é testar a viabilidade de utilização das redes neurais na detecção e diagnóstico de falhas em processos complexos, e a importância de se conhecer os estados intermediários de um sistema entre um estado e outro. A literatura pesquisada, apenas traz como exemplos de aplicação das técnicas, dados referentes ao estado estacionário dos sistemas. Como explicado em capítulos anteriores[43], este procedimento não é suficiente na prática, principalmente quando o que se deseja é detectar/diagnosticar a falha o mais rápido possível depois que ela tenha ocorrido, o que implica ter que manipular dados do processo que muito provavelmente não se encontra em estado estacionário. Os três primeiros exemplos descritos neste Capítulo são exatamente uma amostra do que se encontra presente na literatura pesquisada.

O grande problema encontrado, quando se pensou em trabalhar com sistemas de maiores dimensões, está relacionado à dificuldade de treinamento das redes. Deste modo, foi preciso pesquisar novas arquiteturas e novas técnicas[41] de implementação de algoritmos mais eficientes que possibilitassem justamente esses ajustes requeridos pelo problema apresentado. A sensível melhora conseguida pode ser comprovada quando se compara os números de apresentações[42] necessárias para o treinamento quando se utilizou os algoritmos descritos pelos autores estudados e pelo algoritmo aqui implementado. Houve casos, como por exemplo o do exemplo 3, em que o autor afirma terem sido necessárias

---

[40] Ver capítulo 2

[41] Ver capítulos 3, 5 e 6.

[42] Neste trabalho são usados de forma indistinta os termos apresentação e iteração. A rigor, nenhum dos dois é totalmente apropriado, contudo, na falta de um termo mais específico, estas duas formas são aqui tomadas como equivalentes.

mais de 70.000 iterações para se conseguir o treinamento da rede e no entanto com o algoritmo implementado aqui, em nenhum caso foi necessária mais que 500 apresentações.

Visto por este lado, pode parecer que o algoritmo de treinamento implementado neste trabalho, seja a solução para 'todos' os problemas concernentes a detecção/diagnóstico de falhas utilizando redes neurais. Em absoluto, pois as redes neurais se por um lado apresentam-se como poderosas ferramentas de aproximação e mapeamento de superfícies, ainda carece de estudos mais aprofundados para um perfeito entendimento de suas potencialidades. O aumento da dificuldade de treinamento das redes conforme se aumenta o tamanho do problema é realmente espantosa. Isto pode ser comprovado quando se verifica os números reunidos nas tabelas seguintes, onde estão sumarizados os resultados do treinamento de redes para detecção/diagnóstico de falhas na coluna de destilação.

O objetivo em sistemas de controle de processos deve ser o de determinar quando ocorrem as falhas (detectar) bem como onde e porque elas ocorrem(diagnosticar), o que na verdade são fatos que estão ligados, o mais rápido possível após as mesmas terem ocorrido. Deste modo, o ideal seria conseguir encontrar os possíveis erros assim que eles ocorressem, ou pouco depois, sem que para isto fosse necessário treinar as redes com longos períodos de funcionamento, ou de outra forma sem necessitar se conhecer suas características no novo estado estacionário, depois de ocorrida a falha, isto considerando-se que a falha não causaria oscilações no sistema, impedindo-o de encontrar uma nova condição estacionária. O tempo de funcionamento do sistema com falha utilizado para o treinamento das redes influirá diretamente na dificuldade em se ajustar os parâmetros necessários das mesmas. Quanto maior for o número de estados e o número de variáveis necessárias para se conseguir o ajuste desejado, pior será o desempenho das redes.

O número de exemplos utilizados no treinamento da rede também influenciará no procedimento de verificação de falhas.

Imagine que se tome um determinado número de amostras das condições do sistema durante um período de tempo. Dentro deste intervalo, quanto menos tempo decorrer entre uma amostra e outra, melhor será a representação do comportamento dinâmico do sistema. Por outro lado, quanto menor a 'distância' entre uma amostra e outra, maior a quantidade de amostras e consequentemente maior a dificuldade de treinamento da rede. A fim de se contornar isto, pode-se diminuir o intervalo de tempo utilizado para o

treinamento, o que faz diminuir o número de amostras, tendo como inconveniente a aplicação do procedimento de verificação de falhas a intervalos menores. Contudo, esta diminuição pode levar à não ocorrência de modificações detectáveis pelo procedimento de controle.

Desta forma, é necessário se encontrar um equilíbrio entre os diversos parâmetros que regem a detecção/diagnóstico de falhas utilizando redes neurais a fim de se chegar a um bom termo de relação entre confiabilidade de detecção versus dificuldade de treinamento das redes.

Como características de funcionamento, as redes neurais se apresentam, na maioria dos casos, difíceis para serem treinadas; contudo, uma vez feito este treinamento, o inferenciamento de um determinado estado previamente apresentado, ou não, no treinamento[43], é rápido, e tem tempo de execução desprezível, em relação à dinâmica de operação do sistema. Desta forma, a utilização do sistema de inferenciamento ou de verificação não se apresenta como obstáculo ao procedimento de detecção/diagnóstico de falhas, podendo ser feito a curtos intervalos durante o monitoramento da planta química.

Um defeito do processo é detectado quando um sintoma observado 'cruza' um determinado *threshold*. O tempo em que o sintoma alcança o *threshold* depende da natureza e da magnitude da falha bem como dos limites do *threshold*. Não é possível se ter em mente o tempo exato em que a falha pode ocorrer, e consequentemente, o intervalo após o qual as leituras dos sensores serão tomadas para inferenciamento. Em diagnóstico em tempo real, a metodologia não pode depender da disponibilidade dos dados dos sensores a um específico intervalo de tempo desde o começo da falha. De qualquer forma, não é factível manter um conjunto de dados extensivo para vários níveis percentuais de desvios dos parâmetros. Deste modo, é importante que a metodologia de diagnóstico seja o menos dependente, senão completamente independente, do tempo de início da falha e da extensão da mesma.

Como forma de se garantir que o estado atual do sistema está sempre dentro do universo utilizado para o treinamento da rede, pode-se utilizar como tempo de verificação, um tempo sempre menor que o tempo utilizado na amostragem para o treinamento. Como sugestão de sistematização, pode-se adotar como tempo de verificação, ou inferenciamento, um valor igual à metade do tempo utilizado na tomada de amostras ou padrões para o

treinamento. Deste modo, qualquer que seja o momento de ocorrência da falha, a partir de uma primeira verificação, tem-se um conjunto de valores como sendo igual ao utilizado no treinamento, o que dentro das condições aqui propostas, garante 100% de acerto quanto a detecção da falha, caso a mesma seja na extensão para a qual a rede tenha sido treinada. Para outras extensões, este percentual é sempre menor que 100%.

Graficamente pode-se representar este procedimento por

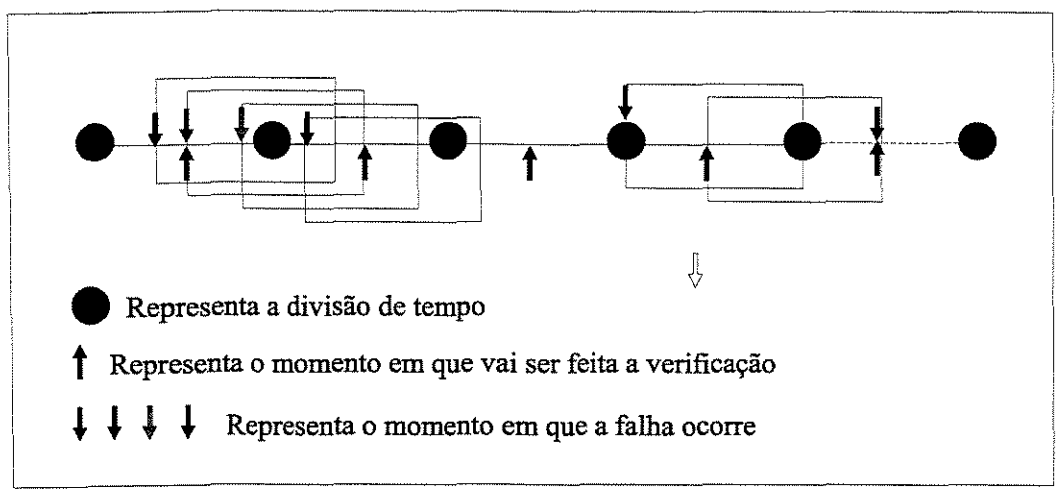


Figura 7.1 Comportamento dinâmico do processo representado no tempo.

As setas inferiores representam o momento da verificação da condição do sistema falha/não-falha. As setas superiores representam esquematicamente o possível momento de ocorrência de uma falha, após a entrada em funcionamento do sistema. Os retângulos iniciam-se sempre no ponto em que aparece a falha, pois o treinamento da rede contém os valores dos parâmetros controlados desde o início da situação de falha, e mostra que sempre, neste método de inferenciamento proposto, vai haver pelo menos uma inferenciação (setas inferiores) capaz de detectar a falha. Caso a falha ocorra no limite do tempo de inferenciamento, tem-se, teoricamente, dois momentos possíveis para se detectar esta falha.

Na possibilidade de se conseguir detectar corretamente a falha está a justificativa desta tese. Utilizando-se dados do comportamento dinâmico do sistema (com e sem falhas)

---

[43] Ver Capítulo 2.

para treinamento das redes neurais em algum momento após acontecer uma determinada falha (em diversos níveis), o dispositivo inferenciador deverá encontrar um padrão conhecido igual ou suficientemente próximo (numericamente), do valor dos parâmetros utilizados para o monitoramento da planta, naquele momento, de forma a permitir uma perfeita associação, a qual permita a correta detecção e o diagnóstico da falha acontecida.

É por este motivo que a divisão do tempo para a tomada das amostras é muito importante. É da perfeita equalização entre a tomada da amostra e da utilização do dispositivo inferenciador que se terá um sistema de detecção/diagnóstico eficiente.

As possíveis falhas que poderiam ocorrer no funcionamento da coluna de destilação aqui desenvolvida são:

F	DESCRIÇÃO DA FALHA	CAUSA POSSÍVEL DA FALHA
F1	***Mudança na razão de refluxo***	Como conseqüência, por exemplo de uma falha na válvula da tubulação por onde o líquido condensado retorna à coluna.
F2	***Mudança na temperatura de alimentação***	Em conseqüência, por exemplo, de uma falha na válvula de vapor do pré-aquecedor da alimentação.
F3	***Mudança na taxa de fornecimento de calor ao refervedor***	Em conseqüência, por exemplo, de falha na válvula de vapor do refervedor.
F4	***Mudança na concentração de alimentação***	Em conseqüência de falha no equipamento de geração da mistura a ser destilada.
F5	***Mudança na quantidade de alimentação***	Em conseqüência, por exemplo de falha na válvula de controle de vazão de alimentação.

Tabela 7.11: Descrição das falhas e suas possíveis causas, para a coluna de destilação estudada. Para sistemas mais complexos, novas causas de falhas podem ser adicionadas.

Nas Tabelas 7.12 a 7.24 encontram-se dados referentes ao número de apresentações necessárias para se treinar as redes quando se utiliza os dados de seis variáveis nos **05**(cinco)/**10**(dez)/**20**(vinte)/**30**(trinta) primeiros minutos, após a ocorrência da falha. Os valores das variáveis utilizadas para treinamento da rede, e monitoramento da planta, em todos os CASOS deste EXEMPLO, são:

- **a temperatura do topo da coluna,**
- **a temperatura do fundo da coluna,**
- **a vazão dos produtos de topo da coluna,**
- **a vazão dos produtos da base da coluna,**
- **as frações molares de metanol no topo e,**
- **as frações molares de metanol no fundo**.

Não serão aqui apresentados os valores numéricos destas variáveis devido à grande quantidade de números, o que tomaria muito espaço e não seria de interesse prático. O comportamento de cada uma destas variáveis pode ser acompanhado, no entanto, através dos gráficos apresentados no Capítulo 4, para alguns níveis de perturbação.

Para todo este EXEMPLO serão utilizados somente o número de iterações necessárias ao treinamento das redes quando se utilizou o procedimento de inicialização dos pesos conforme proposto por PLATT(1991). Isto porque este procedimento se mostrou superior em desempenho quando confrontado com a inicialização randômica[44]. O algoritmo foi escrito de forma a limitar em 100, no máximo, 100 centros o que correspondem a 100 neurônios na camada intermediária. As arquiteturas são do tipo 6/(5+1), 6/2 e 6/1, ou seja, 6 neurônios de entrada, correspondentes às variáveis monitoradas, 5 neurônios de saída (mais um correspodente à situação normal), correspondentes às 5 falhas possíveis no sistema. No caso de um ou dois neurônios de saída, sem correspondência a nenhuma falha em particular mas apenas correspondente a uma situação de falha ou de não-falha.

---

[44] Ver EXEMPLO 4.

Devido à falta de critério de ajuste dos parâmetros de treinamento das redes, propõe-se testar formas diferentes de se proceder a este ajuste. Como são três[45] os parâmetros a serem ajustados, pode-se combinar os três de forma a começar ora pelo ajuste de um ora pelo ajuste de outro. As tabelas estão organizadas de forma a permitir, apenas com uma consulta rápida, qual a seqüência testada. Basta notar qual dos parâmetros vem primeiro (de cima para baixo) destacado com um sombreado dentro da célula que o mesmo ocupa na tabela. Para se alcançar o ajuste final dos pesos o objetivo é sempre diminuir ao mínimo possível o valor de MDA, o qual, quando a quantidade de treinamento necessária não é impraticável, sempre alcança um patamar mínimo o qual não é possível ultrapassar com a configuração testada.

Os passos seguidos, de forma a se criar uma maneira sistemática de ajuste dos parâmetros são os seguintes:

1. escolhe-se de forma empírica, mas, mais ou menos, próxima dos valores esperados para os parâmetros 'ótimos' (isto foi conseguido com os testes dos exemplos anteriores , quando se utilizou os dados presentes na literatura citada), os valores de cada um dos parâmentros;
2. fixa-se todos os parâmetros, exceto um;
3. escolhe-se um valor 'alto' para a tolerância de forma a não serem necessárias muitas iterações nesta fase de testes;
4. vai-se modificando o valor do parâmetro, não fixo, até que o menor número de iterações seja alcançado;
5. se o número de iterações coincidir com o anterior e o valor do MDA não apresentar diferença significativa em favor de um dos dois, fica-se sempre com o penúltimo valor para a variável em questão. De outra forma dá-se preferência a este último;
6. feito o ajuste de um parâmetro, fixa-se este e modifica-se um dos outros (conforme a ordem escolhida), e retorna-se ao passo 3.

---

[45] Porque a quantidade de centros não é escolhida previamente. O programa faz a escolha do número de

Deve ser notado, entretanto, que esta forma de ajuste proposta é apenas uma das formas de atualização dos parâmetros, de modo a se poder comparar os diversos resultados do treinamento, operação que seria dificultada, senão impossibilitada, caso se utilizassem critérios aleatórios. A preocupação em dar esta justificativa pode ser comprovada quando se olha o item 7.5.4 do CASO 1, onde se conseguiu um ajuste 'aleatório' com o mesmo número de iterações e apenas levemente menos preciso que o apresentado no item 7.5.3 do mesmo CASO 1. Apesar disto, ainda é possível se pensar que uma forma que possa ser reproduzida passo a passo é mais indicada na investigação científica que a busca não sistemática dos valores ótimos.

Para o acompanhamento dos símbolos utilizados nas tabelas adotou-se a mesma convenção que foi utilizada para os exemplos anteriores. Pelo fato de alguns serem exclusivos deste EXEMPLO, serão todos aqui redefinidos.

- **SIGMA, ETA e ALFA** são parâmetros do algoritmo de treinamento. Empiricamente, sabe-se que os valores de ETA e ALFA devem ser tais que $0 < (ETA ; ALFA) < 1$ e devem ser os maiores possíveis sem que cause oscilação no treinamento da rede;

- **TOL** é a tolerância que se aceita para o valor da diferença quadrática dos erros de ajuste;

- **DQ** é a diferença quadrática dos erros do treinamento;

- **MDA** é a máxima diferença absoluta dentre todos os exemplos utilizados e o respectivo valor calculado pela rede ajustada, o valor do inferenciamento; é o nível de garantia que para todos os valores utilizados no treinamento, o inferenciamento é correto.

- **NME** indica o número do neurônio e o número do exemplo em que acontece o maior erro absoluto, serve para indicar onde está o pior ajuste da rede. É útil para se prever em que região do conjunto de dados estão os piores do inferenciamento;

---

centros, que no caso atual está limitado a 100, para não comprometer muito o desempenho da rede.

- **MXER** é o valor do erro percentual para a variável com maior erro absoluto[46]. É calculado dividindo-se o erro calculado pelo valor esperado e multiplicando-se por 100;

- **NITER**[47] indica o número de iterações necessárias para se atingir o treinamento da rede nas condições desejadas.

Os valores destacados com o sombreamento se referem aos valores 'ótimos[48]'.

A melhor combinação de resultados se dá quando se tem os mais baixos valores de MDA/DQ/NITER. Apesar de não se poder olhar isoladamente para apenas um dos aspectos, tem-se no entanto, que dentre os três, o MDA é o mais importante pois é ele quem determina, em última instância, o nível de precisão alcançado. O NITER indica o 'custo' computacional para se atingir este objetivo.

Aqui o ajuste dos parâmetros, conforme detalhado acima, foi conseguido atribuindo-se a TOL, um valor 'alto', de tal forma que a convergência pudesse ser mais rápida. A partir de então muda-se um dos parâmetros deixando os demais fixos até se conseguir o melhor valor com o que se está variando. Atingido este valor, parte-se para o ajuste do próximo e assim sucessivamente até encontrar-se uma combinação, que se não é a ótima[49], é a que reúne as melhores características de precisão de ajuste e número de iterações. A partir de então parte-se para melhorar a precisão de ajuste em detrimento do número de iterações. Não é possível se falar em ajuste ótimo, no sentido estrito da palavra, pois a rigor seria necessário utilizar valores dos parâmetros com mais casas decimais, que o que foi feito aqui, exceto para o caso de sigma, com no máximo duas casa decimais.

Não é preciso se preocupar quanto à significância física dos valores descritos nestas tabelas, pois os mesmos se traduzidos em termos de valores reais de medida, que é na verdade o que representam em última análise, não podem ser medidos com tal precisão, mas como se trata de ajuste, quanto maior for a precisão utilizada, mais próximo do valor

---

[46] Notar que este valor pode não corresponder ao maior erro percentual porque, apesar de os valores das variáveis estarem normalizados, a diferença entre dois valores de duas variáveis pode chegar, no limite a 100%, no caso presente, 0(zero) ou 1(um).

[47] Não esquecer que cada iteração é contada quando todos os exemplos foram apresentados à rede.

[48] Ótimos no sentido de serem os valores que apresentaram o menor necessidade de iterações e/ou menor MDA, quando se utiliza um procedimento sistemático de ajuste dos parâmetros.

[49] Ver, por exemplo, o CASO 1, ITEM 7.8.1, abaixo.

desejado se estará quando se fizer a transformação uma vez que todos os valores utilizados para o treinamento das redes são normalizados.

A seguir, os quatro primeiros CASOS são feitos segundo as condições acima descritas.

Os resultados numéricos estão abaixo detalhados, CASO a CASO.

Nos quatro primeiros CASOS, pretende-se analisar a quantidade de amostras necessárias para se fazer uma correta detecção/diagnóstico de falhas. Os tempos de amostragem foram a cada um minuto e foram estudadas as características dinâmicas do sistema com 5/10/20/30 minutos de funcionamento após o aparecimento de cada falha, o que resulta em 126, 251, 501 e 751 padrões para o treinamento da rede. Foram empregados 7 níveis de falhas para cada um dos parâmetros analisados, sendo que, em todos os CASOS, apenas 5 níveis foram utilizados para o treinamento, restando padrões referentes a dois níveis para se fazer a validação do treinamento. Além da análise do desempenho do procedimento de detecção/diagnóstico, estão apresentados gráficos que mostram o desempenho do algoritmo de treinamento das redes neurais. Para se testar o mecanismo de ajuste dos parâmetros das redes neurais, aqui proposto, os CASOS 1 e 2 trazem cada conjunto de dados treinados com seqüências de ajuste diferentes para que se possa determinar a melhor delas.

7.8.1 CASO 1

7.8.1.1: 05 MINUTOS/AJUSTE AES.

N	SIGMA	ETA	ALFA	TOL	DQ	MDA	MXER	NME	NITER
1	0.125	0.90	0.80	0.13	0.1290	0.1221	13.56%	2/7	476
2	0.125	0.90	0.82	0.13	0.1280	0.1203	13.37%	2/7	439
3	0.125	0.90	0.85	0.13	0.1270	0.1282	14.24%	2/10	384
4	0.125	0.90	0.90	0.13	0.1230	0.1411	15.68%	2/10	294
5	0.125	0.90	0.91	0.13	0.1230	0.1488	17.53%	2/10	272
6	0.125	0.90	0.92	0.13	0.1296	0.1555	17.27%	2/10	258

Continuação

N	SIGMA	ETA	ALFA	TOL	DQ	MDA	MXER	NME	NITER
7	0.125	0.90	0.93	0.13	0.1299	0.2081	23.12%	2/10	214
8	0.125	0.90	0.94	0.13	0.1298	0.2253	25.03%	2/10	175
9	0.125	0.90	0.98	0.13	0.1297	0.2396	27.63%	2/10	92
10	0.125	0.90	0.99	0.13	0.1299	0.1000	11.11%	2/10	988
11	0.125	0.92	0.98	0.13	0.1285	0.2379	27.44%	2/10	92
12	0.125	0.94	0.98	0.13	0.1293	0.2384	27.49%	2/10	91
13	0.125	0.96	0.98	0.13	0.1285	0.2368	27.32%	2/10	91
14	0.0125	0.96	0.98	0.13	0.0879	0.0794	8.82%	5/25	03
15	0.00125	0.96	0.98	0.13	0.0879	0.0794	8.82%	5/25	03
16	0.0125	0.96	0.98	0.01	0.0095	0.0774	8.64%	1/4	86
17	0.0125	0.96	0.98	0.001	0.0004	0.0368	4.09%	1/4	97
18	0.0125	0.96	0.98	1E-05	7.8E-06	0.0038	0.42%	1/1	103
19	0.0125	0.96	0.98	1E-07	1.2E-07	4.4E-04	0.05%	1/1	112
20	0.0125	0.96	0.98	1E-08	1.2E-07	4.4E-04	0.05%	1/1	112

Tabela 7.12: Dados dos 5(cinco) primeiros minutos após a falha ocorrer. A ordem de ajuste dos parâmetros é **ALFA-ETA-SIGMA (AES).**

7.8.1.2: 05 MINUTOS/AJUSTE EAS.

N	SIGMA	ETA	ALFA	TOL	DQ	MDA	MXER	NME	NITER
1	0.125	0.90	0.80	0.13	0.1299	0.1227	13.63%	2/7	476
2	0.125	0.92	0.80	0.13	0.1299	0.1224	13.61%	2/7	468
3	0.125	0.94	0.80	0.13	0.1299	0.1220	13.56%	2/7	459
4	0.125	0.96	0.80	0.13	0.1299	0.1217	13.52%	2/7	451
5	0.125	0.98	0.80	0.13	0.1299	0.1214	13.49%	2/7	443
6	0.125	0.99	0.80	0.13	0.1299	0.1211	13.46%	2/7	440
7	0.125	0.99	0.82	0.13	0.1299	0.1211	13.27%	2/10	405

Continuação

N	SIGMA	ETA	ALFA	TOL	DQ	MDA	MXER	NME	NITER
8	0.125	0.99	0.84	0.13	0.1298	0.1222	13.58%	2/10	372
9	0.125	0.99	0.86	0.13	0.1298	0.1269	14.11%	2/10	339
10	0.125	0.99	0.88	0.13	0.1299	0.1333	14.82%	2/10	307
11	0.125	0.99	0.90	0.13	0.1297	0.1491	15.26%	2/10	270
12	0.125	0.99	0.92	0.13	0.1296	0.1620	18.20%	2/10	243
13	0.125	0.99	0.94	0.13	0.1289	0.2284	25.38%	2/10	163
14	0.125	0.99	0.96	0.13	0.1289	0.2386	27.52%	2/10	124
15	0.125	0.99	0.98	0.13	0.1289	0.2348	27.09%	2/10	91
16	0.125	0.99	0.99	0.13	---	---	---	---	NC[50]
17	0.0125	0.99	0.98	0.13	0.0970	0.0813	9.04%	5/25	03
18	0.00125	0.99	0.98	0.13	0.0970	0.0813	9.04%	5/25	03
19	0.0125	0.99	0.98	0.01	0.0085	0.0749	8.33%	1/4	92
20	0.0125	0.99	0.98	1E-03	8.9E-04	0.0384	4.27%	1/4	101
21	0.0125	0.99	0.98	1E-05	8.9E-06	0.0039	0.43%	1/1	107
22	0.0125	0.99	0.98	1E-06	1.5E-07	0.0004	0.05%	1/1	116
23	0.0125	0.99	0.98	1E-09	1.5E-07	0.0004	0.05%	1/1	116

Tabela 7.13: Dados dos 05(cinco) primeiros minutos após a falha ocorrer. A ordem de ajuste dos parâmetros é **ETA-ALFA-SIGMA (EAS).**

7.8.1.3: 05 MINUTOS/AJUSTE SEA.

No estudo deste CASO, considerou-se que a ordem de ajuste ETA-ALFA não teve grande diferença da ordem ALFA-ETA, sendo que desta forma o primeiro foi, arbitrariamente, escolhido.

[50] NC significa que não se alcançou um treinamento com um número razoável de apresentações ou que o valor de MDA atingiu o estado estacionário num valor muito alto para ser considerado viável.

N	SIGMA	ETA	ALFA	TOL	DQ	MDA	MXER	NME	NITER
1	0.125	0.90	0.80	0.13	0.1299	0.1227	13.63%	2/7	476
2	0.0125	0.90	0.80	0.13	0.1175	0.0826	9.18%	3/15	13
3	0.00125	0.90	0.80	0.13	0.1175	0.0826	9.18%	3/15	13
4	0.0125	0.92	0.80	0.13	0.1114	0.0805	8.95%	3/15	13
5	0.0125	0.89	0.80	0.13	0.1207	0.0837	9.31%	3/15	13
6	0.0125	0.92	0.82	0.13	0.1282	0.0857	9.52%	3/15	11
7	0.0125	0.92	0.84	0.13	0.1208	0.0828	9.20%	3/15	10
8	0.0125	0.92	0.86	0.13	0.1116	0.0791	8.79%	3/15	09
9	0.0125	0.92	0.88	0.13	0.0994	0.0742	8.24%	3/15	08
10	0.0125	0.92	0.90	0.13	0.1198	0.0799	8.88%	3/15	06
11	0.0125	0.92	0.92	0.13	0.0899	0.0697	7.66%	1/5	05
12	0.0125	0.92	0.94	0.13	0.0479	0.0465	5.17%	4/15	04
13	0.0125	0.92	0.96	0.13	0.0417	0.0594	7.60%	5/20	03
14	0.0125	0.92	0.98	0.13	0.0766	0.0765	8.50%	5/25	03
15	0.0125	0.92	0.97	0.13	0.0354	0.0544	7.05%	1/6	03
16	0.0125	0.92	0.97	0.01	0.0073	0.0398	4.42%	1/1	23
17	0.0125	0.92	0.97	0.001	7.6E-04	0.0182	2.02%	1/1	28
18	0.0125	0.92	0.97	1E-05	8.2E-06	0.0027	0.31%	1/1	34
19	0.0125	0.92	0.97	1E-07	8.2E-08	2.9E-04	0.03%	1/1	38
20	0.0125	0.92	0.97	1E-08	8.2E-08	2.9E-04	0.03%	1/1	38

Tabela 7.14: Dados dos 05(cinco) primeiros minutos após a falha ocorrer. A ordem de ajuste dos parâmetros é **SIGMA-ETA-ALFA (SEA).**

7.8.1.4: 05 MINUTOS/SEM ESTRATÉGIA DEFINIDA.

N	SIGMA	ETA	ALFA	TOL	DQ	MDA	MXER	NME	NITER
1	0.125	0.99	0.90	0.13	0.1300	0.1474	17.38%	2/10	270
2	0.125	0.99	0.93	0.13	0.1300	0.2243	24.92%	2/10	188

Continuação

N	SIGMA	ETA	ALFA	TOL	DQ	MDA	MXER	NME	NITER
3	0.0125	0.99	0.93	0.13	0.1300	0.0603	7.70%	1/5	04
4	0.00125	0.99	0.93	0.13	0.0751	0.0603	7.70%	1/5	04
5	0.00125	0.99	0.93	0.001	0.0007	0.0603	7.70%	1/5	04
6	0.00125	0.99	0.93	0.0001	0.00008	0.0072	0.80%	3/15	16
7	0.00125	0.99	0.95	0.0001	0.00007	0.00156	0.17%	3/15	53
8	0.00125	0.99	0.98	0.0001	0.00009	0.03384	3.76%	1/6	101
9	0.00125	0.99	0.92	0.0001	0.0001	0.00780	0.87%	3/15	19
10	0.00125	0.99	0.93	1E-07	9.7E-06	7.9E-04	0.09%	3/15	33
11	0.00125	0.99	0.93	1E-08	2.7E-06	4.3E-04	0.48%	3/15	38
12	0.00125	0.99	0.93	1E-09	2.7E-06	4.3E-04	0.48%	3/15	38

Tabela 7.15: Dados dos 05(cinco) primeiros minutos após a falha ocorrer. A ordem de ajuste dos parâmetros não segue nenhuma sistemática pré-definida. NOTAR QUE O AJUSTE CONSEGUIDO AQUI APONTA PARA UM NÚMERO MÍNIMO DE ITERAÇÕES IGUAL AO DA TABELA ANTERIOR, MAS OS PARÂMETROS TÊM VALORES DIFERENTES E A PRECISÃO DE AJUSTE É UM POUCO MENOR.

Para mostrar como a seqüência de ajuste dos parâmetros influencia o número de iterações necessárias para o ajuste dos pesos da rede neural, neste CASO 1[51], montou-se a tabela abaixo, onde se sumariza os resultados apresentados nas três últimas tabelas. Fica evidente que a seqüência **SEA** (**S**igma-**E**ta-**A**lfa) é a mais eficiente uma vez que para alcançar erros (aqui levados em conta a DQ e MDA) semelhantes no ajuste precisaram de um menor número de iterações. Nas seqüências **AES** e **EAS**, não se pode considerar, para este exemplo, que existam diferenças significativas nos números de iterações necessários para uma mesma precisão de ajuste.

[51] Inclui cada um dos itens do CASO 1.

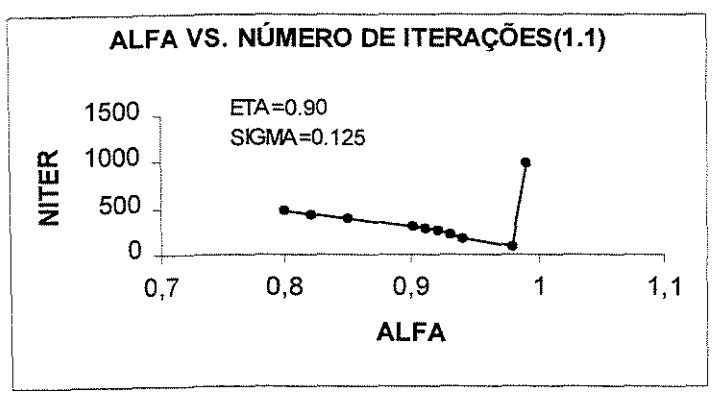
******	SEQÜÊNCIA DE AJUSTE DOS PARÂMETROS		
TOL	AES(1.1)	EAS(1.2)	SEA(1.3)
0.13	03 → 8.82%	03 → 9.04%	03 → 7.05%
1E-02	86 → 8.64%	92 → 8.33%	23 → 4.42%
1E-03	97 → 4.09%	101 → 4.27%	28 → 2.02%
1E-05	103 → 0.42%	101 → 0.43%	34 → 0.31%
1E-07	112 → 0.05%	116 → 0.05%	38 → 0.03%
1E-08	112 → 0.05%	----------	38 → 0.03%
1E-09	----------	116 → 0.05%	----------

Tabela 7.16: Comparativo entre os números de iterações necessários à determinação dos pesos das redes neurais conforme a seqüência utilizada.

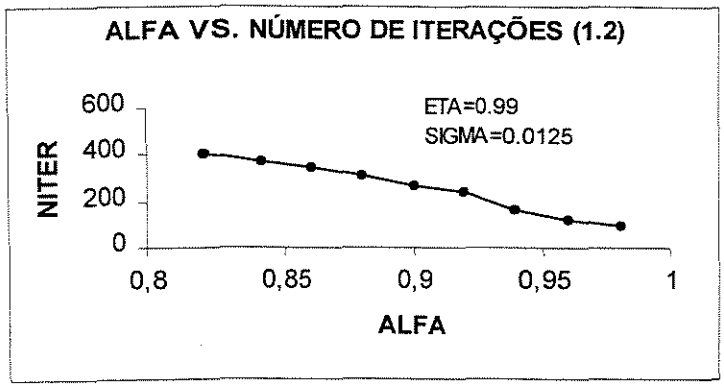
Para uma visão diferente dos parâmetros de treinamento das redes neurais, e suas implicações na precisão do ajuste e no número de iterações, pode-se desenhar estes parâmetros ou a precisão do ajuste como função do número de iterações.

CURVAS DE AJUSTE DOS PARÂMETROS E DA PRECISÃO DE TREINAMENTO VERSUS O NÚMERO DE ITERAÇÕES PARA O CASO 1

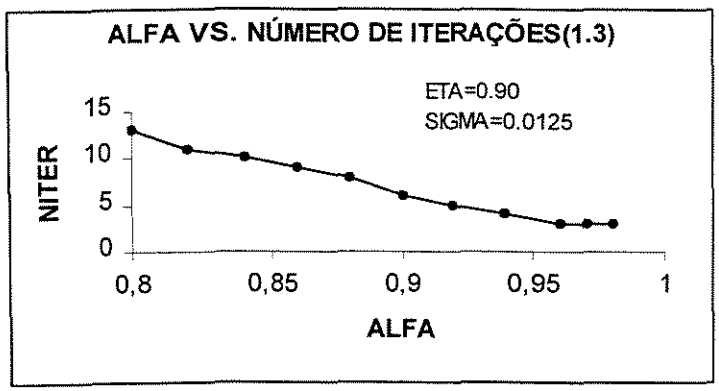
VALORES DE ALFA VERSUS NÚMERO DE ITERAÇÕES



(a)



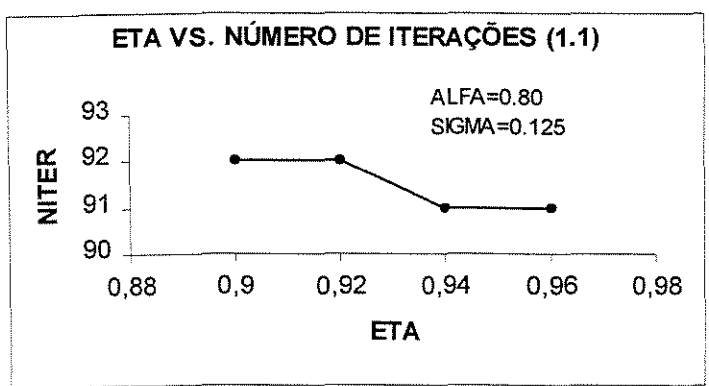
(b)



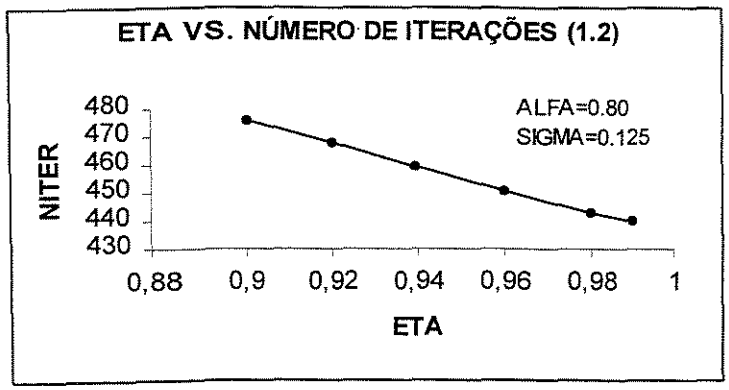
(c)

Figura 7.2: Comportamento dos valores de ALFA, para o CASO 1, quando o mesmo é o primeiro(a), segundo(b) ou terceiro(c) parâmetro a ser ajustado. Notar que seus valores finais não são necessariamente iguais para todas as seqüências dentro do mesmo CASO.

VALORES DE ETA VERSUS NÚMERO DE ITERAÇÕES



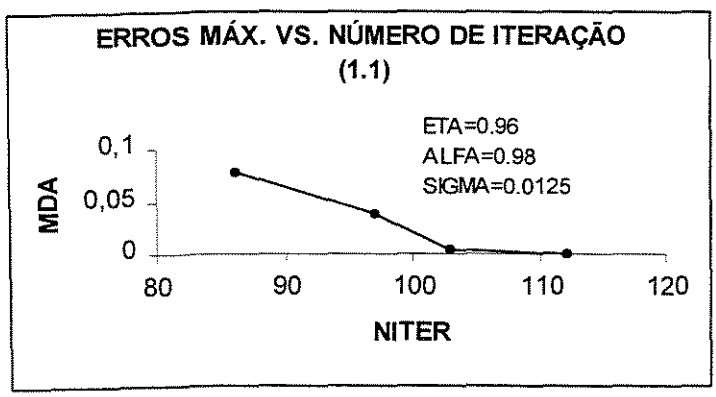
(a)



(b)

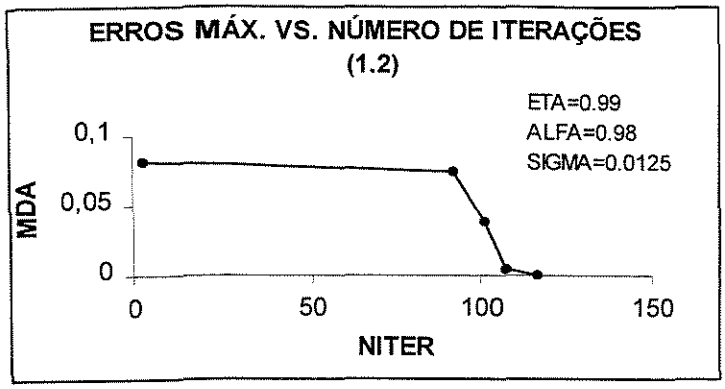
Figura 7.3: Comportamento dos valores de ETA quando o mesmo é o primeiro(b) ou segundo(a) parâmetro a ser ajustado. Notar que seus valores finais não são necessariamente iguais para ambas as seqüências dentro do CASO.

ERROS ABSOLUTOS MÁXIMOS VERSUS NÚMERO DE ITERAÇÕES



(a)

Figura 7.4a: Comparação do número de iterações necessárias para se atingir determinados patamares de precisão, no ajuste dos pesos das redes neurais, quando se tem os valores dos parâmetros já ajustados na forma final.



(b)

Figura 7.4b: Comparação do número de iterações necessárias para se atingir determinados patamares de precisão, no ajuste dos pesos das redes neurais, quando se tem os valores dos parâmetros já ajustados na forma final.

Análise do CASO 1

Quando se tomou os valores dos parâmetros monitorados que não foram utilizados no treinamento da rede neural para os primeiros 5 minutos após o aparecimento da falha, foi possível identificar a situação de falha e associá-la corretamente ao tipo da mesma para ambos os valores. Conforme havia sido dito anteriormente, após pouco tempo do aparecimento da falha, os valores das variáveis monitoradas não apresentaram modificações numéricas suficientes para impedir uma perfeita análise das condições do sistema de processos.

7.8.2 CASO 2

7.8.2.1: 10 MINUTOS/AJUSTE AES.

N	SIGMA	ETA	ALFA	TOL	DQ	MDA	MXER	NME	NITER
1	0.125	0.90	0.75	0.20	0.1998	0.3606	40.06%	2/19	1439
2	0.125	0.90	0.78	0.20	0.1997	0.3661	40.67%	2/19	1384

Continuação

N	SIGMA	ETA	ALFA	TOL	DQ	MDA	MXER	NME	NITER
3	0.125	0.90	0.79	0.20	0.1993	0.3655	40.61%	2/19	1335
4	0.125	0.90	0.80	0.20	0.1990	0.3664	40.71%	2/19	1294
5	0.125	0.90	0.81	0.20	0.1991	0.3819	42.43%	2/19	1551
6	0.125	0.90	0.82	0.20	0.1981	0.3892	43.24%	2/19	1576
7	0.125	0.90	0.85	0.20	0.1989	0.4062	45.13%	2/19	1753
8	0.125	0.95	0.80	0.20	0.1989	0.3820	42.44%	2/19	1554
9	0.125	0.93	0.80	0.20	0.1995	0.3812	42.35%	2/19	1541
10	0.125	0.92	0.80	0.20	0.1994	0.3802	42.24%	2/19	1532
11	0.125	0.85	0.80	0.20	0.1990	0.3672	40.80%	2/19	1356
12	0.125	0.87	0.80	0.20	0.1998	0.3670	40.78%	2/19	1325
13	0.125	0.89	0.80	0.20	0.1992	0.3662	40.69%	2/19	1297
14	0.0125	0.90	0.80	0.20	0.1976	0.0925	10.28%	1/10	12
15	0.00125	0.90	0.80	0.20	0.1976	0.0925	10.28%	1/10	12
16	0.0125	0.90	0.80	0.01	0.0093	0.0214	2.38%	2/20	35
17	0.0125	0.90	0.80	0.001	9.6E-04	0.0071	0.78%	2/20	60
18	0.0125	0.90	0.80	1E-05	9.7E-06	7.7E-04	0.08%	2/20	119
19	0.0125	0.90	0.80	1E-08	3.1E-06	4.4E-04	0.05%	2/20	134
20	0.0125	0.90	0.80	1E-09	3.1E-06	4.4E-04	0.05%	2/20	134

Tabela 7.17: Dados dos 10(dez) primeiros minutos após ocorrer a falha. A ordem de ajuste dos parâmetros é **ALFA-ETA-SIGMA (AES).**

7.8.2.2: 10 MINUTOS/AJUSTE EAS.

N	SIGMA	ETA	ALFA	TOL	DQ	MDA	MXER	NME	NITER
1	0.125	0.90	0.80	0.20	0.1990	0.3663	40.71%	2/19	1294
2	0.125	0.92	0.80	0.20	0.1994	0.3802	42.24%	2/19	1532
3	0.125	0.88	0.80	0.20	0.1998	0.3669	40.77%	2/19	1305
4	0.125	0.89	0.80	0.20	0.1993	0.3662	40.69%	2/19	1297

Continuação

N	SIGMA	ETA	ALFA	TOL	DQ	MDA	MXER	NME	NITER
5	0.125	0.90	0.82	0.20	0.1986	0.3874	43.04%	2/19	1562
6	0.125	0.90	0.78	0.20	0.1998	0.3667	40.75%	2/19	1415
7	0.125	0.90	0.79	0.20	0.1999	0.3668	40.75%	2/19	1364
8	0.0125	0.90	0.80	0.20	0.1940	0.0969	10.19%	1/10	13
9	0.00125	0.90	0.80	0.20	0.1940	0.0916	10.19%	1/10	13
10	0.0125	0.90	0.80	0.01	0.0099	0.0219	2.43%	2/20	37
11	0.0125	0.90	0.80	0.001	0.0009	0.0069	0.78%	2/20	65
12	0.0125	0.90	0.80	1E-04	9.3E-05	0.0023	0.26%	2/20	96
13	0.0125	0.90	0.80	1E-05	9.6E-06	0.0008	0.09%	2/20	128
14	0.0125	0.90	0.80	1E-07	9.6E-06	0.0008	0.09%	2/20	128

Tabela 7.18: Dados dos 10(dez) primeiros minutos após ocorrer a falha. A ordem de ajuste dos parâmetros é **ETA-ALFA-SIGMA(EAS).**

7.8.2.3: 10 MINUTOS/AJUSTE SAE

N	SIGMA	ETA	ALFA	TOL	DQ	MDA	MXER	NME	NITER
1	0.125	0.90	0.80	0.20	0.1990	0.3663	40.71%	2/19	1294
2	0.0125	0.90	0.80	0.20	0.1976	0.0925	10.27%	1/10	12
3	0.00125	0.90	0.80	0.20	0.1976	0.0925	10.27%	1/10	12
4	0.0125	0.90	0.82	0.20	0.1883	0.0904	10.04%	1/10	11
5	0.0125	0.90	0.84	0.20	0.1773	0.0878	9.76%	1/10	10
6	0.0125	0.90	0.86	0.20	0.1638	0.0845	9.39%	1/10	09
7	0.0125	0.90	0.88	0.20	0.1470	0.0803	8.92%	1/10	08
8	0.0125	0.90	0.90	0.20	0.1865	0.0901	10.01%	1/10	06
9	0.0125	0.90	0.92	0.20	0.1573	0.0837	9.30%	1/10	05
10	0.0125	0.90	0.94	0.20	0.1098	0.0716	7.96%	1/10	04
11	0.0125	0.90	0.96	0.20	0.0855	0.0628	7.98%	1/10	03

Continuação

N	SIGMA	ETA	ALFA	TOL	DQ	MDA	MXER	NME	NITER
12	0.0125	0.90	0.98	0.20	0.1944	0.0978	10.87%	3/21	18
13	0.0125	0.90	0.97	0.20	0.1596	0.0789	8.78%	3/41	03
14	0.0125	0.92	0.96	0.20	0.0851	0.0641	7.12%	3/41	03
15	0.0125	0.88	0.96	0.20	0.0869	0.0614	7.83%	3/41	03
16	0.0125	0.88	0.96	0.01	0.0097	0.0505	5.62%	3/41	09
17	0.0125	0.88	0.96	1E-04	7.8E-06	0.0079	0.88%	3/41	25
18	0.0125	0.88	0.96	1E-05	7.8E-06	0.0026	0.29%	3/41	31
19	0.0125	0.88	0.96	1E-07	2.0E-07	4.1E-04	0.05%	3/41	41
20	0.0125	0.88	0.96	1E-08	2.0E-07	4.1E-04	0.05%	3/41	41

Tabela 7.19: Dados dos 10(dez) primeiros minutos após ocorrer a falha. A ordem de ajuste dos parâmetros é **SIGMA-ALFA-ETA(SAE).**

******	SEQÜÊNCIA DE AJUSTE DOS PARÂMETROS		
TOL	**AES(2.1)**	**EAS(2.2)**	**SAE(2.3)**
0.20	12 → 10.27%	13 → 10.19%	03 → 7.83%
1E-02	35 → 2.37%	37 → 2.43%	09 → 5.62%
1E-03	60 → 0.78%	65 → 0.78%	----------
1E-04	----------	96 → 0.26%	25 → 0.88%
1E-05	119 → 0.08%	128 → 0.09%	31 → 0.31%
1E-07	----------	128 → 0.09%	41 → 0.03%
1E-08	134 → 0.05%	----------	41 → 0.03%
1E-09	134 → 0.05%	----------	----------

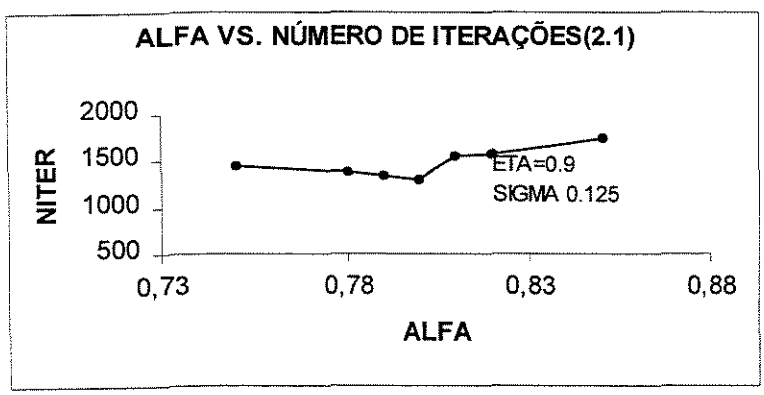
Tabela 7.20: Comparativo entre os números de iterações necessários à determinação dos pesos das redes neurais conforme a seqüência utilizada.

O sumário dos números de iteração para determinados valores de tolerância para o CASO 2 podem ser acompanhados na Tabela 7.20, acima. No CASO 2 comprovou-se que o argumento utilizado para o CASO 1 também se verificou. A seqüência **SEA** é mais

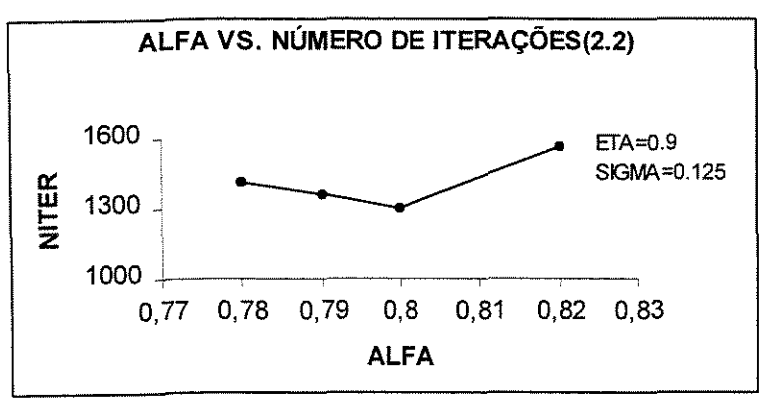
eficiente que as outras duas, que por sua vez não diferem muito entre si em desempenho para o treinamento das redes neurais.

CURVAS DE AJUSTE DOS PARÂMETROS E DA PRECISÃO DE TREINAMENTO VERSUS O NÚMERO DE ITERAÇÕES PARA O CASO 2

VALORES DE ALFA VERSUS NÚMERO DE ITERAÇÕES

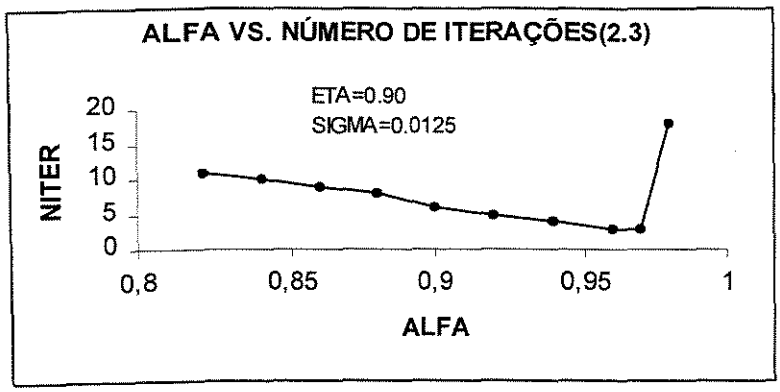


(a)



(b)

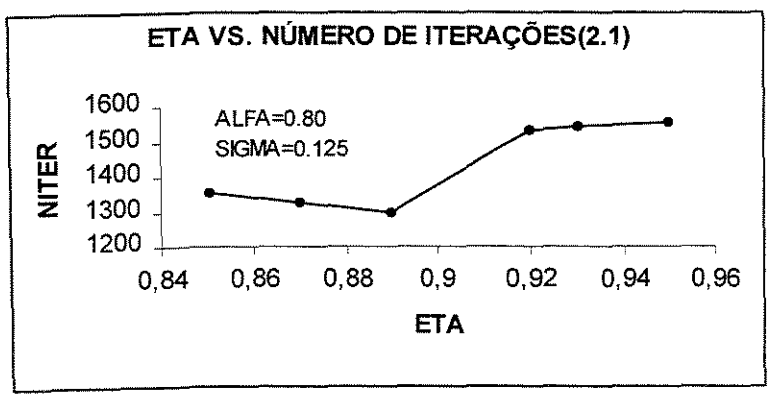
Figura 7.5 : Comportamento dos valores de ALFA, para o CASO 2, quando o mesmo é o primeiro(a) ou segundo(b) a ser ajustado. Notar que seus valores finais não são necessariamente iguais para todas as seqüências dentro do mesmo CASO.



(c)

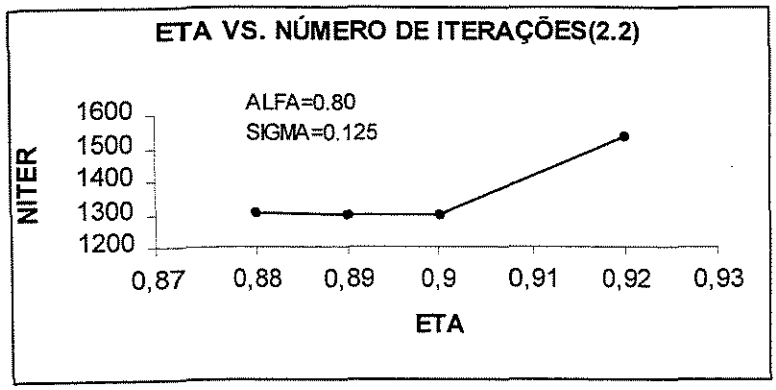
Figura 7.5 : Comportamento dos valores de ALFA, para o CASO 2, quando o mesmo é o terceiro(c) parâmetro a ser ajustado. Notar que seus valores finais não são necessariamente iguais para todas as seqüências dentro do mesmo CASO.

VALORES DE ETA VERSUS NÚMERO DE ITERAÇÕES



(a)

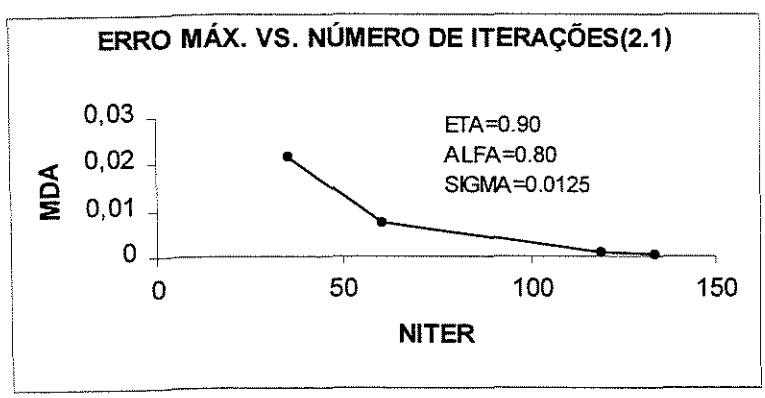
Figura 7.6a: Comportamento dos valores de ETA quando o mesmo é o segundo parâmetro a ser ajustado. Notar que seus valores finais não são necessariamente iguais para ambas as seqüências dentro do CASO.



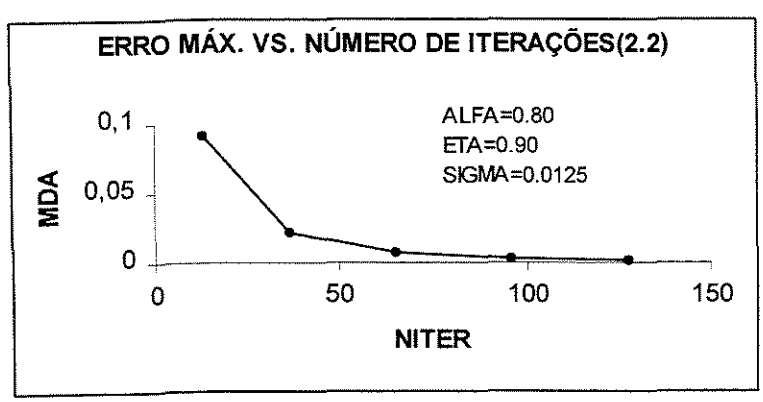
(b)

Figura 7.6: Comportamento dos valores de ETA quando o mesmo é o primeiro(b) parâmetro a ser ajustado. Notar que seus valores finais não são necessariamente iguais para ambas as seqüências dentro do CASO.

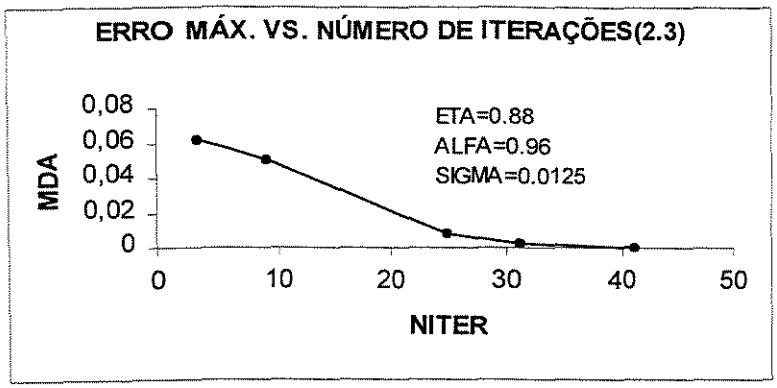
ERROS ABSOLUTOS MÁXIMOS VERSUS NÚMERO DE ITERAÇÕES



(a)



(b)



(c)

Figura 7.7: Comparação do número de iterações necessárias para se atingir determinados patamares de precisão, no ajuste dos pesos das redes neurais, quando se tem os valores dosparâmetros já ajustados na forma final.

Análise do CASO 2

Em conformidade com o CASO 1, mesmo utilizando valores dos parâmetros monitorados do conjunto de padrões referente aos níveis de falhas não utilizados no treinamento da rede, foi possível, para os dois níveis 'extras' detectar e diagnosticar eficientemente as falhas, quando se tomou amostras a 10 minutos do início da falha. Isto mostrou que, para **o sistema utilizado** e para os níveis de falhas propostos, as modificações numéricas nos valores dos parâmetros monitorados também não afetaram a validade do procedimento proposto.

7.8.3 CASO 3

7.8.3.1: 20 MINUTOS/AJUSTE AES.

N	SIGMA	ETA	ALFA	TOL	DQ	MDA	MXER	NME	NITER
1	0.125	0.90	0.80	0.20	0.1999	0.1000	11.11%	5/77	84825
2	0.0125	0.90	0.80	0.20	0.3170	0.1000	11.11%	2/83	7671

Continuação

N	SIGMA	ETA	ALFA	TOL	DQ	MDA	MXER	NME	NITER
3	0.0125	0.90	0.85	0.20	0.3225	0.1000	11.11%	5/77	5705
4	0.0125	0.90	0.88	0.20	0.3555	0.1000	11.11%	5/77	3320
5	0.0125	0.90	0.89	0.20	0.3450	0.1000	11.11%	5/77	3269
6	0.0125	0.90	0.90	0.20	0.2954	0.1000	11.11%	5/77	5886
7	0.0125	0.89	0.89	0.20	0.3361	0.1000	11.11%	5/77	3291
8	0.0125	0.88	0.89	0.20	0.3450	0.1000	11.11%	5/77	3251
9	0.0125	0.87	0.89	0.20	0.3500	0.1000	11.11%	5/77	3187
10	0.0125	0.86	0.89	0.20	0.3499	0.1000	11.11%	5/77	3176
11	0.0125	0.85	0.89	0.20	0.3592	0.1000	11.11%	5/77	3126
12	0.0125	0.84	0.89	0.20	0.3681	0.1000	11.11%	5/77	3114
13	0.0125	0.83	0.89	0.20	0.3594	0.1000	11.11%	5/77	3106
14	0.0125	0.82	0.89	0.20	---	---	---	---	NC
15	0.0125	0.80	0.89	0.20	0.3395	0.1000	11.11%	5/77	3363
16	1.25E-3	0.83	0.89	0.20	0.1997	0.4430	49.22%	2/23	257
17	1.25E-4	0.83	0.89	0.20	0.1668	0.0595	7.61%	3/52	12
18	1.25E-5	0.83	0.89	0.20	0.1668	0.0595	7.61%	3/52	12
19	1.25E-4	0.83	0.89	0.01	0.0098	0.0375	4.16%	5/89	38
20	1.25E-4	0.83	0.89	0.001	9.9E-04	0.0136	1.51%	5/89	331
21	1.25E-4	0.83	0.89	5E-04	4.9E-04	0.0096	1.06%	5/89	662
22	1.25E-4	0.83	0.89	1E-04	9.9E-05	0.0043	0.47%	5/89	3294
23	1.25E-4	0.83	0.89	1E-05	9.9E-06	0.0013	0.15%	5/89	33933

Tabela 7.21: Dados dos 20(vinte) primeiros minutos após ocorrer a falha. A ordem de ajuste dos parâmetros é **ALFA-ETA-SIGMA(AES).**

7.8.3.2: 20 MINUTOS/AJUSTE SAE.

N	SIGMA	ETA	ALFA	TOL	DQ	MDA	MXER	NME	NITER
1	0.0125	0.90	0.80	0.20	0.1999	0.1000	11.11%	5/77	84825

Continuação

N	SIGMA	ETA	ALFA	TOL	DQ	MDA	MXER	NME	NITER
2	0.00125	0.90	0.80	0.20	0.1998	0.4420	49.11%	2/23	241
3	0.00125	0.90	0.80	0.20	0.1998	0.0660	7.33%	3/52	19
4	1.25E-4	0.90	0.80	0.20	0.1998	0.0660	7.33%	3/52	19
5	1.25E-5	0.90	0.82	0.20	0.1994	0.0663	7.37%	3/52	17
6	1.25E-4	0.90	0.84	0.20	0.1717	0.0609	7.77%	3/52	16
7	1.25E-4	0.90	0.86	0.20	0.1705	0.0606	7.73%	3/52	14
8	1.25E-4	0.90	0.88	0.20	0.1691	0.0599	7.67%	3/52	12
9	1.25E-4	0.90	0.90	0.20	0.1671	0.0591	7.57%	3/52	10
10	1.25E-4	0.90	0.92	0.20	0.1650	0.0583	7.50%	5/89	08
11	1.25E-4	0.90	0.94	0.20	0.1663	0.0581	7.46%	5/89	06
12	1.25E-4	0.90	0.96	0.20	0.1900	0.0821	9.13%	2/34	05
13	1.25E-4	0.90	0.98	0.20	0.1975	0.0988	10.99%	2/34	18
14	1.25E-4	0.90	0.97	0.20	0.1895	0.0936	10.40%	2/34	05
15	1.25E-4	0.92	0.96	0.20	0.1852	0.0833	9.26%	2/34	04
16	1.25E-4	0.94	0.96	0.20	0.1812	0.0845	9.38%	2/34	04
17	1.25E-4	0.92	0.96	0.01	0.0099	0.0501	5.57%	2/34	28
18	1.25E-4	0.92	0.96	1E-03	9.9E-04	0.0136	1.51%	5/89	108
19	1.25E-4	0.92	0.96	1E-04	9.9E-05	0.0043	0.47%	5/89	1092
20	1.25E-4	0.92	0.96	1E-05	9.9E-06	0.0013	0.15%	5/89	11157

Tabela 7.22: Dados dos 20(vinte) primeiros minutos após ocorrer a falha. A ordem de ajuste dos parâmetros é **SIGMA-ALFA-ETA(SAE).**

O sumário dos números de iteração para determinados valores de tolerância para o CASO 3 podem ser acompanhados na tabela 7.23. No CASO 3, devido ao fato de já se conhecer que as seqüências **EAS** e **AES** terem pouca diferença, entre si quanto ao número de iterações necessárias ao treinamento da rede, a uma determinada precisão, não se ajustou os valores dos parâmetros para o caso da seqüência **EAS**. Comprovou-se mais uma vez o

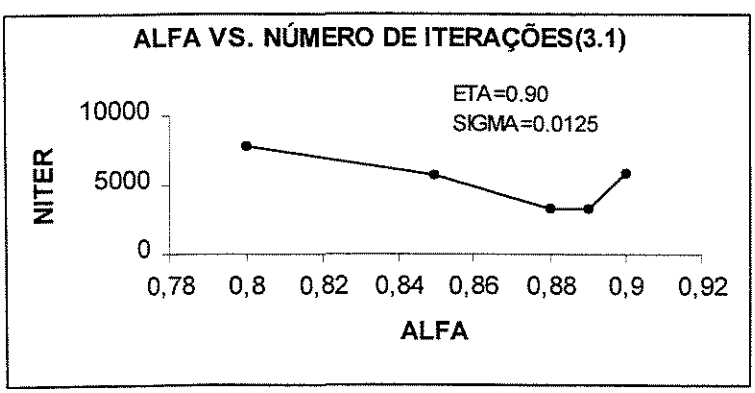
que já tinha sido explicitado nos CASOS 1 e 2. A seqüência **SEA** é mais eficiente que as outras duas.

******	SEQÜÊNCIA DE AJUSTE DOS PARÂMETROS		
TOL	**AES(3.1)**	**EAS(*)**	**SEA(3.2)**
0.20	12 → 7.61%	**********	04 → 9.38%
1E-02	38 → 4.16%	**********	28 → 5.57%
1E-03	331 → 1.51%	**********	108 → 1.51%
5E-04	662 → 1.06%	**********	----------
1E-04	3294 → 0.47%	**********	1092 → 0.47%
1E-05	33933 → 0.15%	**********	11157 → 0.15%

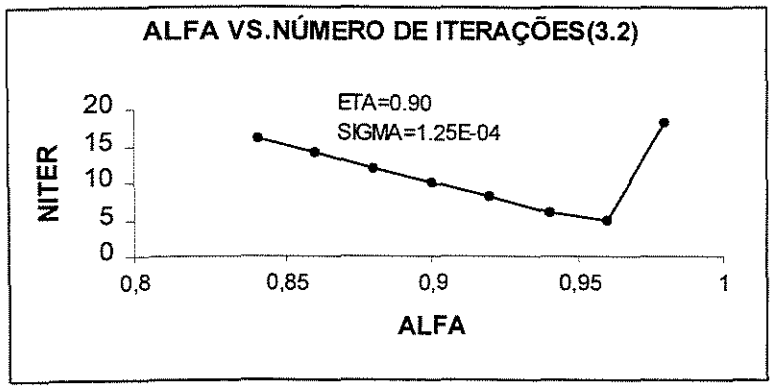
Tabela 7.23: Comparativo entre os números de iterações necessários à determinação dos pesos das redes neurais conforme a seqüência utilizada. Entre parênteses, o número do CASO.

CURVAS DE AJUSTE DOS PARÂMETROS E DA PRECISÃO DE TREINAMENTO VERSUS O NÚMERO DE ITERAÇÕES PARA O CASO 3

VALORES DE ALFA VERSUS NÚMERO DE ITERAÇÕES



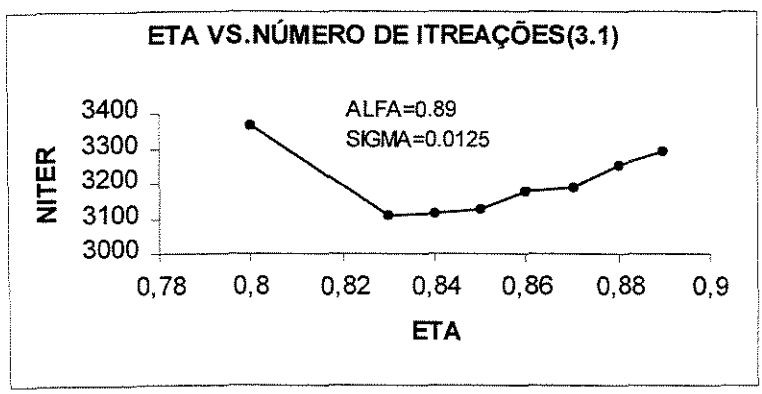
(a)



(b)

Figura 7.8: Comportamento dos valores de ALFA, para o CASO 3, quando o mesmo é o primeiro(a) ou o segundo(b) parâmetro a ser ajustado. Notar que seus valores finais não são necessariamente iguais para todas as seqüências dentro do mesmo CASO.

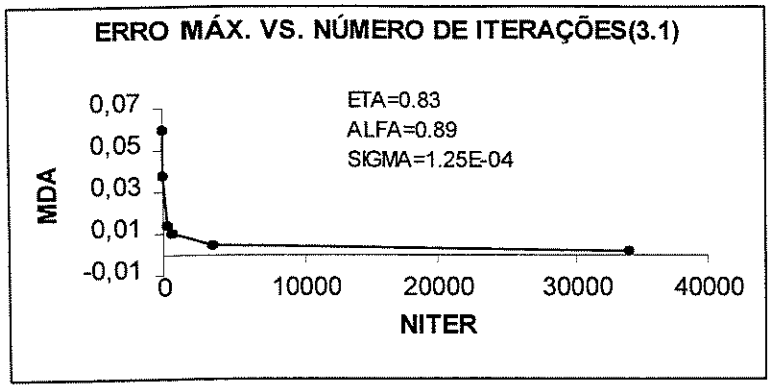
VALORES DE ETA VERSUS NÚMERO DE ITERAÇÕES



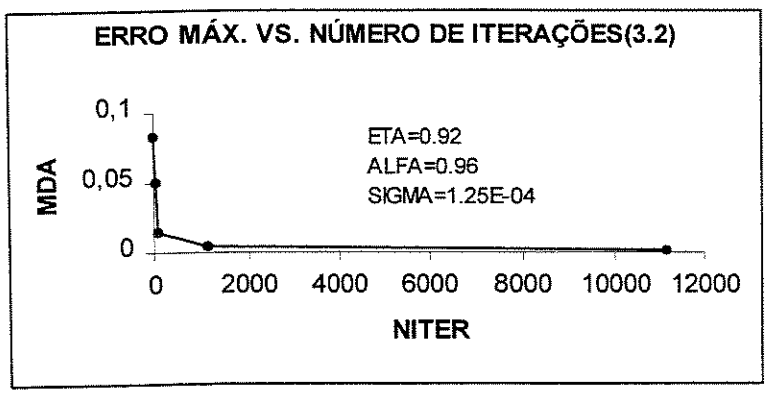
(a)

Figura 7.9: Comportamento dos valores de ETA quando o mesmo é o segundo parâmetro a ser ajustado. Não foi desenhada a curva para o CASO 3.2 por falta de dados. O valor final do parâmetro foi encontrado após apenas três tentativas.

ERROS ABSOLUTOS MÁXIMOS VERSUS NÚMERO DE ITERAÇÕES



(a)



(b)

Figura 7.10: Comparação do número de iterações necessárias para se atingir determinados patamares de precisão, no ajuste dos pesos das redes neurais, quando se tem os valores dos parâmetros já ajustados na forma final.

Análise de CASO 3

A tomada de valores das variáveis após 20 minutos de decorrida a falha não se mostrou tão eficiente quanto nos casos anteriores. Aqui, o maior problema aconteceu com as falhas F1 e F5.

Uma possível explicação está na grande amplitude de variação dos níveis utilizados para estes tipos de falhas. Isto fez com que os comportamentos dinâmicos das variáveis monitoradas apresentassem desvios muito grandes já com 20 minutos de funcionamento em condição de falha.

Uma forma de tentar solucionar o problema é diminuindo o tempo de amostragem das variáveis monitoradas, o que certamente 'força' o ajuste dos parâmetros da rede para onde os valores estão mais próximos.

Nas condições dadas neste trabalho, 20 minutos não seria a melhor escolha como padrão de análise do sistema em funcionamento.

7.8.4 CASO4

7.8.4.1: 30 MINUTOS/AJUSTE SEA.

Devido ao fato de se ter comprovado, através dos CASOS anteriores, que a seqüência de ajuste **SEA** é a mais eficiente, no estudo deste caso apenas os resultados deste ajuste serão apresentados.

N	SIGMA	ETA	ALFA	TOL	DQ	MDA	MXER	NME	NITER
1	0.01250	0.90	0.80	0.10	0.0993	0.0685	7.61%	4/12	36
2	0.00125	0.90	0.80	0.10	---	---	---	---	NC
3	0.12500	0.90	0.80	0.10	0.0999	0.0885	9.83%	1/8	508
4	0.01250	0.92	0.80	0.10	0.0976	0.0677	7.52%	4/12	36
5	0.01250	0.94	0.80	0.10	0.0984	0.0679	7.55%	4/12	35
6	0.01250	0.96	0.80	0.10	0.0993	0.0682	7.58%	4/12	34
7	0.01250	0.98	0.80	0.10	0.0978	0.0675	7.50%	4/12	34
8	0.01250	0.96	0.82	0.10	0.0996	0.0679	7.55%	4/12	31
9	0.01250	0.96	0.84	0.10	0.0976	0.0664	7.39%	4/12	29
10	0.01250	0.96	0.86	0.10	0.0966	0.0652	7.24%	4/12	27
11	0.01250	0.96	0.88	0.10	0.0968	0.0650	7.23%	4/10	26
12	0.01250	0.96	0.90	0.10	0.0967	0.0795	8.83%	3/33	28

Continuação

N	SIGMA	ETA	ALFA	TOL	DQ	MDA	MXER	NME	NITER
13	0.01250	0.96	0.88	0.035	0.0349	0.0613	7.81%	1/10	7295
14	0.01250	0.96	0.88	0.030	0.0299	0.0613	7.81%	3/10	13291
15	0.01250	0.96	0.88	0.0265	0.02649	0.0613	7.81%	3/10	35975

Tabela 7.24: Dados dos 30(trinta) primeiros minutos após ocorrer a falha. A ordem de ajuste dos parâmetros é **SIGMA-ETA-ALFA(SEA).**

Notar que para os itens 13/14/15, os valores do erro máximo (MDA) não variam apesar de o valor da diferença quadrática estar diminuindo conforme se fazem mais apresentações. Isto indica, conforme citado anteriormente, que o valor de MDA atingiu o seu valor mínimo e aumentando-se o número de iterações não é possível baixá-lo. Na hora do inferenciamento[52] o erro máximo conseguido é o mesmo para qualquer um dos três casos, de forma que o que necessita menos iterações é o mais recomendável.

[52] Isto é válido se se considerar no inferenciamento apenas os valores usados no treinamento da rede neural, com valores não iguais ao utilizado no treinamento, o valor de MDA certamente aumentará.

CURVAS DE AJUSTE DOS PARÂMETROS E DA PRECISÃO DE TREINAMENTO VERSUS O NÚMERO DE ITERAÇÕES PARA O CASO 4

VALORES DE ALFA VERSUS NÚMERO DE ITERAÇÕES

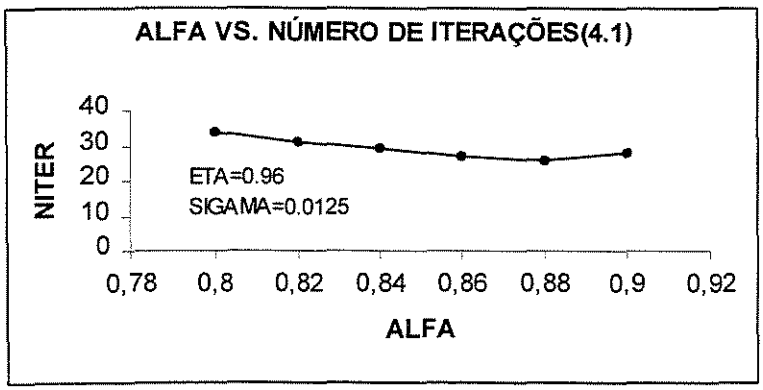


Figura 7.11: Comportamento dos valores de ALFA, para o CASO 4. Neste CASO somente a seqüência SEA foi utilizada para os ajustes dos parâmetros.

VALORES DE ETA VERSUS NÚMERO DE ITERAÇÕES

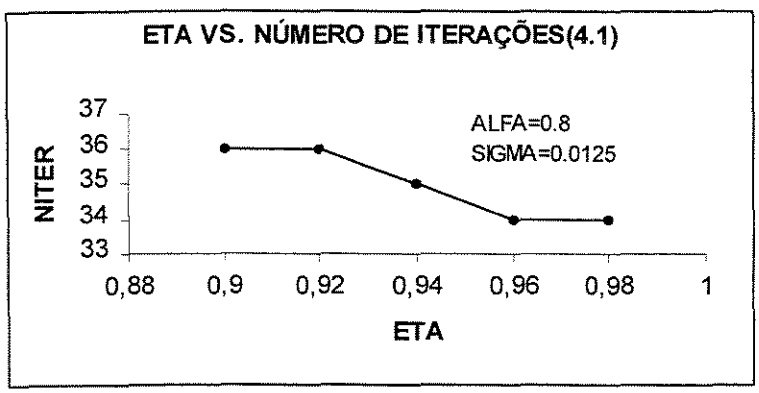


Figura 7.12: Comportamento dos valores de ETA utilizando a seqüência SEA.

Para este caso não foi desenhada a curva **MDA VS NITER** por causa da grande disparidade entre os valores, o que não ficaria bom nas escalas aqui utilizadas.

Análise do CASO 4

Com trinta minutos de funcionamento do sistema com falha, a tendência apresentada com o estudo das variáveis monitoradas se mostrou parecida com a do CASO anterior. Os mesmos dois tipos de falhas, F1 e F5 mais a falha F4, foram os mais problemáticos de serem diagnosticados corretamente. Em ambos os níveis das falhas não utilizados no treinamento da rede, comprova-se que o distanciamento entre os comportamentos do sistema aumenta com o decorrer do tempo.

Novamente aqui, a este tempo de funcionamento do sistema com falhas do tipo F1, F4 e F5, não é possível de forma confiável diagnosticá-la contando com as características de tomadas de padrões para o treinamento das redes da forma que foi feito.

Resumindo-se num quadro qualitativo as análise feitas para os quatro CASOS anteriores, tem-se

TA	AC	1° NÍVEL					2° NÍVEL					ERRO MÁXIMO	
		F1	F2	F3	F4	F5	F1	F2	F3	F4	F5	NI	EP
5	S	S	S	S	S	S	S	S	S	S	S	38	0,03%
10	S	S	S	S	S	S	S	S	S	S	S	41	0,03%
20	S	N	S	S	S	N	N	S	S	S	N	1115	0,15%
30	S	N	S	S	N	N	N	S	S	N	N	7295	7,81%

Quadro 1: Resumo qualitativo do inferenciamento de falhas tal que seu padrões tenham sido usadas ou não no treinamento das redes neurais

TA = tempo de amostragem

AC = inferenciamento com dados usados para o treinamento

S = 100%

N = menos que 100%

NI = número de iterações

EP = erro percentual

De **F1** a **F5** no primeiro nível e nas colunas do segundo nível foram usados padrões para inferenciamento não utilizados no treinamento das redes neurais.

Optou-se por não usar valores numéricos pois para tanto seria conveniente se ter mais níveis de treinamento e mais níveis de conferência ou de validação, ou então fazer a análise para mais tempos intermediários.

A análise dos dados das duas últimas colunas da tabela acima, mostra perfeitamente o distanciamento entre os valores das variáveis monitoradas após certo tempo de aparecimento da falha. A despeito do aumento da quantidade de dados para o treinamento das redes neurais com o passar do tempo, o que em princípio tende a aumentar o número de iterações necessárias para o ajuste, é no erro máximo que se identifica a disparidade entre os padrões a cada nível de falhas à medida que o funcionamento do sistema avança.

Nos quatro CASOS anteriormente analisados, neste EXEMPLO, utilizou-se uma rede 'cheia', ou seja, todos os tipos de falhas mais a situação sem falha eram apresentados à rede para o ajuste de seus pesos, o que possibilitou detectar e diagnosticar com apenas uma rede treinada, qualquer uma das falhas previamente definidas, desde que em tempo conveniente.

A fim de se diminuir o número de neurônios da camada de saída (e com isso diminuir a dificuldade de treinamento da rede), que é onde se encontram os neurônios correspondentes a cada falha ou à não falha, pode-se treinar uma rede que sirva apenas para indicar que existe uma situação de falha ou se o sistema está operando em condição normal de funcionamento. Para tanto basta treinar uma rede com apenas duas saídas onde cada uma representa respectivamente uma das duas situações citadas.

Se por um lado a rede perde uma de suas funções, qual seja, a de diagnosticar a falha ocorrida, por outro permite a utilização de uma quantidade de exemplos bem maior, o

que se traduz em apresentação de mais situações do estado do sistema, ou seja, mais níveis de falhas, fazendo melhorar a precisão de verificação do estado do sistema através do inferenciamento. É também possível aumentar-se o número de amostras diminuindo-se o intervalo de amostragem das mesmas.

Para efeito de comparação direta, veja-se o CASO 4, onde tem-se o maior tempo de amostragem, 30 minutos. Esses 30 minutos multiplicados por cada uma das 5 falhas mais a situação normal, vezes o número de níveis (5), correspondem a 751 amostras. No CASO 5 para um máximo de 200 amostras, (tomadas a cada 0,5 minuto), tem-se 200 vezes 5 falhas, vezes cinco níveis mais a situação normal, o que dá um total de 5001 amostras. Nota-se que no CASO 5 para se atingir uma MDA menor que o menor valor encontrado no CASO 4, foram necessárias apenas 55 iterações contra mais de 7000. Para se atingir uma precisão 13 vezes melhor foram necessárias apenas 4813 iterações.

As Tabelas 7.25, 7.26 e 7.27 mostram os resultados do treinamento de redes utilizando 100, 50 ou 25 minutos (amostrados a cada 0,5 minuto) de dados de funcionamento do sistema em estudo. Isto dá um total de 5001, 2501 e 1251 amostras, respectivamente. A rigor não seria necessário utilizar um tempo tão longo como 100 minutos de funcionamento do sistema. O intuito aqui é aumentar o número de amostras e se isto fosse feito apenas diminuindo o tempo de amostragem, os valores dos parâmetros seriam muito próximos. Para os níveis de falhas forma usados os mesmos dos CASOS de 1 a 4.

7.8.5 CASO 5

7.8.5.1: 100 MINUTOS/AJUSTE SEA/APENAS DETECÇÃO

N	SIGMA	ETA	ALFA	TOL	DQ	MDA	MXER	NME	NITER
1	0.0125	0.90	0.80	0.01	0.0099	0.1156	12.84%	1/2	38
2	0.0125	0.92	0.80	0.01	0.0096	0.1138	12.65%	1/2	38
3	0.0125	0.88	0.80	0.01	0.0098	0.1154	12.82%	1/2	39
4	0.0125	0.90	0.82	0.01	0.0096	0.1137	12.64%	1/2	35
5	0.0125	0.90	0.84	0.01	0.0096	0.1138	12.65%	1/2	31
6	0.0125	0.90	0.86	0.01	0.0097	0.1140	12.67%	1/2	27
7	0.0125	0.90	0.88	0.01	0.0098	0.1141	12.68%	1/2	23
8	0.0125	0.90	0.90	0.01	0.0099	0.1144	12.72%	1/2	19
9	0.0125	0.90	0.92	0.01	0.0099	0.1103	12.26%	1/2	16
10	0.0125	0.90	0.94	0.01	0.0099	0.1123	12.47%	1/2	11
11	0.0125	0.90	0.95	0.01	0.0036	0.0445	4.95%	1/2	45
12	0.0125	0.90	0.96	0.01	0.0097	0.099	11.11%	1/2	1996
13	0.00125	0.90	0.94	0.01	0.0097	0.099	11.11%	1/2	3861
14	0.125	0.90	0.94	0.01	---	---	---	---	NC
15	0.0125	0.90	0.94	0.001	0.0009	0.0466	4.96%	1/2	55
16	0.0125	0.90	0.94	1E-04	9.9E-05	0.0141	1.57%	1/2	492
17	0.0125	0.90	0.94	1E-05	9.9E-06	0.0045	0.50%	1/2	4813

Tabela 7.25: Dados dos 100 primeiros minutos após ocorrer a falha. Rede apenas para detecção de falha sem que seja feito o seu diagnóstico.

7.8.5.2: 50 MINUTOS/AJUSTE SEA/APENAS DETECÇÃO

N	SIGMA	ETA	ALFA	TOL	DQ	MDA	MXER	NME	NITER
1	0.125	0.90	0.80	0.01	0.0099	0.0224	2.50%	1/311	53
2	0.0125	0.90	0.80	0.01	0.0098	0.0276	3.07%	1/2	21
3	0.00125	0.90	0.80	0.01	0.3211	0.7997	88.86%	1/20	229
4	0.0125	0.92	0.80	0.01	0.0096	0.0269	2.99%	1/2	21

Continuação

N	SIGMA	ETA	ALFA	TOL	DQ	MDA	MXER	NME	NITER
5	0.0125	0.88	0.80	0.01	0.0095	0.0265	2.99%	1/2	22
6	0.0125	0.92	0.82	0.01	0.0096	0.0243	2.95%	1/2	19
7	0.0125	0.92	0.84	0.01	0.0096	0.0249	2.90%	1/2	17
8	0.0125	0.92	0.86	0.01	0.0095	0.0254	2.82%	1/2	15
9	0.0125	0.92	0.88	0.01	0.0095	0.0254	2.82%	1/2	13
10	0.0125	0.92	0.90	0.01	0.0095	0.0257	2.80%	1/2	11
11	0.0125	0.92	0.92	0.01	0.0094	0.0264	2.94%	1/116	09
12	0.0125	0.92	0.94	0.01	0.0018	0.0147	1.63%	1/3	22
13	0.0125	0.92	0.92	1.E-03	9.6E-04	0.0109	1.22%	1/1	26
14	0.0125	0.92	0.92	1E-04	1.19E-4	0.0116	1.13%	1/1	135
15	0.0125	0.92	0.92	1E-04	1.19E-4	0.01168	1.13%	1/1	INF

Tabela 7.26: Dados dos 50 primeiros minutos após ocorrer a falha. Rede apenas para detecção de falha sem que seja feito o seu diagnóstico.

7.8.5.3: 25 MINUTOS/AJUSTE SEA/APENAS DETECÇÃO

N	SIGMA	ETA	ALFA	TOL	DQ	MDA	MXER	NME	NITER
1	0.0125	0.90	0.80	0.01	0.0098	0.0249	2.77%	1/57	20
2	0.00125	0.90	0.80	0.01	0.0097	0.0254	2.82%	1/57	21
3	0.125	0.90	0.80	0.01	0.0098	0.0311	3.46%	1/273	25
4	0.0125	0.92	0.80	0.01	0.0097	0.0248	2.76%	1/57	20
5	0.0125	0.88	0.80	0.01	0.0096	0.0247	2.74%	1/57	21
6	0.0125	0.90	0.82	0.01	0.0099	0.0249	2.78%	1/57	18
7	0.0125	0.90	0.84	0.01	0.0095	0.0245	2.72%	1/57	17
8	0.0125	0.90	0.86	0.01	0.0095	0.0245	2.73%	1/57	15
9	0.0125	0.90	0.88	0.01	0.0096	0.0248	2.75%	1/57	13
10	0.0125	0.90	0.90	0.01	0.0096	0.0250	2.78%	1/57	11
11	0.0125	0.90	0.92	0.01	0.0098	0.0259	2.89%	1/57	09

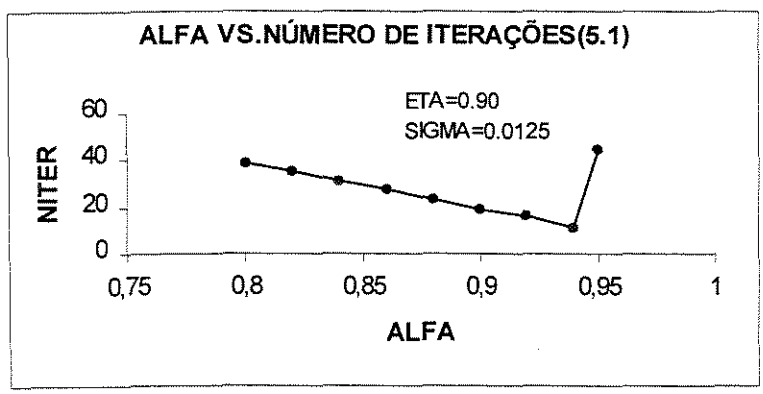
Continuação

N	SIGMA	ETA	ALFA	TOL	DQ	MDA	MXER	NME	NITER
12	0.0125	0.90	0.93	0.01	0.0033	0.0153	1.70%	1/57	17
13	0.0125	0.90	0.94	0.01	0.2432	0.0984	10.96%	1/1	35
14	0.0125	0.90	0.92	0.001	9.5E-04	7.3E-03	0.82%	1/57	27
15	0.0125	0.90	0.92	1E-04	9.3E-05	1.5E-03	0.17%	1/59	48
16	0.0125	0.90	0.92	1E-05	9.4E-06	4.6E-04	0.05%	1/274	69
17	0.0125	0.90	0.92	1E-06	8.4E-06	4.4E-04	0.05%	1/274	70

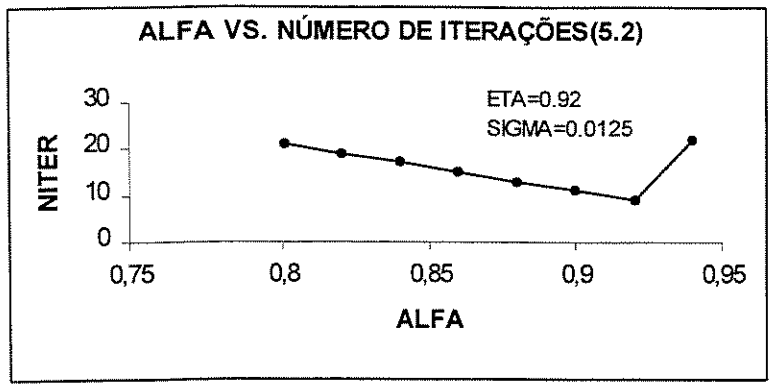
Tabela 7.27: Dados dos 25 primeiros minutos após ocorrer a falha. Rede apenas para detecção de falha sem que seja feito o seu diagnóstico.

CURVAS DE AJUSTE DOS PARÂMETROS E DA PRECISÃO DE TREINAMENTO VERSUS O NÚMERO DE ITERAÇÕES PARA O CASO 5

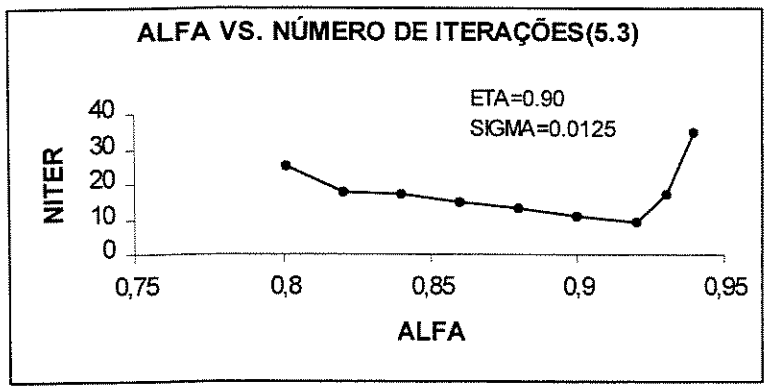
VALORES DE ALFA VERSUS NÚMERO DE ITERAÇÕES



(a)



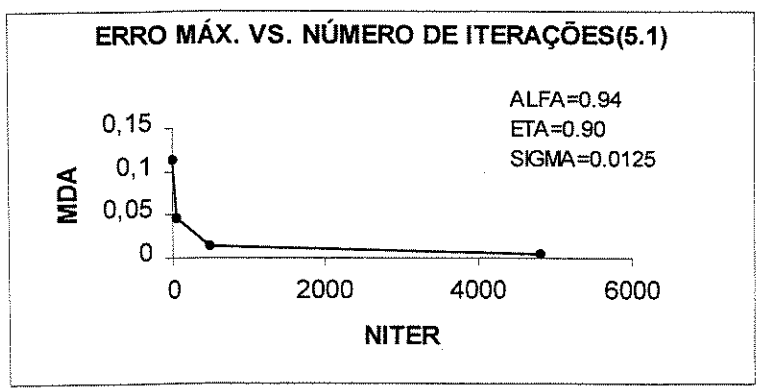
(b)



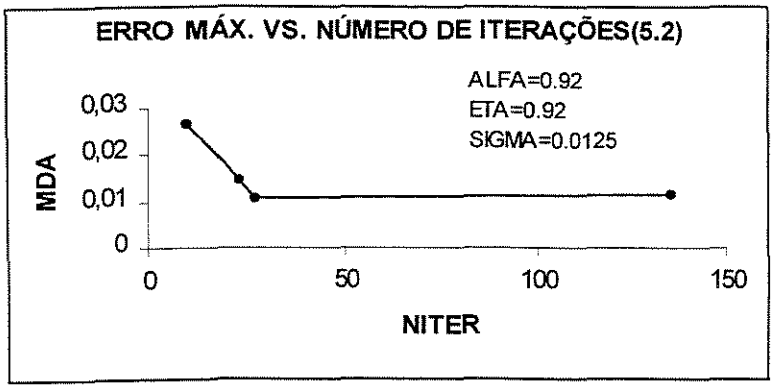
(c)

Figura 7.13: Comportamento dos valores de ALFA, para o CASO 5, onde são utilizados os valores das 6 variáveis para treinamento da rede durante 100, 50 e 25 minutos, respectivamente, o que leva a uma conjunto de 5001, 2501 e 1251 exemplos para o treinamento da rede neural.

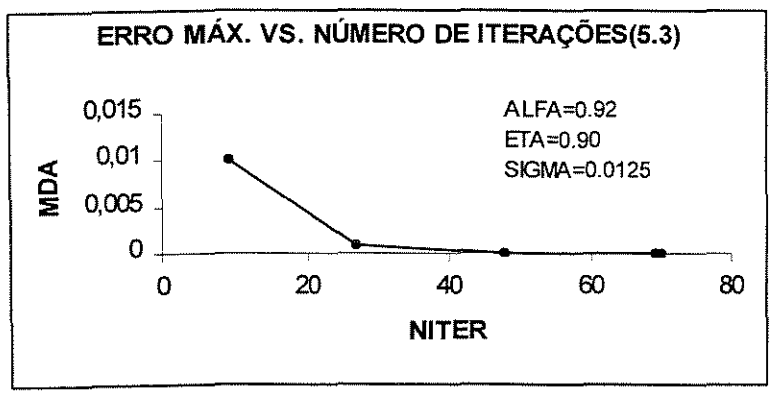
ERROS ABSOLUTOS MÁXIMOS VERSUS NÚMERO DE ITERAÇÕES



(a)



(b)



(c)

Figura 7.14: Comparação do número de iterações necessárias para se atingir determinados patamares de precisão, no ajuste dos pesos das redes neurais, quando se tem os valores dos parâmetros já ajustados na forma final, para as mesmas situações descritas na Figura 7.28.

Não está aqui apresentada a curva de **ETA vs. NITER** por causa da pequena quantidade de dados. Novamente, o ETA final foi rapidamente encontrado.

Análise do CASO 5

Aqui, devido à utilização de apenas duas saídas, as quais representam a situação de falha ou não falha, pode-se usar o mesmo artifício usado no EXEMPLO 1, no item 7.1. Considerando-se qualquer saída até 0,2 como sendo zero e qualquer saída maior que 0,8 como sendo um, todas as falhas puderam ser identificadas, tanto aquelas cujos níveis tiveram valores empregados no treinamento das redes como aqueles cujos valores dos

parâmetros em determinados níveis foram usados apenas para teste de validação. O tempo de inferenciamento foi de 10 minutos em todos os conjuntos de dados.

Devido à alta precisão e ao, relativamente baixo, número de iterações necessárias ao treinamento das redes conseguida com os procedimentos desenvolvidos neste trabalho, principalmente quando se usa poucos neurônios de saída, pode-se utilizar estas vantagens para estender o uso de tais procedimentos aqui mostrados para um sistema unitário, representado por uma coluna de destilação, para sistemas mais complexos, envolvendo uma parcela maior de equipamentos, o que se traduz em mais possibilidades de falhas, de uma planta química ou mesmo a planta inteira.

O procedimento consiste em treinar redes com apenas uma saída atribuindo-se a este neurônio um valor de 0.00 a 1.00, por exemplo. No caso de se usar com a precisão centesimal apresentada, poder-se-ia, em tese, detectar até 100 diferentes tipos de falhas.

Pode-se ao invés de treinar redes com grande número de níveis diferentes de falhas dentro de um mesmo conjunto de dados, separá-los em subredes menores, aumentando-se o número de redes a serem treinadas mas melhorando-se a velocidade de cálculo dos pesos das mesmas. Deste modo, ter-se-ia, da Figura 7.15, F1.1, F1.2, ... , até onde se achasse conveniente.

O que garante que seria possível se fazer este tipo de detecção é o alto grau de precisão do cálculo do valor inferenciado quando se tem um valor idêntico ao valor utilizado para treinamento da rede. Precisão na casa de décimos de milésimo não foram incomuns neste trabalho, conforme pode ser comprovado pelas tabelas aqui apresentadas. Para se contornar o problema da distância entre o valor real e o valor utilizado para treinamento da rede pode-se aumentar o número de amostras o que, se por um lado compromete tempo de treinamento da rede, por outro possibilita um melhor conhecimento, pela rede, de cada estágio pelo qual passa o sistema em análise.

Depois de treinada cada rede com uma determinada situação de falha que se ser detectar mais tarde, pode-se construir um programa computacional bastante simples que a cada intervalo de tempo determinado, conforme discutido anteriormente, faz a verificação de uma determinada amostra do estado do sistema para saber se ela se encontra em situação de falha ou em condições normais de operação. Por ser quase instantâneo o inferenciamento, mesmo se tendo muitas situações a serem conferidas, não se gasta mais que alguns poucos

segundos, ou talvez menos, em se tratando de computadores de alto desempenho. O grande problema, portanto, é treinar as inúmeras redes necessárias. O trabalho é realmente tedioso.

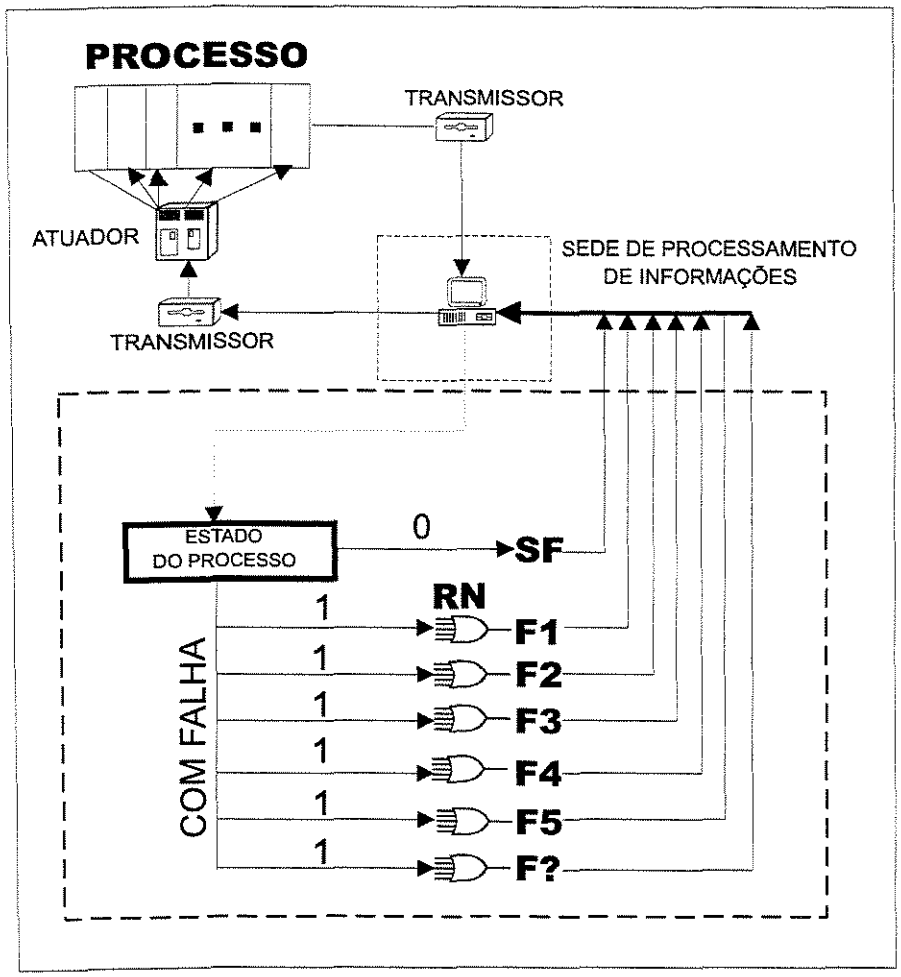


Figura 7.15: Esquema de funcionamento do procedimento de redes neurais para a determinação da presença ou não de falhas num determinado sistema de processo. RN corresponde à rede neural de várias entradas e com apenas uma saída. Testa-se primeiro para o caso normal. Se não tiver falha, o sistema retorna o valor zero (SF), ou outro valor conforme convenção a ser adotada. Se houver falha, indicado pela resposta diferente de zero, o sistema confere cada uma das redes treinadas, até encontrar uma saída inferenciada que seja igual à atribuída àquela rede treinada. F? é alguma situação de falha que pode ser acrescentada às cinco anteriormente previstas. Caso nenhum dos testes produza um valor igual ao utilizado para treinar cada uma das redes neurais, o sistema retorna um aviso de falha desconhecida. Apenas detectada mas não diagnosticada.

A seguir, no CASO 6, estão apresentadas tabelas que resumem, para o sistema estudado, o número de iterações necessárias para se treinar uma rede com saída unitária que

será utilizada posteriormente na montagem de uma rede composta de várias subredes. No caso deste trabalho, onde apenas 05 falhas foram definidas como possíveis de ocorrer durante o funcionamento do sistema, a verificação fica na dependência de uma precisão de inferenciamento muito menor. Na casa de décimos. Assim, pode-se dividir o intervalo 0.00 a 1.00 de forma a se ter redes cujas saídas são, 0,30, 0,45, 0,60, 0,75 e 0,90. Segundo esta divisão, pode-se utilizar o valor 0,15 como correspondente à situação de normalidade nas condições de funcionamento, e os demais, progressivamente, estão associados às falhas de F1 a F5.

A Figura 7.15 mostra esquematicamente o funcionamento da rede montada para se determinar a falha que está ocorrendo. Segundo este esquema, caso alguma situação de falha ocorra, a sede de processamento pode, por comparação, determinar qual a falha que está corrompendo o sistema e desta forma enviar uma ordem de correção para o atuador ou para o operador que esteja controlando a planta.

7.8.6 CASO 6

Conjunto de 700 padrões, dados para um tempo de amostragem de 100 minutos para os sete níveis de falhas de cada uma das falhas estudadas para a utilização do mecanismo proposto e detalhado na Figura 7.15.

7.8.6.1: 700 AMOSTRAS/AJUSTE SEA/UM NEURÔNIO DE SAÍDA

N	SIGMA	ETA	ALFA	TOL	DQ	MDA	MXER	NME	NITER
1	0.125	0.90	0.80	0.01	0.0099	0.0421	42.16%	1/8	05
2	0.0125	0.90	0.80	0.01	0.0099	0.0213	21.33%	1/98	13
3	0.00125	0.90	0.80	0.01	0.0095	0.0225	22.53%	1/222	14
4	1.25	0.90	0.80	0.01	---	---	---	---	NC

Continuação

N	SIGMA	ETA	ALFA	TOL	DQ	MDA	MXER	NME	NITER
5	0.125	0.92	0.80	0.01	0.0099	0.0421	42.12%	1/8	05
6	0.125	0.88	0.80	0.01	0.0099	0.0422	42.19%	1/8	05
7	0.125	0.90	0.82	0.01	0.0084	0.0381	38.08%	1/8	05
8	0.125	0.90	0.78	0.01	0.0068	0.0347	34.72%	1/8	06
9	0.125	0.90	0.82	0.001	9.9E-04	0.0105	10.52%	1/37	53
10	0.125	0.90	0.82	1E-04	9.9E-05	0.0038	3.86%	1/40	874
11	0.125	0.90	0.82	1E-05	2.6E-05	0.0016	1.56%	1/32	2130
12	0.125	0.90	0.82	1E-05	9.9E-06	9.5E-04	0.96%	1/30	3262
13	0.125	0.90	0.82	1E-06	2.7E-06	4.8E-04	0.48%	1/30	4972
14	0.125	0.90	0.82	1E-06	1.2E-06	3.3E-04	0.33%	1/18	6163
15	0.125	0.90	0.82	1E-06	9.9E-07	2.9E-04	0.29%	1/18	6500

Tabela 7.28: Dados dos 700 amostras após ocorrer a falha. Subrede treinada com apenas dados da primeira falha, para montagem da rede geral.

7.8.6.2: 700 AMOSTRAS/AJUSTE SEA/UM NEURÔNIO DE SAÍDA

N	SIGMA	ETA	ALFA	TOL	DQ	MDA	MXER	NME	NITER
1	0.125	0.90	0.80	0.01	0.0099	---	---	---	NC
2	0.0125	0.90	0.80	0.01	0.0093	0.0175	5.83%	1/82	11
3	0.00125	0.90	0.80	0.01	0.0087	0.0161	5.38%	1/50	10
4	1.25E04	0.90	0.80	0.01	0.0087	0.0161	5.38%	1/50	10
5	0.00125	0.92	0.80	0.01	0.0083	0.0156	5.20%	1/50	10
6	0.00125	0.88	0.80	0.01	0.0092	0.0167	5.56%	1/50	10
7	0.00125	0.92	0.82	0.01	0.0084	0.0156	5.22%	1/50	09
8	0.00125	0.92	0.84	0.01	0.0085	0.0157	5.23%	1/50	08
9	0.00125	0.92	0.86	0.01	0.0086	0.0157	5.25%	1/50	07
10	0.00125	0.92	0.88	0.01	0.0089	0.0156	5.22%	1/50	06
11	0.00125	0.92	0.90	0.01	0.0092	0.0159	5.29%	1/50	05

Continuação

N	SIGMA	ETA	ALFA	TOL	DQ	MDA	MXER	NME	NITER
12	0.00125	0.92	0.92	0.01	0.0097	0.0159	5.31%	1/50	04
13	0.00125	0.92	0.94	0.01	0.0045	0.0159	3.55%	1/60	04
14	0.00125	0.92	0.94	0.001	7.5E-04	4.7E-03	1.56%	1/81	06
15	0.00125	0.92	0.94	1E-04	4.2E-05	1.2E-03	0.40%	1/113	09
16	0.00125	0.92	0.94	1E-05	5.3E-06	4.8E-04	0.16%	1/117	11
17	0.00125	0.92	0.94	1E-06	7.2E-07	1.8E-04	0.06%	1/117	13
18	0.00125	0.92	0.94	1E-07	2.0E-07	1.1E-04	0.04%	1/117	14
19	0.00125	0.92	0.94	1E-08	2.0E-07	1.1E-04	0.04%	1/117	14

Tabela 7.29: Dados dos 700 amostras após ocorrer a falha. Subrede treinada com apenas dados da segunda falha, para montagem da rede geral.

7.8.6.3: 700 AMOSTRAS/AJUSTE SEA/UM NEURÔNIO DE SAÍDA

N	SIGMA	ETA	ALFA	TOL	DQ	MDA	MXER	NME	NITER
1	0.125	0.90	0.80	0.01	---	---	---	---	NC
2	0.0125	0.90	0.80	0.01	0.0087	0.0192	4.27%	1/106	09
3	0.00125	0.90	0.80	0.01	0.0074	0.0156	3.47%	1/105	09
4	1.25E-4	0.90	0.80	0.01	0.0074	0.0156	3.47%	1/105	09
5	0.00125	0.90	0.80	0.01	0.0096	0.0175	3.89%	1/105	08
6	0.00125	0.92	0.80	0.01	0.0091	0.0171	3.80%	1/105	08
7	0.00125	0.94	0.80	0.01	0.0073	0.0156	3.46%	1/105	08
8	0.00125	0.92	0.82	0.01	0.0089	0.0169	3.77%	1/105	09
9	0.00125	0.92	0.78	0.001	8.6E-04	0.0055	1.22%	1/105	14
10	0.00125	0.92	0.82	1E-04	9.8E-05	1.9E-03	0.43%	1/104	20
11	0.00125	0.92	0.82	1E-05	7.2E-06	5.4E-04	0.12%	1/104	27
12	0.00125	0.92	0.82	1E-06	1.1E-06	2.0E-04	0.05%	1/108	32
13	0.00125	0.92	0.82	1E-07	1.1E-06	2.0E-04	0.05%	1/108	32

Tabela 7.30: Dados de 700 amostras após ocorrer a falha. Subrede treinada apenas com dados da terceira falha, para montagem da rede geral.

7.8.6.4: 700 AMOSTRAS/AJUSTE SEA/UM NEURÔNIO DE SAÍDA

N	SIGMA	ETA	ALFA	TOL	DQ	MDA	MXER	NME	NITER
1	0.125	0.90	0.80	0.01	---	---	---	---	NC
2	0.0125	0.90	0.80	0.01	0.0098	0.0265	4.42%	1/44	12
3	0.00125	0.90	0.80	0.01	0.0079	0.0181	3.02%	1/426	09
4	1.25E-4	0.90	0.80	0.01	0.0079	0.0180	3.01%	1/426	09
5	0.00125	0.92	0.80	0.01	0.0074	0.0176	2.94%	1/426	09
6	0.00125	0.88	0.80	0.01	0.0084	0.0186	3.10%	1/426	09
7	0.00125	0.92	0.82	0.01	0.0078	0.0181	3.02%	1/426	08
8	0.00125	0.92	0.84	0.01	0.0083	0.0187	3.11%	1/426	07
9	0.00125	0.92	0.86	0.01	0.0091	0.0194	3.24%	1/426	06
10	0.00125	0.92	0.88	0.01	0.0058	0.0161	2.68%	1/426	06
11	0.00125	0.92	0.88	0.001	7.9E-04	5.9E-03	0.98%	1/425	10
12	0.00125	0.92	0.88	1E-04	7.1E-05	2.1E-03	0.35%	1/425	14
13	0.00125	0.92	0.88	1E-05	7.1E-06	7.6E-04	0.11%	1/433	18
14	0.00125	0.92	0.88	1E-06	8.9E-07	2.6E-04	0.04%	1/433	21
15	0.00125	0.92	0.88	1E-07	8.9E-07	2.6E-04	0.04%	1/433	21

Tabela 7.31: Dados de 700 amostras após ocorrer a falha. Subrede treinada com apenas dados da quarta falha, para montagem da rede geral.

7.8.6.5: 700 AMOSTRAS/AJUSTE SEA/UM NEURÔNIO DE SAÍDA

N	SIGMA	ETA	ALFA	TOL	DQ	MDA	MXER	NME	NITER
1	0.125	0.90	0.80	0.01	0.0094	0.0330	3.67%	1/39	11
2	0.0125	0.90	0.80	0.01	0.0096	0.0221	2.49%	1/19	16
3	0.0125	0.92	0.80	0.01	0.0099	0.0227	2.52%	1/19	15
4	0.0125	0.94	0.80	0.01	0.0098	0.0225	2.51%	1/19	15
5	0.0125	0.92	0.82	0.01	0.0097	0.0225	2.50%	1/19	14
6	0.0125	0.92	0.84	0.01	0.0095	0.0222	2.47%	1/19	13

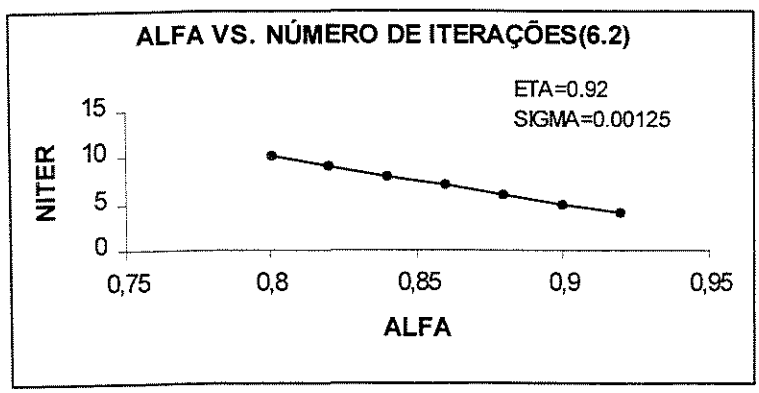
Continuação

N	SIGMA	ETA	ALFA	TOL	DQ	MDA	MXER	NME	NITER
7	0.0125	0.92	0.86	0.01	0.0098	0.0225	2.50%	1/19	11
8	0.0125	0.92	0.88	0.01	0.0094	0.0222	2.47%	1/19	10
9	0.0125	0.92	0.90	0.01	0.0099	0.0226	2.51%	1/19	08
10	0.0125	0.92	0.92	0.01	0.0095	0.0221	2.46%	1/19	07
11	0.0125	0.92	0.94	0.01	0.0088	0.0213	2.37%	1/19	06
12	0.0125	0.92	0.96	0.01	0.0094	0.0214	2.38%	1/19	04
13	0.0125	0.92	0.97	0.01	0.0095	0.0261	2.90%	1/19	06
14	0.0125	0.92	0.98	0.01	---	---	---	---	NC
15	0.0125	0.92	0.96	0.001	8.9E-04	4.9E-03	0.55%	1/59	16
16	0.0125	0.92	0.96	1E-04	9.7E-05	1.2E-03	0.14%	1/121	28
17	0.0125	0.92	0.96	1E-05	9.7E-06	4.2E-04	0.05%	1/121	40
18	0.0125	0.92	0.96	1E-06	9.7E-06	4.2E-04	0.05%	1/121	40

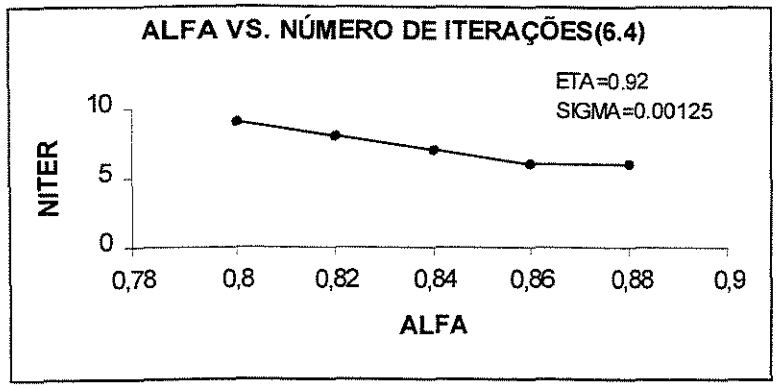
Tabela 7.32: Dados de 700 amostras após ocorrer a falha. Subrede treinada com apenas dados da primeira falha, para montagem da rede geral.

CURVAS DE AJUSTE DOS PARÂMETROS E DA PRECISÃO DE TREINAMENTO VERSUS O NÚMERO DE ITERAÇÕES PARA O CASO 6

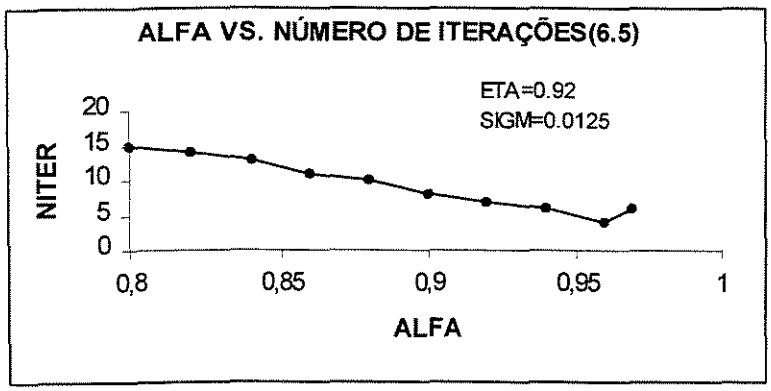
VALORES DE ALFA VERSUS NÚMERO DE ITERAÇÕES



(a)



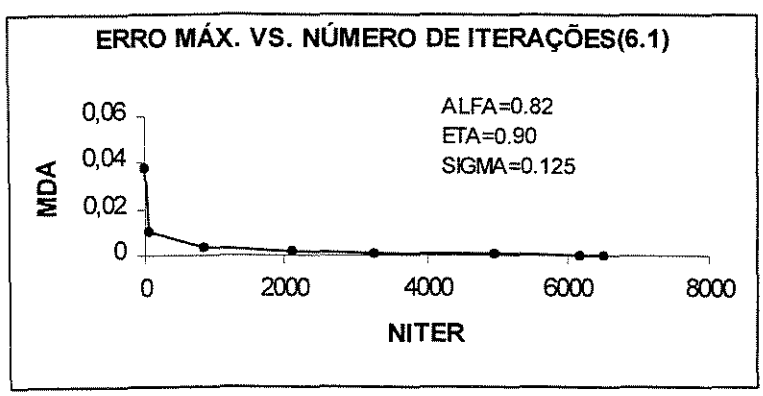
(b)



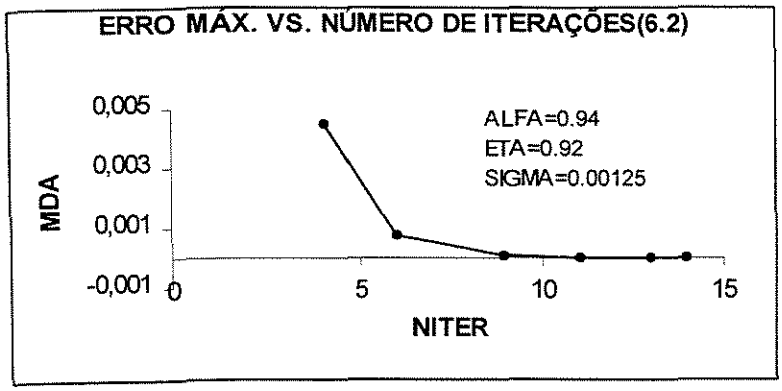
(c)

Figura 7.16: Comportamento dos valores de ALFA, para o CASO 6, onde são treinadas redes com os valores das variáveis controladas nos primeiros 500 minutos após o aparecimento da falha. Cada rede conta com apenas uma saída, treinadas para identificar apenas um erro específico. A rede representada por (a) é treinada para detectar uma falha do tipo F2; a rede representada por (b) é treinada para detectar a presença de uma falha do tipo F4; e finalmente a rede representada por (c) é treinada para detectar uma falha do tipo F5.

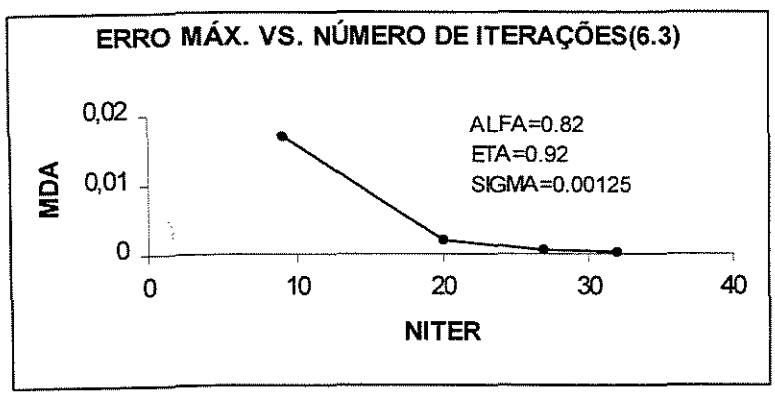
ERROS ABSOLUTOS MÁXIMOS VERSUS NÚMERO DE ITERAÇÕES



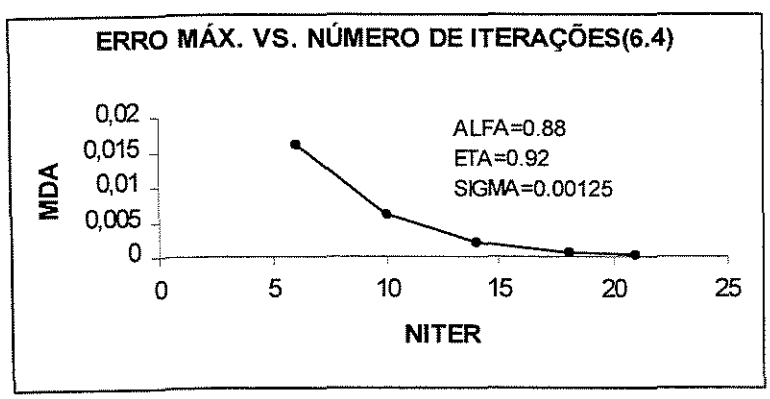
(a)


ERRO MÁX. VS. NÚMERO DE ITERAÇÕES(6.2)
MDA
0,005
0,003
0,001
-0,001
0
5
10
15
NITER
ALFA=0.94
ETA=0.92
SIGMA=0.00125

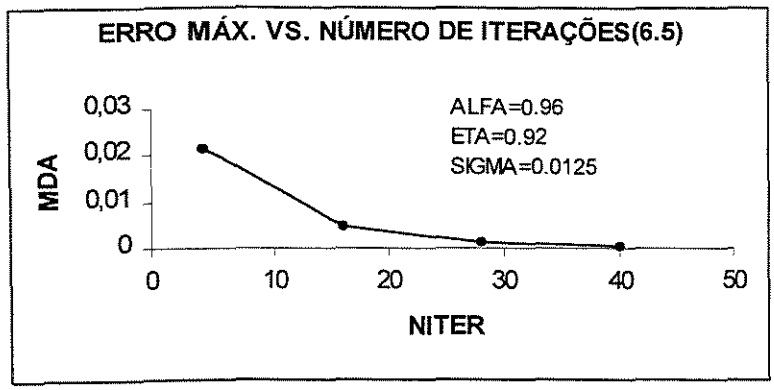
(b)


ERRO MÁX. VS. NÚMERO DE ITERAÇÕES(6.3)
MDA
0,02
0,01
0
0
10
20
30
40
NITER
ALFA=0.82
ETA=0.92
SIGMA=0.00125

(c)


ERRO MÁX. VS. NÚMERO DE ITERAÇÕES(6.4)
MDA
0,02
0,015
0,01
0,005
0
0
5
10
15
20
25
NITER
ALFA=0.88
ETA=0.92
SIGMA=0.00125

(d)



(e)

Figura 7.17: Comparação do número de iterações necessárias para se atingir determinados patamares de precisão, no ajuste dos pesos das redes neurais, quando se tem os valores dos parâmetros já ajustados na forma final. Cada uma das redes treinadas é capaz de detectar apenas uma falha , que vai de F1 a F5, representadas pelos gráficos de (a) a (e).

Não estão apresentadas as curvas para **ETA vs. NITER** e algumas de **ALFA vs. NITER** pelo mesmo motivo anteriormente citado. Os dados para se traçar as curvas não são suficientes.

Análise do CASO 6

Neste CASO, por terem sido utilizados para cada falha um conjunto contendo todos os níveis das mesmas, o inferenciamento apresentou 100% de acertos.

A Maior dificuldade em treinar a rede com a falha F1 pode ser imputada aos níveis de falhas utilizados.a maior oscilação entre os valores dos parâmetros monitoradoscausou uma maior dificuldade no treinamento da respectiva rede.

Até aqui, o que se fez foi determinar corretamente as falhas ocorridas mesmo que a níveis diferentes dos utilizados para o treinamento das redes, sem no entanto identificar, qual seria este nível. O que se propõe neste ponto é um novo mecanismo de agrupamento de padrões e de redes neurais de tal forma que seja possível identificar se não o nível da falha, pelo menos sua faixa de ocorrência.

A fim de se poder testar os procedimentos aqui desenvolvidos na determinação do nível, ou da extensão da falha que está ocorrendo, propôs-se um esquema de treinamento das redes neurais baseado no desenho mostrado na Figura 7.18. Na primeira parte do estudo, que corresponde apenas à parte superior da referida figura, tem-se uma rede com seis neurônios de entrada, representando, como antes, as variáveis que estão sendo monitoradas, um neurônio de saída com representações para o estado de falha e representações para o estado de normalidade ou de não falha. Até este ponto não é possível diagnosticar a falha, apenas detectar a sua presença sem no entanto identificá-la. Os resultados deste procedimento estão sumarizados nos itens de 7.5.22 a 7.5.26[53]. Caso a quantidade de padrões provenientes do sistema a ser utilizada no treinamento da rede seja muito grande não permitindo o treinamento da mesma com uma quantidade de iterações razoável, pode-se alternativamente, usar apenas os dados da condição de normalidade do sistema. Caso o sistema esteja normal, o inferenciamento deve produzir uma saída para a rede igual à utilizada para o treinamento da mesma, sendo diferente, tem-se evidência de falha.

O próximo passo deste arranjo das redes neurais apresenta-se na segunda etapa da Figura 7.18, onde é possível se determinar não só a presença da falha como também onde ela ocorre, ou seja diagnosticá-la. Como no CASO 6, uma vez constatada a presença da falha, apresenta-se os valores das variáveis monitoradas a cada uma das cinco redes. Aquela cuja saída produzir um resultado em concordância com o qual a mesma foi treinada, identifica a falha que está ocorrendo no sistema. O terceiro passo vai além e possibilita também determinar o nível de cada uma das três falhas para as quais a rede foi treinada. Para tanto, pode-se utilizar uma nova rede com a mesma arquitetura do CASO 7, item 7.5.22, exceto para a saída, onde apesar de existir apenas um neurônio, podem ser atribuídos valores diferentes para cada falha diferente, seguindo o mesmo raciocínio explicitado no CASO 6, onde se aproveitou a grande precisão possível de ser alcançada com esta arquitetura de rede.

Uma outra maneira alternativa de se fazer o diagnóstico da falha, antes da determinação do seu nível, seria mudando-se a rede para uma com a mesma entrada mas ao invés de uma saída, fazê-la com seis saídas sendo que cada uma representa uma das cinco

---

[53] Aqui foi possível o treinamento com todas as falhas devido ao pequeno tamanho da rede capaz de detectar

possíveis situações de falha mais a condição de normalidade. O segundo passo seguiria o mesmo caminho descrito para a forma utilizada no CASO anterior a este. Não se tentou determinar as iterações necessárias para este caso uma vez que de experiências passadas pode-se saber que o número de iterações necessárias ao treinamento da rede seria muito maior para o CASO de se ter seis neurônios de saída do que para o CASO de se ter apenas um.

Uma vez que com um neurônio de saída é possível obter o mesmo resultado, com menos esforço computacional, opta-se, logicamente, por este modelo em detrimento do outro.

---

somente cinco falhas.

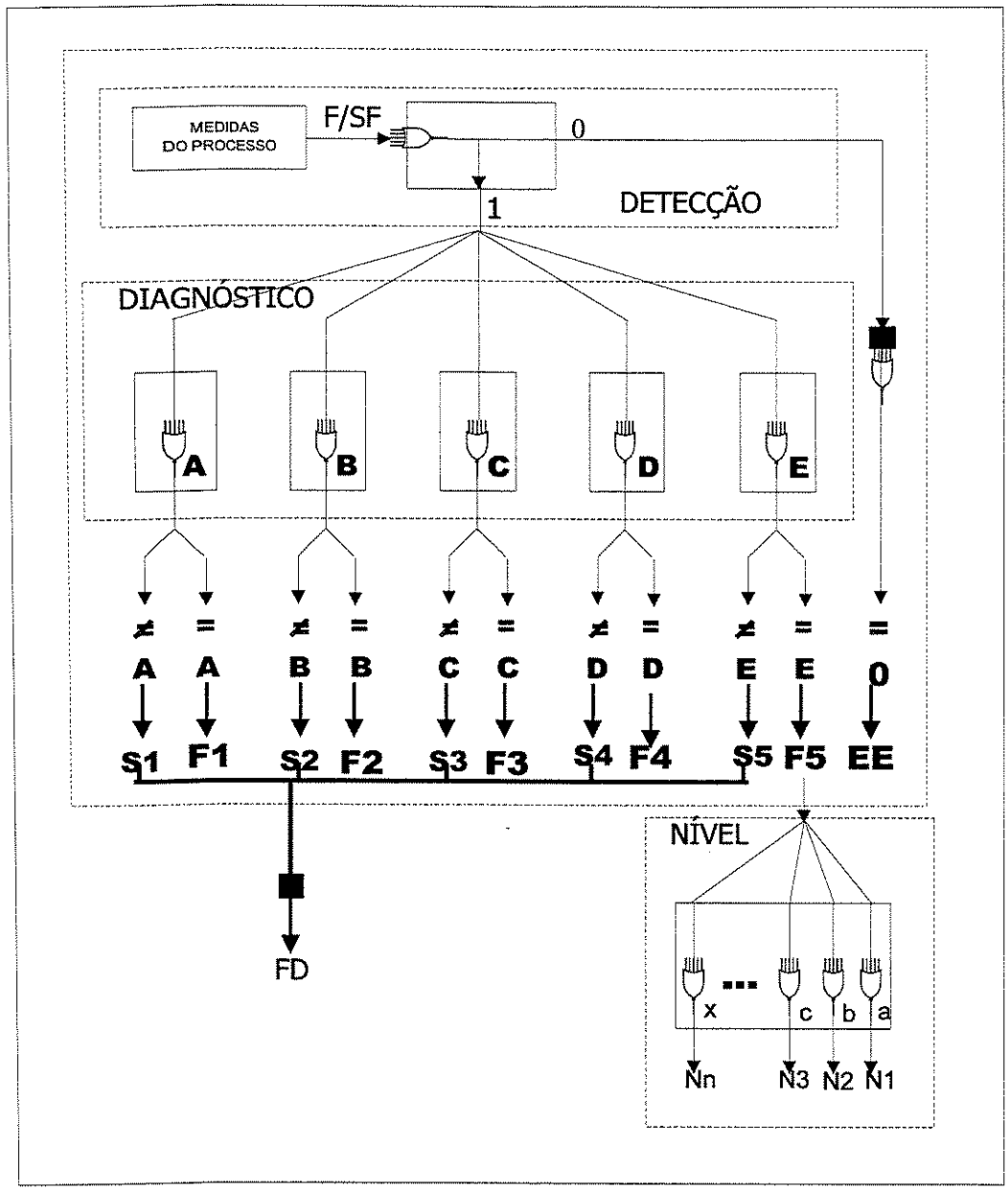


Figura 7.18: Esquema de funcionamento do procedimento de determinação do nível de uma falha presente em um sistema. Uma vez detectada a falha, o mecanismo de diagnóstico é acionado. A, B, C, D e E são os valores esperados à saída das redes em cada condição de falha para os quais a rede está treinada. Caso não exista falha, o sistema interrompe o procedimento. Se uma das condições de igualdade se verifica, a terceira parte do procedimento é então acionada. A cada falha corresponde um conjunto de subredes conforme explicitado para F5. Se uma falha necessariamente ocorre e todas as condições de desigualdade se verificam ao mesmo tempo, tem-se um erro desconhecido, ou seja, a rede não foi treinada para detectá-lo e retorna algum sinal equivalente a FD.

7.8.7 CASO 7

7.8.7.1: 100 MINUTOS/AJUSTE SEA/APENAS DETECÇÃO/5 NÍVEIS DE COMPROMETIMENTO PARA CADA FALHA

N	SIGMA	ETA	ALFA	TOL	DQ	MDA	MXER	NME	NITER
1	0.125	0.90	0.80	0.01	---	---	---	---	NC
2	0.0125	0.90	0.80	0.01	0.0099	0.1301	14.46%	1/301	47
3	0.00125	0.90	0.80	0.01	0.0096	0.0249	2.78%	1/501	20
4	1.25E-4	0.90	0.80	0.01	0.00096	0.0249	2.78%	1/501	20
5	0.00125	0.92	0.80	0.01	0.0098	0.0253	2.81%	1/501	19
6	0.00125	0.94	0.80	0.01	0.0096	0.0251	2.79%	1/501	19
7	0.00125	0.92	0.82	0.01	0.0099	0.0254	2.82%	1/501	17
8	0.00125	0.92	0.84	0.01	0.0095	0.0248	2.77%	1/501	16
9	0.00125	0.92	0.86	0.01	0.0095	0.0249	2.70%	1/501	14
10	0.00125	0.92	0.88	0.01	0.0097	0.0251	2.79%	1/501	12
11	0.00125	0.92	0.90	0.01	0.0098	0.0252	2.81%	1/501	10
12	0.00125	0.92	0.92	0.01	0.0068	0.0214	2.39%	1/501	11
13	0.00125	0.92	0.90	0.001	8.8E-04	0.0082	0.91%	1/75	31
14	0.00125	0.92	0.90	1E-04	9.4E-05	0.0026	0.30%	1/75	48
15	0.00125	0.92	0.90	1E-05	9.5E-06	8.2E-04	0.09%	1/106	65
16	0.00125	0.92	0.90	1E-06	2.7E-06	4.4E-04	0.05%	1/106	74
17	0.00125	0.92	0.90	1E-07	2.7E-06	4.4E-04	0.05%	1/106	74

Tabela 7.33: Dados dos 100 primeiros minutos após ocorrer a falha, com cinco níveis de extensão da falha para as variáveis monitoradas e mais um padrão correspondente à condição normal. Rede apenas para detecção de falha sem que seja feito o seu diagnóstico.

RESULTADOS DO TREINAMENTO DAS SUBREDES PARA A DETERMINAÇÃO DA FALHA E DO NÍVEL DA MESMA

Os dados referentes ao treinamento das redes apresentadas nos CASO 7 são destinados à determinação da falha que está ocorrendo. Para cada uma das falhas, F1,F2, F3,F4 e F5, são utilizados cinco níveis. Cada um dos níveis está representado por 100 minutos de simulação o que dá um total de 500 padrões amostrados para cada falha.

Os dados referentes ao treinamento das redes apresentados nos CASOS 7.8.1.20 7.8.1.22 e 7.8.1.23 são destinados à determinação dos níveis das falhas F1, F3 e F5 respectivamente. Cada uma das três possíveis de se determinar cada um dos níveis em que foram treinadas as redes.

7.8.7.2: 100 MINUTOS/AJUSTE SEA/F1/F2/F3/F4/F5

Cada uma das linhas desta tabela representa uma falha. O conjunto de padrões para cada falha envolve 5 níveis.

NA	SIGMA	ETA	ALFA	TOL	DQ	MDA	MXER	NME	NITER
500	0.0125	0.90	0.82	1E-07	9.9E-08	3.2E-04	0.30%	1/200	11395
500	0.0125	0.90	0.80	1E-07	1.2E-08	2.4E-04	0.05%	1/29	22
500	0.0125	0.90	0.80	1E-07	2.1E-08	4.5E-04	0.05%	1/228	931
500	0.0125	0.90	0.82	1E-07	7.0E-08	1.1E-04	0.04%	1/24	33
500	0.0125	0.90	0.80	1E-07	4.6E-08	3.2E-04	0.04%	1/20	30

Figura 7.34: Dados do comportamento dinâmico da coluna de destilação até os primeiros 100 minutos para as falhas. **NA** é o número de amostras utilizadas na subrede (500 amostras correspondentes a 100 minutos de funcionamento do sistema após ocorrer a falha, para cada um dos 05 níveis da falha, o que corresponde a um total de 500 amostras). Cada linha da tabela corresponde a uma falha das cinco falhas. Apenas os resultados finais do treinamento estão aqui descritos.

Pode-se notar que no último CASO apresentado, a rede mais difícil de ser treinada foi a que contém os dados com a falha F1. Justamente esta falha é a que apresenta os maiores níveis, ou seja, maior magnitude de comprometimento das variáveis monitoradas, o que por sua vez faz com que sejam menos contínuos seus comportamentos, dificultando o treinamento da rede.

7.8.7.3: 100 MINUTOS/ AJUSTE SEA/F1 A VÁRIOS NÍVEIS

Cada linha representa 1 nível da falha F1. A saída de cada rede é exatamente o nível de cada falha.

F1	SIGMA	ETA	ALFA	TOL	DQ	MDA	MXER	NME	NITER
0.5RR	0.0125	0.90	0.82	1E-07	9.9E-08	3.1E-04	0.31%	1/49	14949
0.8RR	0.0125	0.90	0.80	1E-07	9.7E-07	4.1E-04	0.16%	1/38	6194
1.1RR	0.0125	0.90	0.80	1E-07	2.9E-07	2.4E-04	0.05%	1/9	19
1.3RR	0.0125	0.90	0.80	1E-07	7.7E-07	3.7E-04	0.05%	1/27	29
1.5RR	0.0125	0.90	0.80	1E-07	1.1E-07	4.5E-04	0.05%	1/32	108

Tabela 7.35: Dados dos 100 primeiros minutos após ocorrer a falha. Os valores na primeira coluna indicam o nível da falha, por exemplo para uma coluna operando em estado estacionário, quando subitamente, ocorre uma perturbação em degrau que diminui o valor de sua razão de refluxo de 50%, para o primeiro nível, e assim sucessivamente. Cada linha corresponde aos melhores resultados, ou resultados finais, para cada subrede.

7.8.7.4: 100 MINUTOS/ AJUSTE SEA/F3 A VÁRIOS NÍVEIS

Cada linha representa 1 nível da falha F3. A saída da rede é o valor de cada nível.

F3	SIGMA	ETA	ALFA	TOL	DQ	MDA	MXER	NME	NITER
0.7Q	0.0125	0.90	0.80	1E-08	9.9E-09	7.5E-05	0.06%	1/43	13933
0.8Q	0.0125	0.90	0.80	1E-08	2.0E-08	1.1E-04	0.05	1/48	49
0.9Q	0.0125	0.90	0.80	1E-08	1.3E-07	2.4E-04	0.05	1/23	24
1.1Q	0.0125	0.90	0.80	1E-08	2.7E-07	3.7E-04	0.05	1/47	69
1.3Q	0.0125	0.90	0.80	1E-08	1.2E-06	4.4E-04	0.05	1/7	109

Tabela 7.36: Dados dos 100 primeiros minutos após ocorrer a falha. Os valores na primeira coluna indicam o nível da falha, por exemplo para uma coluna operando em estado estacionário, quando subitamente, ocorre uma perturbação em degrau que diminui o valor de sua razão de fornecimento de calor ao refervedor de 30%, para o primeiro nível, e assim sucessivamente. Cada linha corresponde aos melhores resultados, ou resultados finais, para cada subrede.

7.8.7.5: 100 MINUTOS/ AJUSTE SEA/F5 A VÁRIOS NÍVEIS

Cada linha representa um nível da falha F5. A saída da rede é o valor do nível.

F5	SIGMA	ETA	ALFA	TOL	DQ	MDA	MXER	NME	NITER
0.6FL	0.0125	0.90	0.80	1E-08	1.0E-08	7.9E-05	008.%	1/47	429
0.8FL	0.0125	0.90	0.80	1E-07	1.7E-07	1.2E-04	0.05%	1/37	201
1.1FL	0.0125	0.90	0.80	1E-07	4.2E-07	4.7E-04	0.93%	1/22	461
1.3FL	0.0125	0.90	0.80	1E-07	4.4E-07	3.3E-04	0.04%	1/49	32
1.5FL	0.0125	0.90	0.80	1E-07	7.9E-07	4.5E-04	0.05%	1/14	112

Tabela 7.37: Dados dos 100 primeiros minutos após ocorrer a falha. Os valores na primeira coluna indicam o nível da falha, por exemplo para uma coluna operando em estado estacionário, quando subitamente, ocorre uma perturbação em degrau que diminui no valor de sua taxa de alimentação da coluna de 40%, para o primeiro nível, e assim sucessivamente. Cada linha corresponde aos melhores resultados, ou resultados finais, para cada subrede.

Análise do CASO 7

A utilização de cinco níveis de falhas para o treinamento das redes, neste CASO, possibilitou detectar e diagnosticar corretamente em 100% dos CASOS as falhas quando se fez o inferenciamento, com situações não usadas no treinamento, para um tempo de 10 minutos após ocorrer a falha.

A determinação dos níveis das mesmas só foi possível de forma exata quando se usou os valores de inferenciamento também utilizados para o treinamento. Quando se utilizou valores intermediários para o inferenciamento, o índice de determinação dos níveis, de forma exata, foi menor que 100%.

Os resultados para os níveis inferenciados não utilizados no treinamento são dados na tabela abaixo, Tabela 7.39.

FALHA	NE(1)	NC(1)	DIF %	NE(2)	NC(2)	DIF %
F1	0,70	0,61		1,20	1,28	
F3	0,75	0,71		1,20	1,17	
F5	0,70	0,74		1,20	1,11	

Tabela 7.38.: Diferenças de inferenciamento do nível das falhas cujos valores não foram usados no treinamento das redes neurais.

Em termos de valores normalizados, as diferenças não parecem grandes mas levando-se em conta que estes valores devem ser transformados novamente em valores absolutos para se ter os verdadeiros valores das variáveis monitoradas, vê-se que podem ser, sim, significativos. Contudo, o procedimento é plenamente aceitável quando se procura pelo menos ter-se uma idéia aproximada do nível da falha.

Dois aspectos devem ser notados quando se analisa as quatro últimas tabelas. Em primeiro lugar, pode-se notar que quanto mais se afasta do estado estacionário, mais o procedimento tem dificuldades em treinar as redes neurais, o que fica evidenciado na quantidade de iterações necessárias para o treinamento das mesmas. Isto pode ser entendido pelo fato de quanto maior é a falha, maiores são os distúrbios que a mesma causa nas variáveis monitoradas e desta forma menos homogêneo será o conjunto de amostras, modificando a característica de continuidade do comportamento das variáveis, dificultando o treinamento das redes.

Em segundo lugar, deve-se notar a proximidade entre os valores dos parâmetros da rede neural em todos os exemplos. Isto se deve à forma de atualização dos mesmos utilizada neste trabalho, conforme explicitado anteriormente, no início do EXEMPLO 5. Como os conjuntos de amostras na maioria das vezes era homogêneo, quase sempre, com o conjunto de valores iniciais já se atingia um valor de iterações bem pequeno, tornando quase impossível baixá-lo apenas com a mudança dos valores dos parâmetros. Deste modo, os respectivos valores eram conservados e apenas a TOL era mudada a fim de se chegar aos valores finais referentes ao treinamento da respectiva rede.

Uma última forma de detecção diagnóstico que estava previsto no projeto deste trabalho, se refere à possibilidade de detecção de múltiplas falhas. Segundo alguns dos autores citados, situações como esta são menos incomuns que se possa pensar à primeira vista. Imagine-se que se tenha duas ou mais válvulas ou controladores comandados por energia elétrica ou por diferenças de pressão, no caso dos aparelhos pneumáticos e que um distúrbio ocorra na transmissão de energia ou na linha de transmissão pneumática. Logicamente esta falha pode estar prevista dentre aquelas que estão sendo estudadas, contudo, se este não for o caso, pode-se ter uma situação em que duas ou mais falhas ocorram simultaneamente. Isto gerará uma situação imprevista em termos de comportamento destas variáveis.

É sempre possível se treinar uma rede prevendo-se este tipo de situação, contudo, não é sempre viável fazê-lo devido à grande combinação de situações possíveis de se ter que enfrentar. Neste CASO o melhor que se teria a fazer seria incluir estas possíveis situações como casos de falha previstas. Um outro aspecto que pode ser levado em conta quando se tem qualquer tipo de falha seja ela unitária ou múltipla é o fato de que se se tem uma rede treinada para situações de falhas e para a situação de normalidade, pode-se "reservar um valor" diferente dos utilizados para o treinamento para indicar que alguma anormalidade, mesmo que desconhecida ocorreu. Uma vez que não seja possível se saber qual a causa da falha nem onde ela ocorre, é melhor se ter uma indicação de que algo errado está ocorrendo com a planta, que não se ter nenhuma indicação ou uma indicação incorreta de falha.

Como forma de ilustração[54] destes CASOS treinou-se duas redes neurais com uma situação em que duas ou três falhas apareciam simultaneamente. Os resultados estão sumarizados na tabelas do CASO 8.

---

[54] Não se preocupou aqui em encontrar uma relação que pudesse funcionar como causa para uma ocorrência de falhas simultâneas F1 e F3, como por exemplo a apresentada no item 7.5.27, ou das falhas F1, F3 e F5 como no item 7.5.28.

7.8.8 CASO 8

Simulando-se durante 100 minutos o comportamento dinâmico do sistema com duas (F1 e F3) ou três (F1, F3 e F5) falhas como ocorrendo simultaneamente, tem-se 1000 ou 1000 padrões respectivamente, para o treinamento da rede, uma vez que são utilizados 5 níveis diferentes.

7.8.8.1 :100 MINUTOS/AJUSTE SEA/DETECÇÃO DE MÚLTIPLAS FALHAS(2)

N	SIGMA	ETA	ALFA	TOL	DQ	MDA	MXER	NME	NITER
1	0.125	0.90	0.80	0.01	0.0089	0.0697	7.62%	1/51	11
2	0.0125	0.90	0.80	0.01	0.0095	0.0326	3.62%	1/60	05
3	0.00125	0.90	0.80	0.01	0.0099	0.0228	2.53%	1/51	19
4	0.0125	0.92	0.80	0.01	0.0093	0.0323	3.59%	1/60	05
5	0.0125	0.92	0.82	0.01	0.0083	0.0311	3.45%	1/60	05
6	0.0125	0.92	0.82	0.001	9.4E-04	0.0097	1.07%	1/60	20
7	0.0125	0.92	0.82	1E-04	9.9E-05	0.0058	0.65%	1/60	1944
8	0.0125	0.92	0.82	1E-05	9.9E-06	0.0018	0.21%	1/125	9901
9	0.0125	0.92	0.82	1E-06	9.9E-07	5.9E-04	0.06%	1/125	16104
10	0.0125	0.92	0.82	1E-07	5.9E-07	4.5E-04	0.05%	1/122	17771

Tabela 7.39: Resultados do treinamento de uma rede neural representando um sistema no qual ocorrem duas falhas simultâneas. F1 e F3.

7.8.8.2: 100 MINUTOS/AJUSTE SEA/DETECÇÃO DE MÚLTIPLAS FALHAS(3)

N	SIGMA	ETA	ALFA	TOL	DQ	MDA	MXER	NME	NITER
1	0.125	0.90	0.80	0.01	0.0085	0.0568	7.32%	1/52	11

Continuação

N	SIGMA	ETA	ALFA	TOL	DQ	MDA	MXER	NME	NITER
2	0.0125	0.90	0.80	0.01	0.0099	0.0970	10.78%	1/11	11
3	0.125	0.92	0.80	0.01	0.0099	0.0556	7.18%	1/52	11
4	0.125	0.92	0.82	0.01	0.0082	0.0574	7.38%	1/52	10
5	0.125	0.92	0.84	0.01	0.0072	0.0541	7.01%	1/52	09
6	0.125	0.92	0.86	0.01	0.0071	0.0559	7.21%	1/52	14
7	0.125	0.92	0.84	1E-03	0.0014	0.0161	1.71%	1/75	31
8	0.125	0.92	0.84	1E-04	9.9E-05	0.0112	1.24%	1/75	396
9	0.125	0.92	0.84	1E-05	9.9E-06	0.0051	0.56%	1/81	3156
10	0.125	0.92	0.84	1E-06	9.7E-07	0.0031	0.34%	1/81	10197

Tabela 7.40: Resultados do treinamento de uma rede neural representando um sistema no qual ocorrem três falhas simultâneas. F1, F3 e F5.

O inferenciamento utilizando-se a rede treinada com o mesmo conjunto de falhas, apenas com níveis diferentes, se mostrou possível. Para um inferenciamento a 10 minutos do início das falhas F1 e F3 foi possível mudando-se apenas os níveis das mesmas , tanto separada quando simultaneamente, detectar e diagnosticar múltiplas falhas.

Mudando-se o nível para os parâmetros (um de cada vez) somente quando se variou em –30% a taxa de refluxo não foi possível reconhecer a presença das falhas múltiplas F1 e F3, corretamente. Isto sempre tomando-se valores dos parâmetros monitorados a 10 minutos depois de ocorrida a falha. Esta dificuldade já havia se manifestado no CASO do item 7.8.1.24. Aqui foi utilizada uma rede com saídas 0 e 1, sendo o 0 associado à falha F1 e o 1 associado à falha F3. Valores que se afastaram mais que 20% de 0 e 1 foram desprezados e considerados como errados.

Para três falhas simultâneas, ocorre algo semelhante que para duas. Para o inferenciamento com níveis diferentes das três falhas, a 10 minutos da ocorrência da mesma, foi sempre possível identificar, seja de forma combinada ou separada a variação dos níveis. Quando se variou os parâmetros, (sempre 1 de cada vez), novamente para a taxa de refluxo 30% mais baixa, não funcionou corretamente.

Entretanto, não só a mudança nos valores da razão de refluxo impediu a associação correta. A mudança de +20% na alimentação e na quantidade de calor do refervedor

também impediu as respostas de serem corretas. Neste CASO associou-se 0 a F1 0,5 a F3 e 1 a F5. Diferenças maiores que 20% foram consideradas como erradas.

CAPÍTULO 8

8.1 CONCLUSÕES

O estudo de reconhecimento de padrões através da utilização de redes neurais, mostra que existe uma extensa quantidade de possibilidades de combinação de entradas e saídas, agrupamentos, variações de arquiteturas e ajuste de parâmetros para o treinamento das mesmas. O fato de existirem muitas maneiras diferentes de se reunir os parâmetros a fim de se proceder o treinamento de uma rede neural pode dificultar em muito o trabalho de quem as está utilizando, quando o conjunto de dados ou padrões para treinamento for muito grande. Desta forma, quando se tem um conjunto de dados com um determinado 'comportamento', a procura pelo melhor conjunto destes parâmetros pode consumir mais tempo do que se não se utilizasse um conjunto otimizado fazendo-se o ajuste dos pesos com uma combinação qualquer que possa treinar as redes com o nível de precisão requerido pelo problema.

Durante o desenvolvimento deste trabalho, foram testadas diversas maneiras de se proceder a esta escolha de parâmetros até que fosse possível se encontrar, de uma forma sistemática, a melhor maneira de atualização dos mesmos, dentro das características dos dados do sistema dinâmico utilizado. Assim, em termos de redes neurais, as contribuições dadas por este trabalho são:

- apresentação de extensa literatura a respeito dos modelos e métodos diferentes de treinamento de redes neurais;

- agrupamento de melhoramentos sugeridos por diversos autores, conforme a descrição apresentada ao longo do trabalho e mais especificamente nos capítulos 2, 5 e 6, e desenvolvimento do programa de treinamento de redes neurais eficiente;

- desenvolvimento de uma metodologia de ajuste dos parâmetros seguindo uma ordem tal que sempre é possível, de acordo com a precisão que se deseje, encontrar um conjunto

de parâmetros que faça o treinamento das redes neurais com o menor número de iterações possíveis;

- estudo da influência de cada um dos três parâmetros variáveis na redes com funções radiais base no número de iterações necessárias para o treinamento das redes;

- implementação, no programa, de um mecanismo para se saber a cada instante do treinamento, o ponto que está apresentando o maior desvio absoluto, o que possibilita decidir por descartá-lo ou não;

- possibilidade de se saber ao fim do treinamento exatamente onde está o padrão-problema, que dificulta o treinamento daquele conjunto de dados, mostrando onde está o padrão que serviu como elemento de parada para o critério de precisão adotado.

Depois de se encontrar uma forma eficiente, ainda que não ótima no sentido estrito do termo, de se treinar redes neurais para complexos conjuntos de dados, foi possível testar o que era a proposição central deste trabalho, a importância do conhecimento dos estados intermediários durante a ocorrência de uma falha em um sistema de processos. Pelo fato de que dificilmente se sabe a que nível uma falha vai ocorrer, é muito importante se ter um mecanismo de controle do sistema que seja o menos dependente, se não completamente independente, do fator nível da falha. Os estudos apresentados na literatura têm pouca eficiência prática uma vez que para se reconhecer uma falha, o sistema precisa entrar em um outro estado estacionário depois de ocorrida a falha. Esta entrada em novo estado estacionário pode ser muito demorada ou nem mesmo ocorrer, o que não é bom para o processo pois desbalanceia-o fazendo com que, no mínimo, se esteja operando fora das condições ideais.

Em termos dos mecanismos de detecção e diagnóstico de falhas, as principais contribuições deste trabalho são:

- desenvolvimento de um programa computacional para treinamento de uma coluna de destilação multicomponente com considerações rigorosas, conforme descrito no capítulo 4;

- proposição de uma nova forma de analisar os dados de um sistema de processos a fim de se detectar anomalias no mesmo. Ao invés de analisar dados em estado estacionário, utiliza-se em qualquer instante, em qualquer condição de funcionamento do sistema;

- desenvolvimento de um mecanismo de divisão de tempo de amostragem/tempo de verificação que permite fazer o controle das condições do processo de acordo com o conjunto de dados utilizados para o treinamento das redes neurais, (Figura 7.1);

- simulação a vários intervalos de tempos, e com vários níveis de corrompimento, das condições de funcionamento da planta, uma vez acontecida a falha;

- simulação de ocorrência de duas ou mais falhas simultaneamente a fim de se testar o mecanismo de detecção/diagnóstico para um sistema nestas condições, ou seja, com múltiplas falhas;

- análise da possibilidade de detecção/diagnóstico de falhas corretamente, sem se ter usado o padrão que apresenta a falha no treinamento da rede neural;

- desenvolvimento de um mecanismo de detecção/diagnóstico de falhas para sistemas de grandes dimensões, (Figura 7.15);

- desenvolvimento de um mecanismo para detecção/diagnóstico e determinação do nível da falha que está ocorrendo, (Figura 7.18), que independe do nível da falha;

Os mecanismos aqui propostos, de se usar os valores de alguns parâmetros de monitoramento, nos estados intermediários, mostrou-se eficiente e perfeitamente implementável quando a ocorrência da falha ainda é incipiente. Isto traz como vantagem o fato de se poder identificar pouco depois de ocorrida, uma falha no sistema. A utilização de valores de parâmetros de monitoramento para inferenciamento do estado do processo (falha/não-falha) para perturbações ocorridas a níveis diferentes dos utilizados para treinamento das redes neurais se mostrou seguro, em algumas condições, sendo capaz de associar, corretamente, a resposta a um padrão desconhecido com base nos dados conhecidos. Na prática, isto significa que é possível, utilizando uma divisão de tempos de amostragem conveniente e uma distribuição dos níveis de desvio de cada variável monitorada de forma balanceada, detectar e diagnosticar corretamente uma falha num sistema de processos.

Em conclusão, pôde-se comprovar a tese inicialmente proposta que era a de estudar a importância do conhecimento dos estados intermediários de um sistema dinâmico na detecção e no diagnóstico de falhas em sistemas de processos. A importância deste conhecimento pôde ser acompanhada através dos diversos esquemas propostos, desenvolvidos e testados neste trabalho. A possibilidade de ter um controle melhorado do sistema "sem precisar se preocupar" com o nível que a falha vá ocorrer.

8.2 SUGESTÕES PARA TRABALHOS FUTUROS

A utilização de simuladores quando se deseja estudar o comportamento dinâmico de um sistema de processos oferece valiosos subsídios para se compreender os rumos que o mesmo tomará face a determinadas mudanças nos valores das variáveis que o regulam.

Entretanto, alguns condicionantes não levados em conta quando da modelagem do sistema que se quer simular, pode trazer desvios nos valores de algumas das variáveis de forma que o que se está simulando guarda pouca ou, às vezes, nenhuma relação com os dados reais.

Pensando nisto e consciente que o modelo de coluna de destilação aqui implementado, apesar de ser rigoroso pode - fator menos relevante dentro dos objetivos deste trabalho - fornecer valores para as variáveis manipuladas e controladas diferentes daqueles valores reais, até mesmo pela possibilidade de existência de ruídos no sistema - seria interessante se utilizar dados de um processo real a fim de testar todo o mecanismo proposto e testado com êxito neste trabalho.

Quanto ao modelo de redes neurais aqui utilizado, com RBF, pode-se dizer que possibilitou alcançar os objetivos do trabalho. Permitiram treinar grandes conjuntos de dados e conseguiu recuperar muito bem os valores para os quais foram treinadas e encontrar padrões 'suficientemente próximos' para valores não utilizados no treinamento. O aumento da escala do sistema de processo pode exigir um refinamento maior que o seguido aqui, aspecto que se mostrou possível ao longo de todo o capítulo de resultados quando se demonstrou que pequenas mudanças nos valores dos seus parâmetros implicavam variações significativas no número de iterações necessárias ao seu treinamento. Esta seria então uma segunda sugestão para futuros trabalhos na área.

Como terceira proposta, se poderia incluir a mudança de algoritmo visto que a cada dia novos modelos de treinamento de redes aparecem, conforme pode ser observado na extensa análise bibliográfica apresentada no capítulo 2 deste trabalho. Algoritmos que utilizam modelos genéticos, lógica fuzzy ou redutores de dimensionalidade do sistema como por exemplo análise do componente principal, podem ser testados para melhorar o desempenho das redes neurais.

Como quarta sugestão, fica a utilização de um modelo híbrido, utilizando, agora, o já testado conhecimento de dados dinâmicos do sistema, para detecção/diagnóstico de falhas seguindo idéias de autores, antes citados, de que a combinação da análise qualitativa aliada à análise quantitativa é a que fornece os melhores resultados quando se intenta localizar e analisar falhas em um sistema de processos.

Finalmente, pode-se sugerir, que utilizando as redes neurais se faça um estudo do número ótimo de variáveis necessárias para um perfeito monitoramento do processo de

forma que se possa ter informações acerca de algumas variáveis sem que seja preciso medi-las. A cooptação de resultados é importante para se diminuir o custo de instalações uma vez que permite a utilização de menor quantidade de instrumentos de medida além de diminuir a quantidade de variáveis a serem manipuladas ao se fazer a análise do sistema em estudo.

CAPÍTULO 9

9.0 BIBLIOGRAFIA

ADAMS, A.; STERLING, L., (1992), "*Evap: The Genesis of a Industrial Expert System*" , **Proceedings of the 5th Australian Joint Conference on Artificial Intelligence, AI '92,** pp7-12.

AFONSO, P.A.F.N.A.; FERREIRA, J.M.L.; CASTRO, J.A.A.M., (1998), "*Sensor Fault Detection and Identification in a Pilot Plant under Process Control*", **Trans IchemE,** vol76, Part A, pp 490-498.

AFONSO, P.A.F.N.A. ; CASTRO, J.A.A.M., (1995), "*An Experimental Evaluation of Fault Detection and Identification Scheme in a Chemical Process*" , **IChemE Research Event,** v1, pp25-27.

ALT, F.B., (1982),"*Multivariate Quality Control*", **in S. Kotz and N.L. Johnson, (editors), Encyclopedia of Statistical Sciences**, pp110-122. Wiley, New York.

AN, G., (1996), "*The Effects of Adding Noise During Backpropagation Training on a Generalization Performance*", **Neural Computation**, n8, pp643-674.

BAGAJEWICZ, M.J.; JIANG, Q., (1998), "*Gross Error Modeling and Detection in Plant Linear Dynamic Reconciliation*", **Computers and Chemical Engineering**, v22, n12, pp1789-1809.

BAINBRIDGE, L., (1981), "*Mathematical Equations or Processing Routines?*", **in J. Rasmussen and W.B. Rouse, (editors), Human Detection and Diagnosis of Systems Failures**. Plenum Press: New York.

BARRON, A.R., (1993), "*Universal Approximation Bounds for Superpositions of a Sigmoid Function*", **IEEE Transactions on Information Theory**, 39, pp930-945.

BASSEVILLE, M., (1998), "*Detecting Changes in Signals and Systems – Survey*", **Automatica**, v24, n3, pp309-326.

BATCHELOR, B.G.; WILKINS, B.R, (1969), " *Method for Location of Clusters of Patterns to Inicialize a Learning Machine*", **Electronic Letter** 5, pp481-483.

BAUM, E.B.,(1988), "*On Capabilities of Multilayer Perceptrons*", **J. Complexity,**v4,pp193-215.

BECRAFT, W.R.; LEE, P.L., (1991), "*Using Neural Networks for Diagnosing Focus in Chemical Process Plants*" , **Proceedings. Fourth International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE-91)** pp345-354, v1.

BECRAFT, W.R.; LEE, P.L., (1993), "*An Integrated Neural Network Expert System Approach for Fault Diagnosis*", **Computers and Chemical Engineering (Oxford),** v17, Issue:10, pp1001-1014.

BENEDETTO, J.J.; FRAZIER, M.W., (1994), "*Wavelets – Mathematics and Applications*", CRC Press, 575pp.

BEIREITER, S.R.; MILLER, S.M., (1988), "*Sources of Difficulty in Troubleshooting Automated Manufacturing Systems*", in **W. Karwowski, H.R. Parsaes e M.R.Wilhelm. Ergonomics of Hybrid Automated Systems I**. Elsevier Publishing, B.V.Amsterdam.

BERTSEKAS, D.P.; TSITSIKLIS, J.N., (1996), "*Neuro-Dynamic Programming*", **Belmont, MA: Athena Scientific, ISBN 1-886529-10-8.**

BIARDI, F.; ROVAGLIO, C.; MEOTTI, M.G.; PELLEGRINI, L., (1987), "*Simulation of Real Trays Fractionating Units: A New Rigorous Approach*", **Proceedings of CEF**, v87p489, Taormina.

BISHOP, C.M., (1991), "*Improving the Generalization Properties of Radial Basis Function Neural Networks*", **Neural Computation**, v3, n4, pp229-236.

BISHOP, C.M., (1995), "*Neural Networks for Pattern Recognition*", Oxford: Oxford University Press.

BLUE, J.L.; HALL, L.O, (1996), "*Collapsing Multiple Hidden Layers in Feedforward Neural Networks to a single Hidden Layer*", **SPIE Proceedings**, v2760, pp44-52, Orlando, FL. USA.

BRENAN, K.E.; CAMPBELL, S.L.; PETZOLD, L.R, (1990), "*The Numerical Solution of Initial Value Problems in Differential-Algebraic Equations*", Elsevier Science Publishing Co.

BUNTINE, W.L.; WEIGAND, A.S., (1991), "*Bayesian Back-Propagation*", **Complex Systems** 5, pp603-643.

BROOMHEAD, D.S; LOWE, D., (1988), "*Multivariable Functional Interpolation and Adaptive Networks*", **Complex Systems** 5, pp321-355.

BROWNSTON, L.; FARREL, E. K.; MARTIN, N., (1985), "*Programming Expert System in OPS5: An Introduction to Rule-Based Programming*", Addison-Wesley, Reading, MA.

BURDSAL, B.; GIRAUD-CARRIER,C., (1998), "*GA-RBF: A Self-Optimising RBF Network*", ENST de Bretagne, BP82 (burdsall@gti.ernst-bretagne.fr) e University of Bristol (cgc@cs.bris.ac.uk).

CARNAHAN, B.; LUTHER, H.A; WILKES, J.O., (1969), "*Applied Numerical Methods*", John Wiley and Sons.

CARPENTER, G.A.; GROSSBERG, S.,(1986), "*Neural Dynamics of Category Learning and Recognition: Attention, Memory Consolidation, and Amnesia*", **in J. Davis, R. Newburgh, and E. Wegan (Editors) Brain Structure, Learning and Memory, AAAS Symposium Series.**

CHAN, C.W.; LAU, P., (1999), "*Representing User in Engineering Design Domains Using na Enhanced Weighted Fuzzy Reasoning Algorithm*", **Artificial Intelligence in Engineering**, n13, pp 1-10.

CHANG, CHUEI-TIN; CHEN, JEN-WEN, (1995), "*Implementation Issues Concerning the Ekf-Based Fault Diagnosis Techniques*", **Chemical Engineering Science,** v50, Issue:18, pp2861-2882.

CHEN, S.; BILLINGS, S.A., LUO, W., (1989), "*Orthogonal Least Squares Methods and their Application to Nonlinear System Identification*", **International Journal of Control**, 50,(5), pp1873-1896.

CHEN, S.; COWAN, C.F.N.; GRANT, P.M., (1991), "*Orthogonal Least Squares Learning algorithm for Radial Basis Function Networks*", **IEEE Transactions on Neural Networks**, 2(2), pp302-309.

CHEN, B.H.; WANG, X.Z.; YANG, S.H.; McGREAVY, C., (1999), "*Application of Wavelets and Neural Networks to Diagnostic System Development, 1, Feature Extraction*", **Computers and Chemical Engineering**, v23, pp899-906.

CHEN, B.H.; WANG, X.Z.; YANG, S.H.; McGREAVY, C., (1999), "*Application of Wavelets and Neural Networks to Diagnostic System Development, 2, Na Integratedm Framework and its Application*", **Computers and Chemical Engineering**, v23, pp906.

CHII-SHANG TSAI E CHUEI-TIN CHANG, (1995), "*Dynamic Process Diagnosis via Integrated Neural Networks*", **Computers and Chemical Engineering**, v19,Suppl., ppS747-S752.

CHING, E.; YANG,H.; SHARBEK, W., (1995), "*Reduced Complexity Implemenation of the Bayesian Equalizer using Locla RBF Network for Channel Equalization Problem*", SPIE Proceedings, v2760, pp56-59, Orlando, FL. USA.

CHO, Y.S; JOSEPH, B., (1983a), "Reduced-Order Steady-State and Dynamic Models for Separation Process – part I. Development of the Model Reduction Procedure", **AIChe. Journal**, v29, n2, pp 261-269.

CHO, Y.S.; JOSEPH, B., (1983b), "*Reduced-Order Steady-State and Dynamic Models for Separation Process – part II. Application to Nonlinear Multicomponent Systems*", **AIChe. Journal**, v29, n2, pp270-2276.

CILLIERS, J. J., (1992), "*Neural Networks for Safety Steady-State Process Modelling and Fault Diagnosis*", **Expert Systems in Mineral and Metal Processing. Proceedings of the IFAC Workshop,** pp161-165.

COVER, T.M.,(1961), "*Geometrical and Statistical Properties of Linear Threshold Devices*", Ph.D. Thesis, Tech. Rep. 6107-1, Stanford Eletron. Labs., Stanford CA.

COURANT, R.; HILBERT, D., (1962), "*Methods of Mathematical Physics*", v2, Interscience, London, England.

CVETKOVIC, Z.; VETTERLI, M., (1995), "*Discrete Time Wavelet Extrema*", **Transactions on Signal Processing,** n43, pp681-693.

CYBENKO, B., (1989), "*Approximation by Superposition of a Sigmoidal Function*", **Math. Control. Signals Systems,** n2, pp303-314.

DARKEN, C.; MOODY, J., (1992), "*Towards Faster Stochastic Gradient Search*", in Moody, J.E., Hanson, S.J., and Lippmann, R.P., **Advances in Neural Information Processing Systems 4,** pp1009-1016.

DECOSTE, D., (1990), "*Dynamic Across-Time Measurement Interpretation*", **in Proceedings of Eighth National Conference on Artificial Intelligence (AAAI-90),** pp373-379, AAAI Press, Cambridge, MA.

DISTEFANO, G.P.; 1968), AIChe Journal, v14, n1.

DOIG, I., (1986), "*Process Diagnostic Exercises*", CACHE Corp., Austin, TX.

DRAPER, N.R.; SMITH, H., (1995), "*Applied Regression Analysis*", 2nd ed. John Wiley and Sons.

DUCHON, J., (1976), "*Splines Minimizing Rotation -Invariant Semi-Norms in Sobolev Space*", In: Schempp & Zeller (Eds.): **Constrative Theory of Functions of Several Variables, Lectures Notes in Mathematics**, 571, pp85-100.

DUDA, R.O.; HART, P.E.,(1973), "*Pattern Classification and Scene Analysis*", Wiley, New York.

DUNIA, R.; QIN, S. J.; EDGAR, T. F.; McAVOY, T, J., (1995), "*Use of Principal Component Analysis for Sensor Fault Identification*", **Computers and Chemical Engineering,** v20, Iss:pt A, Suppl.is ppS713-718.

DUNIA, R.; QIN, J.S., (1998), "*A Unified Geometric Approach to Process and Sensor Fault Identification and Reconstruction: The Unidimensional Fault Case*", **Computers and Chemical Engineering**, v22, n7-8, pp927-943.

DVORAK, D; KIUPERS, B.J., (1991), " *Process Monitoring and Diagnosis*", **IEEE Expert**, v6, n3, pp67-74.

EVERITT, B.S., (1993), "*Cluster Analysis*", John Wiley and Sons, 3rd edition, New York.

EFRON; B., (1983), "*Estimating the Error Rate of a Prediction Rule: Improvement in Cross Validation*", **Journal of the American Statistical Association**, n78, pp316-331.

EFRON, B., (1986), "*How Biased is the Apparent Error Rate of a Prediction Rule*?", **J. of the American Statistical Association**, 81, 461-470.

FAHLMAN, S.E., (1988), "*An Empirical Study of Learning Speed in Backpropagation Networks*", **Tech Rep. CMU-CS-88-162**, School of Computer Science, Carnegie Mellon University, Pittsburg, PA.

FAN, J. Y.; NIKOLAOU, M.; WHITE, R. E., (1993), "*An Approach to Fault Diagnosis of Chemical Processes via Neural Networks*", **AIChe. Journal,** v39, Issue:1, pp82-88.

FINCH, F.E.; KRAMER, M.A., (1987), "*Narrowing Diagnostic Focus Using Functional Decomposition*", **AIChe. Journal**, v34, n1, pp130-140.

FORBUS, K., (1984), "*Qualitative Process Theory*", **Artificial Intelligence,** v24, pp46-53.

FRANKE, R., (1982), "*Scattered Data Interpolation: Tests of Some Methods*", **Mathematics and Computing**, 38, pp181-200.

FRANK, P.M., (1990), "*Fault Diagnosis in Dynamic Systems Using Analitical and Knowledge-Based Redundancy – A Survey and Some New Results*", **Automatica**, v26, n3, pp459-474.

FREDESLUND, A.; GMEHLING, J.; RASMUSSEN, P., (1977), "*Vapor-liquid Equilibria Using UNIFAC*", Amsterdam: Elsevier.

FRITZKE, B., (1994), "*Fast Learning with Incremental RBF Networks*", **Neural Processing Letters**, v1, n1, pp2-5.

FRIEDMAN, J.H.; STUELZE, W., (1981), "*Projection Pursuit Regression*", **Journal of American Statistical Association**, 76 (376), pp817-823.

FUKUNAGA, K., (1982), "*Introduction to Statistical Pattern Recognition*", 2nd edition, San Diego, Academic Press.

FURUOKA, Y.; MATSUKI, H.; MINAMITANI, H.; ISHIDA, A., (1998), "*A Modified Back-Propagation Method to Avoid False Local Minima*", **Neural Networks**, N11, pp1059-1072.

GEAR, C.W., (1971), "*Numerical initial Value Problems in Ordinary Differential Equations*", Prentice Hall.

GEMAN, S.; BIENENSTOCK, E.; DOURSA, R., (1992),"*Neural Networks and the Bias/Variance Dilemma*", **Neural Computation**, 4 (1), pp1-58.

GERTLER, J., (1991), "*Analytical Redundancy Methods in Fault Detection and Isolation - Survey and Synthesis*", **Proceedings of the IFAC SFEPROCESS Symposium**, Baden – Baden, Germany.

GERTLER, J., (1992), "*Structured Residuals for Fault Isolation, Disturbance Decoupling and Modelling Error Robustness*", **Proceedings of the IFAC Symposium Online Fault Detection and Supervision in the Chemical Process Industries**.

GHAHRAMANI, Z.; JORDAN, I.M., (1994), "*Supervised Learning from Incomplete Data via an EM Approach*", in J.D. Cowan, G.T. Tesauro, and J. Alspector (editors),

Advances in Neural Information Processing Systems, v6, pp120-127. San Mateo, CA: Morgan Kauffmann.

AL-GHONEIM, K.A; KUMAR, B.V.K.V., (1996), "*Combining Neural Networks Using the Ranking Figure of Merit*", SPIE Proceedings, v2760, pp 2-13, Orlando, FL, USA.

GILES, C.L.; MAXWELL, T., (1987), "*Learning, Invariance, and Generalization in High-Order Neural Networks*", **Applied Optics**, 26(23), pp4972-4978.

GILL, P.E.; MURRAY, W.; WRIGHT, M.H., (1981), "*Practical Optimization*", Academic Press 6th edition.

GIORDANO, R., et al., (1984), Anais do VI Congresso Brasileiro de Engenharia Química, São Paulo, UNICAMP.

GIROSI, F.; POGGIO, T., (1990), "*Networks and the Best Approximation Property*", **Biological Cybernetics** n63, pp169-176.

GOMM, J. B., (1994), "*Fault Detection in a Multivariable Chemical Process by Monitoring Process Dynamics*", **IFAC Symposium on Fault Detection, Supervision and Safety for Technical Process- SAFER|PROCESS '94**. Peprints pp177-82, v1.

GOUETTE, C, (1997), "*Note on Free Lunches and Cross-validation*", **Neural computation**, 9, pp1211-1215.

GRANTAN, S. D.; UNGAR, L. H., (1990), "*A First Principles Approach to Automated Troubleshooting of Chemical Plants*" , **Computers and Chemical Engineering**, v14, n7, pp783-798.

GROSSBERG, S., (1976), "*Adaptative Pattern Classification and Universal Recording, II: Feedback, Expectation, Olfation, and Illusions*", **Biolog. Cybernetics,** vol 23, pp187-202.

GREENSTEIN, J.S.; ROUSE, W.B., (1982), "*A Model of Human Decision Making in Multiple Process Monitoring Situations*", **IEEE Trans Syst and Cyber.**, smc-12, pp182-187.

HAGAN, M. T. ; MENHAJ, M. B., (1994), "*Training Feedforward Networks with the Marquardt Algorithm*" , "**IEEE Transactions on Neural Networks",** v5, n6, nov.1994.

HARTMAN, E.J.; KEELER, J.D., (1990), "*Layered Neural Networks with Gaussian Hidden Units as Universal Approximations*", **Neural Computation** vol2, n2, pp210-215.

HASTIE, T.J.; TIBSHIRANI, R.J., (1990), "*Generalized Additive Models*", London, Chapman and Hall.

HERTZ, J; KROG, A.; PALMER, R. G., (1991), "*Introduction to the Theory of Neural Computation*" , **Reading, MA**: Addison-Wesley

HIMMELBLAU, D.M., (1983), "*Fault detection and Diagnosis in Chemical and Petrochemical Process*", Elsevier publications, New York,NY .

HIMMELBLAU, D. M., (1992), "*Fault Detection in Heat Exchangers*" , **Proceedings of the American Control Conference v3. Publ by American Automatic Control Council, Green Valley, AZ, USA**. pp269-2372.

HINTON, G.E.; SEJNOVSKI, R.J., (1986), "*Learning and Relearning in Boltzmann Machines*", **in Parallel distributed Processing, v1, ch7, D. E. Rumelhart and J.L. McClelland, (Editors),** Cambridge, MA, MIT Press.

HINTON, G.E.; SEJNOVSKI, R.J.; ACKLEY, D.H., (1984), "*Boltzmann Machines: Constraint Satisfaction Networks that Learn*", **Tech. Rep. CMU-CS-84-119,** Carnegie-Mellon University, Dept of Computer Science.

HOLLAND,C.H.,(1963), "*Multicomponent Distillation*", Prentice-Hall,

HOPFIELD, J. J., (1982), "*Neural Networks and Physical Systems with Emergent Colletive Computacional Abilities*", **Proc. Nat. Academy Sci**. 79, 2554-2558(1982).

HORNIK, K.M.; STINCHCOMBE, M.; WHITE, H., (1989), "*Mulilayer Feedforward Networks are Universal Approximators*", **Neural Networks**, n2, pp359-366.

HOSKINS, J. C.; HIMMELBLAU, D. M., (1988), "*Artificial Neural Network Models of Knowledge Representation in Chemical Engineering*" , **Computers and Chemical Engineering**, v12,n9/10,pp881-890.

HOSKINS, J. C.; KALIYUR, K. M. and HIMMELBLAU, D. M., (1991), "*Fault Diagnosis in Complex Chemical Plants Using Artificial Neural Networks*" , **AIChe. Journal**, v37, n1, pp137-141.

HUSSAIN, M.A.,(1999), "*Review of the Applications of Neural Networks in Chemical Process Control – Simulation and Online Implementation*", **Artificial Intelligence in Engineering**, n13, pp55-68.

ISERMANN, R., (1984), "*Fault Detection Based on Modelling and Estimation Methods: A Survey*", **Automatica**, n20, pp387-404.

JACKSON, I.R., (1988), "*Convergence Properties of Radial Basis Functions*", **Constructive Approximation**, 4, pp243-264.

JACKSON, J.E., (1985), "*Multivariate Quality Control*", **Communications in Statistics – Theory and Methods**, v14, n11, pp2657-2688.

JOKINEN, P. A.; (1991), "*Comparison of Neural Network Models for Process Fault Detection and Diagnosis Problems*", **Int Jt Conf Neural Networks IJCNN 91 Seattle. Publ by IEEE, IEEE Service Center, Piscataway, NJ, USA,** (IEEEcat n91CH3049-4). pp239-244.

JONES B.; J. C. HOSKINS,(1987), "*Backpropagation*", **BYTE** 12, 155-162(1987).

KAMINSKI, W.; STRUMILLO, P., (1997), "*Kernel Orthonormalization in Radial Basis Function Neural Networks*", **IEEE Transactions on Neural Networks**, v8, n5.

KASSIDAS, A.; TAYLOR, P.A.; MacGREGOR, J.F., (1998), "Off-Line Diagnosis of Deterministic Faults in Continuous of Dynamic Multivariable Process Using Speech Recognization Methods", Journal of Proc. Cont., v8, n5-6, pp381-393.

KAVURI, S.V., (1993), "*Robust Fault Diagnosis of Process Systems Using Neural Networks with Ellipsoidal Units*", **PhD Thesis**, Purdue University.

KAVURI, S. V.; VENKATASUBRAMANIAN, V., (1994), "*Neural Network Decomposition Strategies for Large-Scale Fault Diagnosis*" , **International Journal of Control,** v59, Iss:3, pp767-792.

KAVURI, S.N.; VENKATASUBRAMANIAN, V.,(1993), "*Using Fuzzy Clustering with Ellipsoidal Units in Neural Networks for Robust Fault Classification*", **Computers and Chemical Engineering**, v17, pp765-784.

KAVURI, S.N.; VENKATASUBRAMANIAN, V.,(1993), "*Representing Bounded Fault Classes Using Neural Networks With Ellipsoidal Activation Functions*", **Computers and Chemical Engineering,** v17,n2,pp139-163.

KENDRA, S. J.; BASILA, M. R.; CINAR, A., (1994), "*Intelligent Process Control with Supervisory Knowledge-Based Systems*" , **IEEE Control Systems Magazine,** v14, Iss:3, pp37-47.

KELLER, J. -Y.; DAROUACH, M. and KRZAKALA, G., (1994), "*Fault detection of Multiple Biases or Process Leaks in Linear Steady State Systems*" , **Computers and Chemical Engineering**, v18, n10, pp1001-1004.

KERR, T.H., (1987), "*Decentralised Filtered and Redundancy Management for Multisensor Navigation*", **IEEE Transactions on Aerospace and Electronic Systems**, vAES-23, pp83-119.

KING, C.J., (1980), "*Separation Process*", McGraw-Hill, New York, 2nd edition.

KOHONEN, T., (1984), "*Self-Organisation and Associative Memory*", Springer-Verlag, Berlin.

KOHONEN,T., (1990), "*Self-Organising Maps*", **Proceedings of IEEE**, vol78, n9, pp1464-1480.

KOHONEN, T., (1995), "*Self-Organising Maps*", Berlin , Springer-Verlag.

KOLEN , J.F.; POLLACK, J.B.; (1990), "*Backpropagation is Sensitive to Initial Conditions*", Lab. Artificial Intell. Res., Comput. Inform. Sci. Dep, Tech. Rep. TR 90-JK-BPSIC,1990.

KRAMER, M. A., (1987) "*Malfunction Diagnosis Using Quantitative Models With Non-Boolean Reasoning In Expert Systems*", **Computers and Chemical Engineering, v12, pp881.**

KRAMER, M. A.; PALOWITCH, Jr., B.L., (1985), "*Expert System and Knowledge based Approaches to Process Malfunction Diagnosis*" , **AIChe. Nat. Meet., Chicago**.

KRAMER, M. A.; PALOWITCH, Jr., B.L. (1989), "*A Rule-Based Approach to Fault Diagnosis Using the signed Digraph*", **AIChe Journal**, v33, n7, pp1067-1068

KUMAR, A., (1982), "*Chemical Process Synthesis and Engineering Design*", McGraw-Hill, Nova Delhi.

KUSHNER, H.J.; YIN, G., (1997), "*Stochastic Approximation Algorithms and Applications*", N.Y., Springer-Verlag.

LAPP, S.A.; POWERS, G.A., (1977), "*Computer-Aided Synthesis of Fault Trees*", **IEEE Trans. Reability, R-37**, pp 2-13.

LAWRENCE, S.; GILES, C.L.,; TSOI, A.C., (1996), "*What Size Neural Network Gives Optimal Generalization? Convergence Properties of Backpropagtion*", Technical Report UMIACS-TR-96-22 and CS-TR-3617, Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742.

LeCUN, Y.; SYMARD, P.Y., and PEARLMETTER, B., (1993), "*Automatic Learning Rate Maximisation by on-line Estimation of the Hessian Eigenvalues*", in Hanson , S.J. Cowan, and Giles, C.L. (editors), **Advances in Neural Information Processing System 5**, San Mateo, CA: Morgan Kauffmann, pp156-163.

LEGER, R.P.; GARLAND, J.Wm.; POEHLMAN, W.F.S., (1998), "*Fault Detection and Diagnosis Using Statistical Control Charts and Artificial Neural Networks*", **Artificial Intelligence in Engineering**, n12, pp35-47.

LIPPMANN, R.P., (1987), "*An Introduction to Computing with Neural Nets*", **IEEE ASSP Magazine,** n4 , pp04-22.

LLOYD, S.P., (1982), "*Least Squares Quantization in PCM*", Bell Laboratories Internal Report. **IEEE Transactions on Information Theory** v28, pp127-130.

LOPARO, K.A.; BUCHNER, M.R.; VASUDEVA, K.S., (1991), "*Leak Detection in a Experimental Heat Exchanger Process: A Multiple Model Approach*", **IEEE Transactions on Automatic Control**, v36, n2, pp167-177.

LORENTZ, G.G., (1976), "*The 13th Problem of Hilbert*", in F.E. Browder (Ed.), Mathematical Developments Arising from Hilbert Problems, American Mathematical Society, Providence, R.I.

LUYBEN, W.L., (1990), "*Process Modelling , Simulation and Control for Chemical Engineers*", 2nd edition, McGraw Hill.

MacQUEEN, J., (1967), "Some Methods for Classification and Analysis of Multivariate Observations", Proceeding of the Fifth Berkeley symposium on Mathematics, Statistics and, Probability", eds., L. M. LeCam and J. Neyman Berkeley: University of California Press, 281pp.

MAH, R.S.H.; MICHAELSON, M.; SARGENT, R.W.H., (1962), "*Dynamic Behaviour of Multicomponent Multi-Stage Systems – Numerical Methods for Solution*", **Chemical Engineering Science**, n17, pp619-624.

MASTERS, T., (1995), "*Advanced Algorithms for Neural Networks: a C++ Sourcebook*", NY: John Wiley and Sons, ISBN 0-471-10588-0.

MAVROUVONIOTIS, M.; STEPHANOPOULOS, G., (1987), "*Reasoning with Orders of Magnitude and Approximate Relations*", **Proceedings of AAAI-87**, Morgan Kaufman Publishing, INC., Los Altos, CA.

McCULLAGH, W.S.; PITTS, W. ,(1943), "*A Logical Calculus Of The Ideas Imminent In Nervous Activity*", **Bulletin of Mathematical Biophysics,** n5, pp115-133.

McCULLOCH, P.; NELDER,J.A., (1989), "*Generalized Linear Models*", 2nd edition, London: Chapman and Hall.

MEINGUEST, J., (1979), "*Multivariate interpolation at Arbitrary Points Made Simple*", **Journal of Applied Mathematics and Physics**, 30, pp292-304.

MICCHELLI, C.A. (1986), " *Interpolation of Scattered Data: Distance Matrices and Conditionally Positive Definite Functions*", **Constructive Approximations** 2, pp11-22.

MILNE, R., (1987), "*Strategies for Diagnosis*", **IEEE Transactions Systems., Man, and Cybernetics, SMC-17**.

MINSKY, L.M.; PAPERT, S.A., (1990), "*Perceptrons*", expanded edition, MIT Press.

MOODY, J.E.; DARKEN, C.J., (1989), "*Fast Learning in Networks of Locality-Tuned Processing Units*", **Neural Computation** 1(2), pp281-294.

MOORE, B., (1988), " *ART 1 and Pattern Clustering*", in Touretzky, D., Hinton, G and Sejnowsky, T., eds, **Proceedings of the 1988 Connectionist Models Summer School**, pp174-185, San Mateo, CA: Morgan Kauffmann.

MORRISON, D. L., (1994), "*Fault Diagnosis and Computer Integrated Manufactoring Systems*", **IEEE Transactions on engineering Management,** v41, Iss:1, pp69-83.

MYLARASWAMY, D.; VENKATASUBRAMANIAN, V., (1997), "*A Hybrid Framework for Large Scale Process Fault Dianosis*", **Computers and Chemical Engineering**, v21, Suppl, pp.S935-S940.

NARASIMHAN, S.; MAH, R.S.H., (1987), "*Generalized Likelihood Ratio Method for Gross Error Detection*", **AIChe, J.**, n33, pp1514-1520.

NEAL, R.M., (1993a), "*Bayesian Learning via Stochastic Dynamics*", in Giles, C.L., Hanson, S.J., and Cowan, J.D., (eds), **Advances in Neural Information Processing Systems 5**, San Mateo, Califirnia: Morgan Kauffmann, pp475-482.

NEAL, R.M., (1993b), "*Probabilistic Inference Using Markov Chain Monte Carlo Methods*", in ftp://ftp.cs.utoronto.ca/pub/radford/review.ps.z.

NEAL, R.M.,(1995), "*Bayesian Learning for Neural Networks*", PhD. Thesis, University of Toronto,

NELSON, M.C., ILLINGWORTH, W.T., (1991), "*A Practical Guide to Neural Nets*", Reading, MA: Addison-Wesley.

NERUDA, R., (1995), "*Functional Equivalence and Genetic Learning*", **In Proceedings of ICAN-NGA'95**, pp 53-56.

van NESS, S., (1987), "*Introduction to Chemical Engineering Thermodynamics*", McGraw-Hill.

NIEMELA, R.J.; KRENDELL, E.S., (1975), "*Detection of a Change in Plant Dynamics in a Man-Machine System*", **IEEE Trans Sys, Man and Cyber.**, smc-5, pp615-620.

NIGRIN, A, (1993), "*Neural Networks for Pattern Recognition*", Cambridge, MA: The MIT Press, p11.

ORR, M.J.L, (1996), "Introduction to Radial Basis Function Networks", Centre for Cognitive Science, University of Edinbourgh EH8 9LW, Scothland.

ORR, G.B.; LEEN, T.K., (1997), "*Using Curvature Information for Fast Stochastic Search*", in Mozer, M.C., Jordan, M.I., and Petsch, T., (editors), **Advances in Neural Information Processing Systems 9**, Cambridge, MA: The MIT Press, pp606-612.

OZYURT, B.; SUNOL, A. K.; ÇAMURDAN, M. C.; MOGILI, P.; HALL, L. O., (1995), "*Chemical Plant Fault Diagnosis Through a Hibrid Symbolic-Connectionist Approach and Comparison with Neural Networks*", **Computers and Chemical Engineering,** v19, iss: suppl.issue ppS753-758.

OZYURT, B.; SUNOL, A. K.; ÇAMURDAN, M. C.; MOGILI, P.; HALL, L. O., (1995), "*Chemical Plant Fault Diagnosis Through a Hibrid Symbolic-Connectionist Machine Learning Approach*", **Computers and Chemical Engineering,** v22, n1-2, pp299-321.

PATTON, R.; FRANK, P.; CLARK, R., (1989), "*Fault Diagnosis in Dynamic Systems*", Prentice-Hall.

PEDROSA, L. S., (1998), "*Controle Adaptativo de uma Coluna Piloto de Destilação em Batelada com Inferenciação de Composição Através de Redes Neurais Artificiais*", **Tese de Mestrado, UNICAMP.**

PENG, Y.; REGGIA J.A., (1987a), "*A probabilistic Causal model for diagnostic Problem Solving – Part I: Integrating Symbolic Causal Inference with numeric Probabilistic inference*", **IEEE Trans. Syst., Man Cybern., SMC-17**, pp146-162.

PENG, Y.; REGGIA J.A., (1987b), "*A probabilistic Causal model for diagnostic Problem Solving – Part II: Diagnostic Strategy*", **IEEE Trans., Syst., Man, Cybern., SMC-17**, pp146-162.

PERRY'S CHEMICAL HANDBOOK,(1984), 6th edition, McGraw-Hill.

PLATT, J.C., (1991), "*Learning by Combining Memorization and Gradient Descent*", in Advances in Neural Information Processing Systems3, Lippmann, R.P., Moody, J.E. (editors), Morgan Kauffman, San Mateo, CA.

PONTON, J.W.; KLEMES, J., (1993), "*Alternatives to Neural Networks for Inferential Measurement*", **Computers and Chemical Engineering**, v17, n10, pp991-1000.

POGGIO, T.; GIROSI, F., (1988), "*A Theory of Networks for Approximation and Learning*", A.I. Memo 1140, CBIP Paper n31, Massachusetts Institute of Technology.

POGGIO, T.; GIROSI, F., (1989), "*Networks and the Best Approximation Property*", A.I.Memo 1161, CBIP Paper n15, Massachussetts Institute of Technology.

POGGIO, T.; GIROSI, F., (1990a), "*Networks for Approximation and Learning*", **Proceedings of the IEEE**, v78, n9, pp1481-1497.

POGGIO, T.; GIROSI, F., (1990b), "*Regularization Algorithms for Learning that are Equivalent to Multilayer Networks*", **Science**, v247, pp978-982.

POIANI, L.M., (1994), "*Dinâmica e Controle de Colunas de Destilação – Aplicação a Sistemas de Elevada Pureza*", **Tese de Doutorado**, DESQ-UNICAMP.

POWEL, M.J.D., (1987), " *Radial Basis Functions for Multivariable Interpolation: a Review*", In J.C.Mason and M. G. Cox (Eds.), **Algorithms for Approximation**, pp143-167. Oxford: Claredon Press.

PRAUSNITZ, J.; ANDERSON, T.; GRENS, E.; ECKERT, C.; HSIEH, R.; O'CONNELL,J., (1980), "*Computer Calculations for Multicomponent Vapor-liquid and Liquid-Liquid Equilibria*", Prentice-Hall.

PRAUSNITZ, J., (1986), "*Molecular Thermodynamics of Fluid Phase Equilibria*", Prentice Hall.

PRECHELT, L., (1994), "*PROBEN1- A set of Neural Network Benchmark problems and benchmarking Rules*", Technical Report 21/94, Universitat Karlsruhe, 76128 Karlsruhe, Germany.

PRESS, W.H.; TEUKOLSKY, A.S.; VETTERLING, W.T.;FLANNERY, B.P.,(1992), "*Numerical Recipes in C: The Art of Scientific Computing*", 2nd edition Cambridge University Press.

QUANTRILLE, T.E.; LIU, Y.A.,(1991), "*Artificial Intelligence in Chemical Engineering*", Academic Press, San Diego, CA.

RAFTERY, A.E., (1995), "*Bayesian Model Selection in Social Research*", in Marsden, P.V. (editor), Sociological Methodology 1995, Cambridge, MA: Blackwell.

RAMAMURTI, V.; GOSH, J., (1996), "*Flexible Modular Architecture for Changing Enviroments*", TR-DECE: University of Texas (viswa.gosh@pine.ecu.utexas.edu).

RAMESH, T. S.; SHUM, S. K. and DAVIS J. F., (1988), "*A Structured Framework for Efficient Problem Solving in Diagnostic Expert Systems*" , **Computer and Chemical Engineering**, v12, pp891-902.

RASMUSSEN, J., (1986), "*Information Processing and Human-Machine Interaction*", North Holland, New York

RASMUSSEN, J.; JENSEN, A., (1974), "*Mental Procedures in Real Life Tasks: A Case Study of Electronic Troubleshooting*", **Ergonomics**, v17, n3, pp293-307.

REDDY, V.N.; MAVROVOUNIOTIS, M.L., (1998), "*An Imput-Training Neural Network Approach for Gross Error Detection and Sensor Replacement*", **Trans. Ichem**, v76, part A, pp478-489.

REEVES, C.R., ed., (1993), "*Modern Heuristic Techniques for Combinatorial Problems*", NY: Wiley.

REID, R.C; PRAUSNITZ, J.M.; POLING, S.I., (1987), "*The Properties of Gases and Liquids*", fourth edition, McGraw-Hill.

RICH, S. H.; VENKATASUBRAMANIAN, V., (1985), "*Model-based Reasoning in Diagnostic Expert Systems for Chemical Process Plants*" , **Computer and Chemical Engineering**, v9,pp285-293.

RICH, S. H.; VENKATASUBRAMANIAN, V., (1987a), "*Causality-based Failure-driven Learning in Diagnostic Expert Systems*" , **AIChe. Journal**, v35, pp943-950.

RICH, S. H.; VENKATASUBRAMANIAN, V., (1987b), "*Model-based Reasoning in Diagnostic Expert Systems for Chemical Process Plants*" , **Computers and Chemical Engineering**, v11,pp111-122.

RIEDMILLER, M., (1994), "*Advanced Supervised Learning in Multi-layer Perceptrons - From Backpropagation to Adaptive Learning Algorithms*",

RIEDMILLER, M.; BAUM, H., (1993), "*A Direct Adaptive Method for Faster Backpropagation Learning: The Rprop Algorithm*", **In IEEE International Conference on Neural Networks (ICNN 93).**

ROBITAILLE, B.; MARCOS, B.; VEILLETTE, M.; PAYRE, G., (1995), "*Modified Quasi-Newton Methods for Training Neural Networks*" , **Computers and Chemical Engineering,** v20, n9,pp1133-1140.

ROELE, M.; WARWICK, K., (1993), "*Dynamics Control of Chemical Reactors Distillation Columns and Batch Process (Dycord+'92)*". **Selected Papers from IFAC Symposium,** pp255-260, Pergamon Press, Oxford, UK.

ROJAS-GUZMAN, C.; KRAMER, M. A., (1993), "*Comparison of Belief Networks and Rule-Based Expert Systems for Fault Diagnosis of Chemical Processes*", **Engineering Applications of Artificial Intelligence,** v6, Iss: e, pp191-202.

ROSE, L.M.,(1985), "*Distillation Design in Practice*", Elsevier.

ROSENBLAT, R., (1959), "*Principles of Neurodynamics*", New York, Spartan Books.

ROY, A.; GOVIL, S.; MIRANDA, R., (1997), "*A Neural-Network Learning Theory and a Polynomial Time RBF Algorithm*", **IEEE Transactions on Neural Networks,** v8, n6, pp1301-1313.

RUMELHART, D.E.; HINTON, G.E.; WILLIAMS, R.J., (1986), "*Learning Internal Representations by Error Propagation*", in D. E. Rumelhart, J. L. McCleland, and the PDP Research Group (Editors.), **Parallel Distributed Processing: Explorations in the Microstruture of Cognition**, Vol1: Foundations, pp 318-362. Cambridge, MA: MIT Press.

RUMELHART, D. E.; McCLELLAND, J.L. (Editors),(1986), "*Parallel Distributed Processing: Explorations in the Microstructure of Cognition*", MIT PRESS, Cambridge, Mass. (1986).

RUSSEL, G.; FAUSETT, L.V., (1996), "*Comparison of function Approximation with Sigmoid and Radial Basis Function Networks*", **SPIE Proceedings**, v2760, pp61-72, Orlando, FL,USA.

SAMPAIO, R. P., (1994), "*Modelagem e Simulação Dinâmica de Colunas de Destilação a partir de Modelos de Ordem Reduzida*", **Tese de Mestrado**, DESQ-UNICAMP.

SANDLER, S.I., (1977), "*Chemical and Engineering Thermodynamics*", John Wiley and Sons.

SANTEN, A.; KOOT, G.L.M., (1997), "*Statistical Data Analysis of a Chemical Plant*", **Computers and Chemical Engineering**, v21, Suppl., ppS1123-S1129.

SARKAR, D., (1996), "*Empirical Stimation of Generalization ability of Neural Networks*", SPIE Proceedings, v2760, pp54-60, Orlando,FL,USA.

SARLE, W.S. (1994), "*Neural Networks and Statistical Models*", in SAS Institute Inc., Proceedings of the Nineteenth Annual SAS Users Group International Conference, Cary, NC: SAS Institute Inc., pp1538-1550.

SARLE, W.S., (1995), "Stopped Training and Other Remedies for Overfitting", in Proc. of the 27th Symposium on the Interface.

SARLE, W.S., (editor), (1997), "*Neural Network FAQ*", **Periodic Posting to the Usenet Newsgroup:** http://www.comp.ai.neural-net".

SAVKOVIC-STEFANOVIC, J., (1995), "*Qualitative Modelling and Simulation of a Complete Chemical Plant Behaviour*", **Acta Chimica Slovenica,** v42, Issue:1, pp63-68.

SCHIÖLER, H.; HARTMAN, U., (1992), "*Mapping Neural Network Derived from the Parzen Window Estimator*", **Neural Networks**, 5, pp903-909.

SEBORG.,D.E.; EDGAR, T.F.; MELLICHAMP, D.A.,(1989), "*Process Dynamics and Control*", John Wiley and Sons.

SELIGER, S.O.; FRANK, P.M., (1991), "*Fault Diagnosis in Dynamic Systems Using Analytical and Knowledge-Based Redundancy – A survey and Some New Results*", **Automatica**, v22, n1, pp49-474.

SHAELWITZ, J.A; LAPP, S.A., POWERS, G.J., (1977), "*Fault Tree Analysis of Sequential Systems*", **Ind. Eng. Chem. Process Dev**., n16, v4, pp524-549.

SHAFAGHI, A.; ANDROW, P.W.; LEES, F.P., (1984), "*Fault Tree Synthesis Based on Control Loop Structure*", **Chemical Engineering Research and Development**, n62, pp101-110.

SHAO, J., (1993), "*Linear Model Selection by Cross-Validation*", **Journal of the American Statistical Association**, 88, pp486-494.

SHAO, R.; MARTIN, E.B.; ZHANG, J.; MORRIS, A.J., (1997), "*Confidence Bounds for Neural Network Representations*", **Computers and Chemical Engineering**, v21, Suppl., ppS1173-1178.

SHIOZAKI, J.; MATSUYAMA, H.; O'SHIMA, E.; IRI, M., (1985), "*An improved Algorithm for Diagnosis of System Failures in the Chemical Process*", **Computers and Chemical Engineering**, n9, pp285-297.

SIMON, H.A., (1977), "*Models of Discovery*", Reidel Publishing Company, Boston.

SMITH, B.D., (1963), "*Design of Equilibrium Stage Process*", McGraw-Hill, New York.

SONTAG, E.D., (1992), "*Feedback Stabilization Using Two-hidden-layer Nets*", **IEEE Transactions on Neural Networks**, 3,981-990.

SORSA, T.; KOIVO, H. N., (1993), "*Application of Neural Networks in Process Fault Diagnosis*" , **Automatica,** v29, Iss;4, pp843-849.

SPECHT, D.F., (1990), "*Probabilistic Neural Networks*", **Neural Networks**, 3, pp110-118.

SUNGGU, L.; SHIN, K.G., (1994), "*Probabilistic Diagnosis of Multiprocessor Systems*", **ACM Computing Surveys**, v26, n1, pp121-139.

SRINIVASAN, R.; DIMITRIADIS, V.D.; SHAH, N.; VENKATASUBRAMANIAN, V., (1997), "*Integrating Knowledge-Based and Mathematical Programming Approachs for Process Safety Verification*", **Computers and Chemical Engineering**, v21, Suppl., ppS905-S910.

TEIXEIRA, A.C. (1997), "*Reconciliação de Dados de Processos e Detecção de Erros Grosseiros em Sistemas com Restrições Não-Lineares*", **Tese de Mestrado, UNICAMP.**

THIMM G.; FIESLER, E., (1997), "*High-order and Multilayer Perceptron Initialization*", **IEEE Transactions on Neural Networks**, v8, n2, pp349-359.

TIBSHIRANI, R., (1996), "*A Comparison of Some Error Estimates for Neural Network Models*", **Neural Computation**, 8, pp152-163.

TIKHONOV, A.N.; ARSENIN, V.I., (1997), "*Solutions of Ill-posed Problems*", W.H. Winston, Washington, DC.

TSAI, CHI-SHANG; CHANG, CHUEI-TIN, (1995), "*Dynamic Process Diagnosis via Integrated Neural Networks*", **Computers and Chemical Engineering**, v19, Suppl., pp.S747-s752.

TSAI, CHI-SHANG; CHANG, CHUEI-TIN; CHEN, CHAO-SHIOU, (1996), "*Fault Diagnosis in Batch and Semi-Batch Processes Using Artificial Neural Networks*", **Chemical Engineering Communications**, v143, pp39-71.

TSONOPOULOS, C., (1974), AIChe Journal, n20, pp263-268.

TZAFESTAS, S. G.; DALIANIS, P. J., (1995), "*Artificial Neural Networks in the Fault Diagnosis of Technological Systems: a Case Study in Chemical Engineering Process*" , **Elektronnoe Modelirovanie**, v17, Iss: 6, pp21-27.

ULERICH, N.H; POWERS, G.A., (1987), "*Online Hazard Aversion and Fault Detection: Multiple Loop Control Example*", **AIChe Annual Meeting**, New York.

UMEDA, T.; KURIYAMA, T.; O'SHIMA, E.; MATSUYAMA, H., (1980), "*A Graphical Approach to Cause and Effect Analisys of Chemical Processing systems*", **Chemical Engineering Science**, n35, pp2379-2388.

UPADHYAYA, B.R., (1985), "*Sensor Failure Detection and Estimation*", **Nuclear Safety**, v26, pp32-43.

VAIDYANATHAN, R.; VENKATASUBRAMANIAN, V, (1992a), "*Representing and Diagnosing Dynamics Process Data Using Neural networks*" , **Engineering Applications of Artificial Intelligence**, v5, Iss:1, pp11-21.

VAIDYANATHAN, R.; VENKATASUBRMANIAN, V., (1992b), "*A Knowledge-Based Framework for Automating HAZOP Analysis*" , **AIChe Meeting**.

VENKATASUBRAMANIAN, V., (1987), "*Model-Based Reasoning in Diagnostic Expert Systems for Chemical Process Plants*", **Computer and Chemical Engineering** v11,n2,pp111-122.

VENKATASUBRAMANIAN, V.; CHAN, K., (1989), "*A Neural Network Methodology for Process Fault Diagnosis*" , **AIChe. Journal**, v35, n12, pp1993-2002.

VENKATASUBRAMANIAN, V, DHURJATI, P. S., (1987), "*An Object-oriented Knowledge Base Representation for the Expert System FALCON*", Foundations of

Computer Aided Process Operations, Reklaitis and Spriggs, eds., CACHE/Elsevier,701-670.427 F825

VENKATASUBRAMANIAN, V.; RICH, S. H., (1988), "*An Object-oriented Two-tier Archictecture for Integrating Compiled and Deep-Level Knowledge for Process Diagnosis*" , **Computer and Chemical Engineering**, v12, pp903-921.

VENKATASUBRMANIAN, V.; VAIDYANATHAN, R. and YAMAMOTO, Y.,(1990), "*Process Fault Diagnosis Using Neural Networks: I -Steady State Process*" , **Computers and Chemical Engineering**, v14,pp699-712.

VENKATESWARLU, Ch.; GANGIAH, K.; BHAGAVANTHA RAO, M., (1992), "*Two-Level Methods for Incipient Fault Diagnosis in Nonlinear Chemical Processes*", **Computers and Chemical Engineering(Oxford)**, v16, pp463-476.

VORA, N.; TAMBE, S.S.; KULKARNI, B.D., (1997), "*Counterpropagation Neural Networks for Fault Detection and Diagnosis*", **Computers and Chemical Engineering**, v21, n2, pp177-185.

WASBURN, E.W., (editor), (1928), "*International Critical Tables of Numerical Data Physics, Chemistry and Technology*", v3, McGraw-Hill.

WATANABE, K., (1985), "*General Two-Stage Bias Correction Filter and Predictor for Linear Discrete-Time Systems*", **Control Theory and Advanced Technology**, v1, pp87-101.

WATANABE, K.; HIMMELBLAU, D. M., (1984), "*Incipient Fault Diagnosis of Nonlinear Process with Multiple Causes of Faults*" , **Chemical Engineering Science**, v39, n3, pp491-508.

WATANABE, K.; HIROTA, S. and HOU, L., (1994), "*Diagnosis of Multiple Simultaneous Fault via Hierarchical Artificial Neural Networks*" , **AIChe. Journal**, v40, n5, pp839-848.

WATANABE, K.; HOU, L., (1992), "*An Optimal Neural Network for Diagnosing Multiples Faults in Chemical Process*" , **Proceedings of the 1992 International Conference on Industrial Electronics, Control, Instrumentation, and Automation. Power Electronics and Motion Control, (No.92CH3137-7)**, pp1068-1073, v2., IEEE, New York, USA.

WATANABE, K.; MATSUURA, I.; ABE, M., KUBOTA, M.; HIMMELBLAU, D. M., (1989), "*Incipient Fault Diagnosis of Chemical Process via Neural Networks*" , **AIChe. Journal**, v35, n11, pp1803-1812.

WEN, F.; CHANG, C.S., (1999), " *A New Method for Diagnostic Problem Solving Based on a Fuzzy Abductive Inference Model and the Tabu Search Approach*", **Artificial Intelligence in Engineering**, n13, pp83-90.

WENNERSTEN, R.; NARFELDT, R.; GRÄNFORS, A.; SJÖKVIST, S., (1995), "P*rocess Modeling in Fault Diagnosis*" , **Computers and Chemical Engineering**, v20, Suppl., pp. S665-670

WERBOS, P. J., (1974), "*Beyond Regression: New Tools for Prediction and Analysis in the Behavioural Sciences*", PhD Thesis in Applied Mathematics, Harvard University (1974).

WHITE, H., (1990), "*Conecticionist Nonparametric Estimation via Empirical Risk Minimization*", **IEEE Transactions on Information Theory**, v41, pp677-678.

WIDROW, B, (1962), "*Generalization and Information Storage in Networks of Adaline 'Neurons'*", in Self-Organizing Systems 1962, M. Yovits, G. Jacobi, and G. Goldstein, Editors. Washington, DC: Spartan Books, pp435-461.

WIDROW, B.; LEHR, M.A.,(1990), "*30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation*", **Proc. IEEE,** vol78, n9, pp1415-1442.

WILLSKY, A.S., (1976), "*A Survey of Design Methods for Failure Detection in Dynamic Systems*", **Automatica**, v12, pp601-611.

WILLSKY, A.S.; JONES, H.L., (1990), "*A Generalized Likehood Ratio Approach to State Estimation in Linear Systems Subject to Abrupt Changes*", **Proceedings of IEEE Conf. Decision and Control**, pp846-856.

WON, J.K.; MODARRES, M., (1998), "*Improved Bayesian Method for Diagnosing Equipment Partial Failures in Process Plants*", **Computers and Chemical Engineering**, v22, n10, pp 1483-1502.

ZHAO, J.; CHEN, B.; SHEN, J., (1997), "*A Hybrid ANN_ES System for Dynamic Fault Diagnosis of Hydrocracking Process*", **Computers and Chemical Engineering**, Suppl, ppS929-S933.

ZHAO, J.; CHEN, B.; SHEN, J., (1998), "*Multidimensional non-Ortogonal Wavelet-Sigmoid Basis Function Neural Network for Dynamic Process Fault Diagnosis*", **Computers and Chemical Engineering**, v23, pp83-92.

APÊNDICE A: TÉCNICAS ADAPTATIVAS GLOBAIS

Gradiente Descendente

O gradiente de uma função a um determinado valor de peso é examinada e usada para determinar a mudança nos valores dos pesos que reduza o erro.

Dada a função $\mathbf{E} = \mathbf{f}(\mathbf{w}_{kj})$[1], o gradiente de $\mathbf{f}(\ )$ é o gradiente da tangente da função num ponto conforme o mostrado na Figura (A.1), tal que

$$\textbf{gradiente de } \mathbf{f}(\ ) \textbf{ em } \mathbf{P} = \frac{\Delta \mathbf{E}}{\Delta \mathbf{w}_{kj}}. \tag{A.1}$$

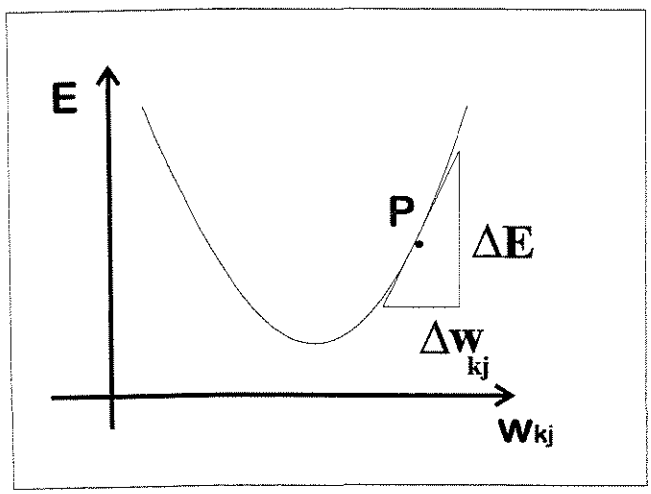


Figura (A.1): gradiente sobre a curva.

Se $\Delta \mathbf{w}_{kj}$ é suficientemente pequeno então $\Delta \mathbf{E} \approx \delta \mathbf{E}$, onde $\delta \mathbf{E}$ é a mudança resultante em $\mathbf{E}$ quando $\mathbf{w}_{kj}$ é alterado por $\Delta \mathbf{w}_{kj}$, levando a

$$\delta \mathbf{E} \approx \Delta \mathbf{E} = \frac{\Delta \mathbf{E}}{\Delta \mathbf{w}_{kj}} \Delta \mathbf{w}_{kj}, \tag{A.2}$$

dando,

$$\delta \mathbf{E} \approx \frac{\partial \mathbf{E}}{\partial \mathbf{w}_{kj}} \Delta \mathbf{w}_{kj} \tag{A.3}$$

o que produz

[1] Aqui, o símbolo do erro adotado é E porque trata-se de uma forma geral de definição e não do erro utilizado para o backpropagation simples ou para as redes com RBF. Ver capítulo 5, no desenvolvimento dos modelos citados.

$$\Delta \mathbf{w}_{kj} = -\alpha \frac{\partial \mathbf{E}}{\partial \mathbf{w}_{kj}}, \tag{A.4}$$

(notar que o peso é atualizado na direção oposta do gradiente)
onde α (taxa de aprendizagem) é maior que zero mas pequeno o suficiente para assegurar que $\partial \mathbf{E} \approx \Delta \mathbf{E}$, dando

$$\delta \mathbf{E} \approx -\alpha \left(\frac{\partial \mathbf{E}}{\partial \mathbf{w}_{kj}} \right)^2, \tag{A.5}$$

a qual dará $\delta \mathbf{E} < \mathbf{0}$ tal que alterando-se $\mathbf{w}_{kj}$ por $\Delta \mathbf{w}_{kj}$, move-se em direção ao mínimo.

A escolha de uma boa taxa de aprendizagem depende da forma da função erro, a qual obviamente muda com a própria aprendizagem.

Gradiente Descendente Para Treinamento de Redes Neurais

Se o gradiente descendente for ser usado no treinamento de uma rede, então o erro deve ser uma função contínua e diferenciável em relação aos pesos tal como a função sigmoidal. Reescrevendo o erro dado pela equação (5.6) de outra forma teremos

$$\mathbf{E}_p = \frac{\mathbf{1}}{\mathbf{2}} (\mathbf{t}_k - \mathbf{y}_k)^2$$

onde, $\mathbf{t}$ é o alvo desejado e $\mathbf{y}$ é o valor calculado. Mas $\mathbf{y}_k = \mathbf{f}(\mathbf{w}_k^t \mathbf{x})$, assim

$$\mathbf{E}_p = \frac{\mathbf{1}}{\mathbf{2}} \left[\mathbf{t}_k - \mathbf{f}(\mathbf{w}_k^t \mathbf{x}) \right]^2 \tag{A.6}$$

para a qual o gradiente obtido é

$$\nabla \mathbf{E} = -(\mathbf{t}_k - \mathbf{y}_k) \mathbf{f}'(\mathbf{w}_k^t \mathbf{x}) \mathbf{x} \tag{A.7}$$

usando a equação (A.4) para obter a direção descendente da curva do erro, a mudança no conjunto de pesos será

$$\Delta \mathbf{w}_k = -\alpha \nabla \mathbf{E} \tag{A.8}$$

e agora usando a equação (A.7), tem-se

$$\Delta \mathbf{w}_k = \alpha (\mathbf{t}_k - \mathbf{y}_k) \mathbf{f}'(\mathbf{w}_k^t \mathbf{x}) \mathbf{x} \tag{A.9}$$

de tal modo que o ajuste para cada peso se torna

$$\Delta \mathbf{w}_{kj} = \alpha (\mathbf{t}_k - \mathbf{y}_k) \mathbf{f}'(\mathbf{w}_k^t \mathbf{x}) \mathbf{x}_j . \tag{A.10}$$

Referindo-se agora, à regra geral de aprendizagem, o sinal de aprendizagem pode agora ser definido como

$$\mathbf{r} \stackrel{\Delta}{=} \left[\mathbf{t}_k - \mathbf{f}(\mathbf{w}_k^t \mathbf{x})\right]\mathbf{f}'(\mathbf{w}_k^t \mathbf{x}). \tag{A.11}$$

Isto pode ser estendido parra mais que um neurônio. Dado que o alvo de saída é, agora, um vetor $\mathbf{t} \stackrel{\Delta}{=} [\mathbf{t}_1, \mathbf{t}_2, ..., \mathbf{t}_k]^T$, o erro da rede para um dado padrão é

$$\mathbf{E}_p = \frac{1}{2}\sum_{k=1}^{K}(\mathbf{t}_k - \mathbf{y}_k)^2 = \frac{1}{2}\|\mathbf{t} - \mathbf{y}\|^2. \tag{A.12}$$

Seguindo (A.8), a mudança nos pesos individuais são calculadas como

$$\Delta \mathbf{w}_{kj} = -\alpha \frac{\partial \mathbf{E}}{\partial \mathbf{w}_{kj}}, \tag{A.13}$$

onde $\mathbf{E}$ é o erro dado em (A.12), omitindo o subscrito $\mathbf{p}$. A ativação $\mathbf{s}_k$ para cada nó na camada k é definida como

$$\mathbf{s}_k = \sum_{j=1}^{J} \mathbf{w}_{kj}\mathbf{x}_j \tag{A.14}$$

fazendo a saída do neurônio ser

$$\mathbf{y}_k = \mathbf{f}(\mathbf{s}_k)$$

o termo do erro do sinal, δ do k-ésimo neurônio é definido como

$$\delta_{yk} \stackrel{\Delta}{=} -\frac{\partial \mathbf{E}}{\partial \mathbf{s}_k}. \tag{A.15}$$

O termo $\partial \mathbf{E}/\partial \mathbf{w}_{kj}$ é dependente somente do valor de $\mathbf{s}_k$ para um neurônio simples desde que o erro para a k-ésima saída é produzida somente pela saída $\mathbf{y}_k$ e conseqüentemente pelos pesos $\mathbf{w}_{kj} = \mathbf{1,2,...,J}$ para um dado valor de $\mathbf{k}$. assim, pela regra da cadeia

$$\frac{\partial \mathbf{E}}{\partial \mathbf{w}_{kj}} = \frac{\partial \mathbf{E}}{\partial \mathbf{s}_k} \cdot \frac{\partial \mathbf{s}_k}{\partial \mathbf{w}_{kj}}. \tag{A.16}$$

O segundo termo aqui é a derivada da equação (A.14), a qual desde que $\mathbf{x}_j = \mathbf{1,2,...,J}$ sejam constantes para um dado padrão de entrada, produz

$$\frac{\partial \mathbf{s}_k}{\partial \mathbf{w}_{kj}} = \mathbf{x}_j \tag{A.17}$$

Assim, usando (A.15) e (A.17), (A.16) pode ser escrita como

$$\frac{\partial \mathbf{E}}{\partial \mathbf{w}_{kj}} = -\delta_{yk}\mathbf{x}_j . \tag{A.18}$$

A mudança nos peso pode ser então calculada usando

$$\Delta \mathbf{w}_{kj} = \alpha \delta_{yk}\mathbf{x}_j . \tag{A.19}$$

Esta é uma equação geral para o ajuste dos peso com a regra delta a qual é fixada sem considerar a forma da função de ativação. Ao invés, é o valor delta, δ_{yk}, que varia de acordo com a função de ativação escolhida e deve ser calculdo para levar isto em conta.

Usando a regra da cadeia na equação (A.15), tem-se

$$\delta_{yk} \overset{\Delta}{=} -\frac{\partial \mathbf{E}}{\partial \mathbf{y}_k}\cdot\frac{\partial \mathbf{y}_k}{\partial \mathbf{s}_k}, \tag{A.20}$$

onde o segundo termo pode ser visto como a derivada da função de ativação

$$\mathbf{f}'_k(\mathbf{s}_k) \overset{\Delta}{=} \frac{\partial \mathbf{y}_k}{\partial \mathbf{s}_k} \tag{A.21}$$

e também

$$\frac{\partial \mathbf{E}}{\partial \mathbf{y}_k} = -(\mathbf{t}_k - \mathbf{y}_k). \tag{A.22}$$

Conseqüentemente, (A.20) pode ser escrita como

$$\delta_{yk} = (\mathbf{t}_k - \mathbf{y}_k).\mathbf{f}'(\mathbf{s}_k) . \tag{A.23}$$

Assim, a fórmula para a mudança de pesos fica

$$\Delta \mathbf{w}_{kj} = \alpha(\mathbf{t}_k - \mathbf{y}_k)\mathbf{f}'(\mathbf{s}_k)\mathbf{x}_j . \tag{A.24}$$

Esta fórmula pode ser usada para qualquer forma de função de ativação diferenciável **f(s)**.

Apesar de a convergência para um mínimo local poder ser provada sob certas circunstâncias, não existe garantia que o algoritmo encontre um mínimo global para a função erro.

Um outro problema com o gradiente descendente é a influência da derivada parcial no valor do peso. Se a função erro é pouco inclinada, a derivada é bastante pequena, resultando em um pequeno valor para a atualização dos pesos. Por outro lado, se a função erro apresenta vales profundos, bastante inclinada, grandes valores de derivada aparecem e levam a grandes correções nos pesos, podendo levar o valor da função objetivo para uma região completamente diferente no espaço. Por esta razão, para se tornar o termo de aprendizado mais estável, adiciona-se um termo de momentum:

$$\Delta \mathbf{w}_{kj}(t) = -\alpha \frac{\partial \mathbf{E}}{\partial \mathbf{w}_{kj}}(t) + \mu \Delta \mathbf{w}_{kj}(t-1). \qquad (A.25)^2$$

O parâmetro de momentum balanceia a influência dos pesos anteriores nos pesos atuais. Apesar de esta técnica funcionar bem para muitos aprendizados, não é uma técnica geral para se ganhar estabilidade ou melhorar a convergência.

O método do passo descendente[3] e o do gradiente conjugado são dois exemplos de técnicas de adaptação global.

A. passo descendente

Enquanto a técnica do gradiente descendente usada nos algoritmos de propagação reversa padrões atualiza os pesos através de um valor constante da taxa de aprendizagem, o passo descendente tenta atingir um peso ótimo encontrando uma taxa de aprendizagem a cada iteração .

A determinação deste parâmetro ótimo pode ser conseguida com a otimização de um problema unidimensional conhecido como linha de busca. No caso mais simples, uma pequena taxa de aprendizagem inicial é usada, a qual é iterativamente aumentada até que a função erro não apresente mais mudanças significativas.

Infelizmente, para cada iteração a avaliação da função erro **E** é requerida, o que significa aumento de tempo de processamento. Em geral, métodos mais elaborados para linha de busca devem ser usados, tais como o método da posição falsa.

Aplicando-se o método do passo descendente, pode-se mostrar que dois pesos atualizados sucessivamente são perpendiculares.

Assuma que uma taxa de aprendizagem tenha sido encontrada e que produza um ótimo valor para a atualização do peso, o que signiofica que $\frac{\partial \mathbf{E}(\mathbf{w}(t+1)) = \mathbf{0}}{\partial \alpha}$. Então,

$$\begin{aligned} \frac{\partial \mathbf{E}(\mathbf{w}(t+1))}{\partial \alpha} &= \frac{\partial \mathbf{E}(\mathbf{w}(t+1))}{\partial \mathbf{w}(t+1)} \frac{\partial (\mathbf{w}(t) + \quad * \mathbf{d}(t))}{\partial} \\ &= \nabla \mathbf{E}(t+1)\mathbf{d}(t) \\ &= \mathbf{0} \end{aligned} \qquad (A.26)$$

[2] Esta equação é apenas uma outra forma de se mostrar a equação (A.24), para chegar a ela basta substituir a equação (A.18) em (A.19) e acrescentar o termo de momentum.

[3] Na verdade, o termo passo descendente não é correto pois em inglês o termo é 'Steepest Descent', mas por motivos associativos (associação com o passo de aprendizagem), será adotado neste trabalho esta tradução livre.

Isto significa que o novo gradiente $\nabla \mathbf{E}(\mathbf{t}+\mathbf{1})$ o qual determina a nova direção $\mathbf{d}(\mathbf{t}+\mathbf{1})$, e a direção de busca antiga $\mathbf{d}(\mathbf{t})$ são perpendiculares. Esta relação é usada para melhorar o desempenho do procedimento do passo descendente.

B. método do gradiente conjugado

Encontrar uma taxa de aprendizagem ótima é uma tarefa iterativa custosa, de modo que o que se quer é minimizar este esforço. De acordo com o exposto na equação (A.26), deve-se ter também a seguinte condição

$$\mathbf{d}(\mathbf{t})\nabla \mathbf{E}(\mathbf{t}+\mathbf{2}) = \mathbf{0} \tag{A.27}$$

esta condição é válida se

$$\mathbf{d}(\mathbf{t})\mathbf{H}\mathbf{d}(\mathbf{t}+\mathbf{1}) = \mathbf{0} \tag{A.28}$$

onde $\mathbf{H}$ denota a matriz de Hessian, contendo as derivadas de segunda ordem dos pesos. Dois vetores preenchendo a condição exposta na equação (A.28) são chamados de conjugados.

Para determinar a nova direção de busca $\mathbf{d}(\mathbf{t}+\mathbf{1})$ que preenche a condição (A.28), tem-se

$$\mathbf{d}(\mathbf{t}+\mathbf{1}) := -\nabla \mathbf{E}(\mathbf{t}+\mathbf{1}) + \beta * \mathbf{d}(\mathbf{t}) \tag{A.29}$$

Isto significa que a nova direção de busca é uma combinação de ambas as direções indicadas pelo gradiente e pela direção de busca prévia.

O parâmetro β é computado, por exemplo, de acordo com a regra de Polak-Ribiere:

$$\beta = \frac{(\nabla \mathbf{E}(\mathbf{t}+\mathbf{1}) - \nabla \mathbf{E}(\mathbf{t}))\nabla \mathbf{E}(\mathbf{t}+\mathbf{1})}{(\nabla \mathbf{E}(\mathbf{t}))^2} \tag{A.30}$$

Assim, no procedimento de passo descendente, uma técnica de linha busca precisa ser aplicada para encontrar uma taxa ótima de aprendizagem que minimize o erro ao longo da nova direção de busca $\mathbf{d}(\mathbf{t}+\mathbf{1})$.

APÊNDICE B: TÉCNICAS ADAPTATIVAS LOCAIS[1]

B1. Regra Delta-Bar-Delta

Para contornar os problemas da atualização dos pesos do backpropagation padrão, JACOBS(1988) propôs taxas de aprendizagem específicas, uma vez que a função erro pode ter uma forma diferente com respeito a cada um dos pesos na rede. Por causa disto, introduziu uma segunda lei de aprendizagem, a qual determina a evolução de uma taxa de aprendizagem de acordo com a estimação local da forma da função erro.

Esta estimação é baseada no comportamento da derivada parcial durante o dois passos sucessivos dos pesos. Se a derivada tem o mesmo sinal, a taxa de aprendizagem é linearmente acrescida por uma pequena constante para acelerar a aprendizagem em regiões menos inclinadas da função erro. Por outro lado, uma mudança no sinal das duas derivadas indica que o procedimento ultrapassou um mínimo local; a atualização dos pesos foi muito grande. Como conseqüência, a taxa de aprendizagem é exponencialmente decrescida multiplicando-a por um fator de decrescimento menor que a unidade:

$$\alpha_{kjj}^{(t)} = \begin{cases} \kappa + \alpha_{kj}^{(t-1)}, & \textbf{se} \dfrac{\partial E^{(t-1)}}{\partial w_{kj}} * \dfrac{\partial E^{(t)}}{\partial w_{kj}} > 0 \\ \eta^{-} * \alpha_{kj}^{(t-1)}, & \textbf{se} \dfrac{\partial E^{(t-1)}}{\partial w_{kj}} * \dfrac{\partial E^{(t)}}{\partial w_{kj}} < 0 \\ \alpha_{kj}^{(t-1)} & \textbf{de outra forma} \end{cases} \qquad \text{(B.1)}$$

com $0 < \eta^{-} < 1$.

O peso auto-atualizado, terceira linha da equação (B.1) é o mesmo que o do backpropagation padrão, exceto que a taxa de aprendizagem global α é substituída por uma taxa específica por peso, a taxa de aprendizagem dinâmica, $\alpha_{kj}^{(t)}$, assim a equação descrita no Apêndice A se torna:

$$\Delta w_{kj}(t) = -\alpha_{kj}(t)\frac{\partial E}{\partial w_{kj}}(t) + \mu \Delta w_{kj}(t-1) \qquad \text{(B.2)}$$

Segundo JACOBS, o Delta-Bar-Delta converge mais rápido que o bakpropagation e é mais robusto com respeito à escolha dos parâmetros.

B2. SuperSAB

SuperSAB é um algoritmo desenvolvido por TOLLENAERE(1990) e é também baseado na idéia do sinal do gradiente, como descrito no item anterior.

---

[1] RIEDMILLER(1994b)

A mudança básica é aumentar a taxa de aprendizagem exponencialmente ao invés de linearmente, como o Delta-Bar-Delta. Isto é feito levando-se em conta uma larga faixa de taxas de aprendizagem.

$$\alpha_{kj}^{(t)} = \begin{cases} \eta^{+} * \alpha_{kj}^{(t-1)}, & \text{se} \dfrac{\partial E^{(t-1)}}{\partial w_{kj}} * \dfrac{\partial E^{(t)}}{\partial w_{kj}} > 0 \\ \eta^{-} * \alpha_{kj}^{(t-1)}, & \text{se} \dfrac{\partial E^{(t-1)}}{\partial w_{kj}} * \dfrac{\partial E^{(t)}}{\partial w_{kj}} < 0 \\ \alpha_{kj}^{(t-1)} & \textbf{de outra forma} \end{cases} \qquad \text{(B.3)}$$

com $0 < \eta^{-} < 1 < \eta^{+}$.

Além do mais, em caso de uma mudança no sinal de duas derivadas sucessivas, o peso do passo é reveretido.

Este algoritmo se mostra mais rápido que o backpropagation simples. Um possível problema do SuperSAB é o grande número de parâmetros que necessita ser determinado a fim de conseguir alcançar boa convergência, a saber, a taxa inicial de aprendizagem, o fator do momentum e o fator de acréscimo/decréscimo.

Outro inconveniente, inerente a todo algoritmo de adaptação de taxas de aprendizagem, é a influência remanescente do tamanho da derivada parcial no peso:

$$\Delta w_{kj}(t) := -\alpha_{kj}(t)\frac{\partial E}{\partial w_{kj}}(t) + \mu \Delta w_{kj}(t-1)$$

A despeito do cuidado na adaptação da taxa de aprendizagem, a derivada em si pode ter uma imprevisível influência no tamanho do peso. Por exemplo, considere a situação, onde uma função erro muito suave leva a um aumento permanente na taxa de aprendizagem. Apesar de a taxa crescer, a atualização do peso permanece pequena, devido à pequena derivada parcial. Quando a região de decaimento íngreme é alcançada, provavelmente indicando a presença de um mínimo, a grande derivada resultante leva o peso para uma região longe do valor do peso anterior.

B3. QUICKPROP

O backpropagation é um algoritmo que calcula as derivadas parciais primeiras do erro global com respeito a cada peso. Dada esta informação pode-se aplicar o método de gradiente descendente no espaço dos pesos. Se se toma passos infinitesimais garante-se que um mínimo local será alcançado, e em muito casos, o mínimo local poderá ser um mínimo global ou quando nada, pelo menos uma boa solução.

Tomar passos infinitesimais pode ser um procedimento muito demorado sendo que desta forma, a idéia é encontrar a solução no tempo de processamento mais curto possível.

O que se quer é usar passos grandes o suficiente para acelerar a convergência sem no entanto, ultrapassar o ponto ótimo.

Conhecendo-se alguma informação a respeito das derivadas de ordem superior a um — a curvatura da função erro — pode-se, presumivelmente, ter-se melhores resultados.

Diferentes formas para aprendizagem adaptativa local são descritas por FAHLMAN(1988). Dois tipos de caminhos foram tentados. O primeiro tenta ajustar dinamicamente a taxa de aprendizagem, globalmente ou localmente para cada peso, baseado em alguma heurística computacional. O termo de momentum usado no backpropagation padrão é uma forma desta estratégia; assim, são os planos de ajuste de parâmetros descritos por PLAUT et al. (1986).

Um outro tipo de caminho faz uso explícito da derivada segunda do erro com respeito a cada peso. Dada esta informação, pode-se selecionar um novo conjunto de pesos usando o método de Newton ou alguma outra técnica de otimização mais sofisticada. Infelizmente, isto requer muito trabalho computacional para determinar a verdadeira derivada segunda de forma que será utilizada uma aproximação.

Antes de se proceder a esta aproximação, faz-se duas considerações de risco segundo o autor. A primeira assume que a função erro é uma parábola cuja concavidade está aberta para cima, e que a inclinação da curva não é afetada pela mudança nos outros pesos na rede. Estimativas da posição do mínimo para cada peso são obtidas resolvendo-se a seguinte equação para as seguintes derivadas parciais $\frac{\partial E}{\partial \alpha_{kj}}(t-1)$ e $\frac{\partial E}{\partial \alpha_{kj}}(t)$:

$$\Delta w_{kj}(t) := \frac{\partial E(t)/\partial w_{kj}}{\partial E(t-1)/\partial w_{kj} - \partial E(t)/\partial w_{kj}} \Delta w(t-1) \qquad \text{(B.4)}$$

Pode-se mostrar que esta atualização dos pesos é equivalente a uma aplicação local do método de Newton aproximado, o qual pode ser derivado a partir da expansão de primeira ordem da série de Taylor para a aproximação do erro. O objetivo é encontrar um mínimo de uma função $\mathbf{f(x)}$, e isto é feito buscando um valor tal que zere a derivada da função, $\mathbf{f'(x)=0}$. Sob a consideração de que a derivada é convexa, o método de Newton iterativamente calcula a atualização de $\mathbf{x}$ de acordo com a seguinte equação:

$$x(t+1) = x(t) + \Delta x(t) \qquad \text{(B.5)}$$

onde

$$\Delta x(t) = -\frac{f'(x(t))}{f''(x(t))} \qquad \text{(B.6)}$$

Se a informação de segunda ordem, $\mathbf{f''(x)}$ não está facilmente disponível (como é o caso dos pesos em redes neurais), uma aproximação é feita usando as derivadas de primeira ordem:

$$f''(x(t)) = \frac{f'(x(t)) - f'(x(t-1))}{x(t) - x(t-1)} = \frac{f'(x(t)) - f'(x(t-1))}{\Delta x(t-1)} \tag{B.7}$$

Substituindo-se (B.7) em (B.6):

$$\Delta x(t) = -\frac{f'(x(t))}{f'(x(t-1)) - f'(x(t-1))}\Delta x(t-1) = \frac{f'(x(t))}{f'(x(t-1)) - f'(x(t))}\Delta x(t-1) \tag{B.8}$$

e esta última equação corresponde exatamente à equação (B.4).

Apesar de a fórmula principal para a atualização dos pesos (B.4) ser direta e fácil de calcular, há algumas modificações necessárias, devido à violação das considerações acima. Primeiramente, para a regra de atualização real, se a inclinação atual é alguma coisa menor que a anterior, mas na mesma direção, o peso mudará novamente na mesma direção. O passo pode ser grande ou pequeno dependendo de quanto a inclinação diminuiu a atualização anterior. Se a inclinação atual está na direção oposta à anterior, significa que se cruzou o ponto de mínimo sem alcançá-lo e que agora se está do lado oposto do vale. Neste caso, o próximo passo será reposicionar em algum local entre os dois últimos pontos. O terceiro caso ocorre quando a inclinação atual está na mesma direção que a anterior, mas tem o mesmo tamanho ou tamanho maior. Seguindo cegamente a fórmula, é posssível que a solução fosse para a direção oposta à desejada, ou seja um máximo local.

Para contornar esta terceira situação, o autor definiu, a fim de evitar grandes pesos arbitrários resultantes de um possível denominador pequeno na equação (B.4), um fator multiplicativo que impede que o peso atual seja muitas vezes maior que o peso anterior. FAHLMAN o chamou de 'fator de crescimento máximo'. Desta forma, a nenhum peso é permitido ser maior, em magnitude, que υ vezes o peso anterior. A idéia é aproveitar a rápida convergência em direção ao mínimo local, mas dentro de certos limites.

Assim, o Quickprop tem dois parâmetros, sendo o primeiro a taxa de aprendizagem para o gradiente descendente, e o segundo parâmetro υ, o qual limita o tamanho do passo (o valor padrão para υ é 1.75).

B4. RPROP

É um algoritmo devido a RIEDMILLER e BRAUN(1993). O princípio básico do Rprop é eliminar a má influência do tamanho da derivada parcial no tamanho do passo de atualização. Como conseqüência, somente o sinal da derivada parcial é considerado para indicar a direção de atualização do peso. O tamanho do peso é atualizado por um 'peso específico', o qual é adotado como tendo um valor fixo a fim de diminuir o número de parâmetros a ser ajustado durante o treinamento da rede neural, Δ_{kj}:

$$\Delta w_{kj}(t) = \begin{cases} -\Delta_{kj}(t), & \text{se } \partial E(t)/\partial w_{kj} > 0 \\ +\Delta_{kj}(t), & \text{se } \partial E(t)/\partial w_{kj} < 0 \\ 0, & \text{se de outra forma} \end{cases} \quad \text{(B.9)}$$

O segundo passo do algoritmo Rprop é determinar os novos valores de $\Delta_{kj}(t)$. Isto é baseado no processo de adaptação dependente do sinal, similar à taxa de aprendizagem da equação (B.3)

$$\Delta_{kj}^{(t)} = \begin{cases} \eta^+ * \Delta_{kj}^{(t-1)}, & \text{se} \dfrac{\partial E^{(t-1)}}{\partial w_{kj}} * \dfrac{\partial E^{(t)}}{\partial w_{kj}} > 0 \\ \eta^- * \Delta_{kj}^{(t-1)}, & \text{se} \dfrac{\partial E^{(t-1)}}{\partial w_{kj}} * \dfrac{\partial E^{(t)}}{\partial w_{kj}} < 0 \\ \Delta_{kj}^{(t-1)} & \text{de outra forma} \end{cases} \quad \text{(B.10)}$$

onde $0 < \eta^- < 1 < \eta^+$.

No início, todos os valores atualizados são ajustados para um valor Δ_0, o qual é um dos parâmetros do Rprop. Desde que Δ_0 determina diretamente o tamanho do primeiro peso na primeira iteração, deve ser escolhido de acordo com os valores iniciais dos próprios pesos, por exemplo $\Delta_0 = 0{,}1$. A escolha deste valor não é crítica para o aprendizado através deste algoritmo.

A fim de impedir que os pesos se tornem muito grandes, o máximo peso determinado pelo atualizador de pesos deve ser limitado. O limite superior é ajustado pelo segundo parâmetro do Rprop, $\Delta_{máx}$. O valor deste limitante superior á algo arbitrário e pode ser escolhido com 50, por exemplo. Usualmente, a convergência é mais a este parâmetro.

Os fatores de acréscimo e o de decréscimo são fixados em $\eta^+ = 1{,}2$ e $\eta^- = 0{,}5$. São Estes valores são baseados em considerações teóricas e empíricas. Desta forma se reduz o número de parâmetros livres a dois, a saber, Δ_0 e $\Delta_{máx}$.

Rprop sofre dos mesmos problemas que qualquer um dos métodos apresentados anteriormente. Por causa da adaptação ser baseada na estimação da topologia da função erro, ambas adaptação e atualização do peso podem ser feitas antes de a informação completa do gradiente estar disponível, em outras palavras após cada padrão ter sido apresentado e a soma dos erros dos padrões ser conhecida. Por conseguinte, procedimentos de aprendizagem adaptativos são tipicamente baseados em aprendizagem por amostras de padrões. Isto possivelmente reduz sua eficiência em treinamentos com padrões redundantes quando comparados ao gradiente descendente estocástico.

Além do mais, um esquema restrito de adaptação local deixa a desejar quanto à visão global que as técnicas globais podem ter. Se por exemplo, uma direção de busca

ótima para o mínimo se encontra ao longo de uma diagonal, o esquema local tentará decrescer o erro em cada dimensão, cuidadosamente buscando o mínimo local com pequenos passos na atualização dos pesos.

APÊNDICE C: PROGRAMA PARA TREINAMENTO DE REDES NEURAIS EM C.

```
/*************************************************************************
ESTE PROGRAMA ESTÁ FUNCIONANDO PARA O CASO DE RBF COM CLUSTERING AUTOMÁTICO
O PRÓPRIO PROGRAMA TEM UM ALGORITMO QUE DETERMINA OS CENTROS PELO MÉTODO
MIN_MAX.COM BACKPROPAGATION APENAS (especial para Visual C++).
ESTÃ COM O PROCEDIMENTO DE INFERENCIAMENTO.
TEM UMA MODIFICA□ÇO NA SAÍDA DE CADA CLUSTER SEGUNDO PLATT.
*************************************************************************/

//02.07.98

#include<graphics.h>
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<malloc.h>
#include<ctype.h>
#include<string.h>
#include<conio.h>
#include<time.h>
#include<process.h>

#define TM(size) (float*)malloc(size*sizeof(float))
#define TMD(size) (double*)malloc(size*sizeof(double))
#define TMI(size) (int*)malloc(size*sizeof(int))
#define TMC(size) (char*)malloc(size*sizeof(char))

#define A(i,j,z) (*(a+nmn*nev*(i)+nev*(j)+z))
#define AAA(i,j) (*(aaa+nev*(i)+j))
#define AAAaux(i,j) (*(aaaaux+nev*(i)+j))
#define Cij(i,j,k) (*(cij+nai*nev*(i)+nev*(j)+k))
#define D(i,j) (*(d+nev*(i)+j))
#define Dmax(i,j) (*(dmax+nev*(i)+j))
#define Dmin(i,j) (*(dmin+nev*(i)+j))
#define Dminb(i,j) (*(dminb+nev*(i)+j))
#define DL(i,j) (*(dl+nev*(i)+j))
#define DLL(i,j) (*(dll+nev*(i)+j))
#define DLmin(i,j) (*(dlmin+nev*(i)+j))
#define Fi(i,j) (*(fi+nev*(i)+j))
#define X_Cij(i,j,k) (*(x_cij+nai*nev*(i)+nev*(j)+k))
#define N_Eucl(i,j) (*(n_eucl+nev*(i)+j))
#define N_EuclB(i,j) (*(n_euclb+nev*(i)+j))
#define I(i,j) (*(ii+nev*(i)+j))
#define IL(i,j) (*(il+nev*(i)+j))
#define ILD(i,j) (*(ild+1*(i)+j))
#define ILmin(i,j) (*(ilmin+nev*(i)+j))
#define Th(i,j) (*(th+(3)*(i)+j))
#define WA(j,z) (*(wa+nah*(j)+z))
#define WAD(i,j) (*(wad+nah*(i)+j))
#define WAP(j,z) (*(wap+nah*(j)+z))
#define WB(j,z) (*(wb+nad*(j)+z))
#define WBD(i,j) (*(wbd+nad*(i)+j))
#define WBP(j,z) (*(wbp+nad*(j)+z))
#define WDEL3(i,j) (*(wdel3+nad*(i)+j))
#define X(i,j,z) (*(x+nmn*nev*(i)+nev*(j)+z))
#define Deltat2(i,j,z) (*(delt2+nah*nev*(i)+nev*(j)+z))
#define Deltat3(i,j,z) (*(delt3+nad*nev*(i)+nev*(j)+z))
#define Deltat1(i,j,z) (*(delt1+nah*nev*(i)+nev*(j)+z))
#define Delta2(i,j,z) (*(del2+nah*nev*(i)+nev*(j)+z))
#define Delta3(i,j,z) (*(del3+nad*nev*(i)+nev*(j)+z))
#define DIF1(i,j) (*(dif1+nev*(i)+j))
#define DIF1A(i,j) (*(dif1a+nev*(i)+j))
```

```
#define Daux(i,j) (*(daux+nev*(i)+j))
#define DIF2(i,j) (*(dif2+nev*(i)+j))
#define DER2(i,j) (*(der2+nah*(i)+j))
#define DERK2(i,j) (*(derk2+nah*(i)+j))
#define DER3(i,j) (*(der3+nad*(i)+j))
#define DEL2(i,j) (*(delm2+nad*(i)+j))
#define DEL3(i,j) (*(delm3+nad*(i)+j))
#define Deltaw(i,j) (*(deltaw+1*(i)+j))
#define Deltawa(i,j,k) (*(delwa+nai*nah*(i)+nah*(j)+k))
#define Deltawap(i,j,k) (*(delwap+nai*nah*(i)+nah*(j)+k))
#define Deltawb(i,j,k) (*(delwb+(nah+1)*nad*(i)+nad*(j)+k))
#define Deltawbp(i,j,k) (*(delwbp+(nah+1)*nad*(i)+nad*(j)+k))
#define J(i,j) (*(jac+ncj*(i)+j))
#define JTJ(i,j) (*(jtj+ncj*(i)+j))
#define JI(i,j) (*(ji+ncj*(i)+j))
#define J_1(i,j) (*(j_1+ncj*(i)+j))
#define JTJT(i,j) (*(jtjt+nlj*(i)+j))
#define miI(i,j) (*(mii+ncj*(i)+j))
#define P1f(i,j) (*(p1f+rf*(i)+j))
#define Uf(j,z) (*(uf+rf*(j)+z))
#define Ex(i,j) (*(ex+1*(i)+j))
#define HN1_1(i,j) (*(hn1_1+ncj*(i)+j))
#define G1(i,j) (*(g1+1*(i)+j))
#define G0(i,j) (*(g0+1*(i)+j))
#define DG(i,j) (*(dg+1*(i)+j))
#define HN_1(i,j) (*(hn_1+ncj*(i)+j))
#define DHN_1(i,j) (*(dhn_1+ncj*(i)+j))
#define LAMB(i,j) (*(lamb+ncj*(i)+j))
#define S(i,j) (*(s+1*(i)+j))
#define R1(i,j) (*(r1+1*(i)+j))
#define R2(i,j) (*(r2+1*(i)+j))
#define R3(i,j) (*(r3+ncj*(i)+j))
#define R4(i,j) (*(r4+ncj*(i)+j))
#define R5(i,j) (*(r5+ncj*(i)+j))
#define COEF(i,j) (*(coef+1*(i)+j))
#define COEF_D(i,j) (*(coefd+1*(i)+j))

float I[30][3000];
float D[30][3000];
float ID[30][3000];
float IL[30][1];
float ILmin[30][1];
float WA[150][150];
float WB[250][50];
float COEF[30][1];
float Cij[250][50];
int marc_nad[3000];
int sinn[250];
int sinnt[250];

void determinar(void);
void treinar(void);
void ler_trein(void);
void ler_inf(void);

void backs(int rf,int rs,int *irf,float *p1f,float *uf);
int  lu(int rf, int *irf, float *p1f);

char menu(void);

int iter,cont,kl,dev,met,resp,marc_abs,max_centros;
int cont1,i,j,z,k,ki,kev,b,v,ncj,ncj,ptx,pty;
float *p1f,*uf,PA,*cij,*fi,*x_cij,*n_eucl,*n_eucl1b,MDAp,Ep1aux,Ep2aux,*daux;
float *dlmin,*dmin,*dminb,*dmax,N_atual,N_ant,q;
floattol,mat,ant,aux,y,ETA,ALFA,IM,DM,MDA,t,Ep1,Ep2,Ept,EPF,mi,BETA,MAA,MAAp,nu;
float*il,*ild,*d,*dl,*dll,*th,*wa,*wad,*wap,*wb,*wbd,*wbp,*delt1,*delt2,*delt3,
```

```
*del2,*del3,*der2,*der3,*deltaw,*delwa,*delwap,*delwb,*delwbp,*difl,*difla,*dif2,*aa
a,*jac,*jtj,*ji,*j_1,*jtjt,*mii,*ex,*del3,*delm2,*delm3,*derk2,*wdel3,
*r1,*r2,*r3,*r4,*r5,*hn1_1,*g1,*g0,*dg,*hn_1,*dhn_1,*s,*lamb,lambda;
int nai,nad,nah,nah_cent,nch,eai,ead,mud,mad,nev,ncj,nlj,*irp,*irf,
*irl,rf,rs,bom;
float *ii,*teste,*ilmin,*coef,*coefd,sigma,Dmaximo_anterior,Dmaximo_atual;
double *a,*x,*aaaaux;
char tt;
double fabs();
double log();
int nmn;

//nai: numero de variaveis do problema
//nev: numero de elementos ou de amostras do problema
main()
{ //ABRE A FUNCAO PRINCIPAL
char ch;
/*int gdriver=DETECT,gmode,errorcode;

initgraph(&gdriver,&gmode," ");

errorcode=graphresult();
if (errorcode != grOk)
   {
   printf("Erro de Funcao Grafica: %s\n",
   grapherrormsg(errorcode));
   printf("Aperte uma tecla para parar:");
   getch();
   exit(1);
  }
setbkcolor(MAGENTA);
*/

for(;;) {
        ch=menu();
                          switch(ch)
                {
                 case 'D': determinar();
                           break;
                 case 'L': ler_trein();
                           break;
                 case 'I': ler_inf();
                           break;
                 case 'T': treinar();
                           break;
                 case 'S': exit(0);
                }
        }

} //FECHA A FUNCAO PRINCIPAL

void treinar(void)
{ //INICIO DA SUBROTINA DE TREINAMENTO
FILE *fp;
int arquit;
char pa[20];

printf("QUAL O NOME DO ARQUIVO DE SAIDA\n");
gets(pa);

if((fp=fopen(pa,"w"))==NULL)
{
printf("NAO E POSSIVEL ABRIR O ARQUIVO\n");
return;
}
```

```
nch=1;

nah=nev;//PARA O CASO MAIS SIMPLES DE RBF

nmn=(nai+nah+nad);

met=1; //0:ML, 1:BP, 2:QN;
sigma=0.0125;//MUDAR PARA O INFERENCIADOR TAMB□M
ETA=0.98;
ALFA=0.95;
q=2.67;
iter=0;
v=0;
cont=0;
cont1=1;
Ep1=0;
Ep2=0;
Ept=0;
dev=0;
num1=0;
den1=0;ptx=0,pty=0;
MAAp=0;MAA=0;
bom=0;
max_centros=0;


ii=TM(nai*nev);
il=TM(nai*nev);
ild=TM(nai*1);
ilmin=TM(nai*nev);
coef=TM(nai*1);
coefd=TM(nad*1);
d=TM(nad*nev);
dl=TM(nad*nev);
dll=TM(nad*nev);
dmin=TM(nev*nev);
dminb=TM(nev*nev);
dmax=TM(nev*nev);
dlmin=TM(nad*nev);
x=TMD(3*nmn*nev);
th=TM(nmn*3);
a=TMD(3*nmn*nev);
aaa=TM(nev*nev);
aaaaux=TMD(nev*nev);
wa=TM(nai*nah); *wa=0;
wap=TM(nai*nah); *wap=0;
wb=TM((nah+1)*nad); *wb=0;
wbp=TM((nah+1)*nad); *wbp=0;
dif1=TM(nad*nev); *dif1=0;
difla=TM(nad*nev);
dif2=TM(nad*nev); *dif2=0;
//der2=TM(3*nah);
der3=TM(3*nad);
//del2=TM(3*nah*nev);
del3=TM(3*nad*nev);
//delt2=TM(3*nah*nev);
delt3=TM(3*nad*nev);
delt1=TM(3*nah*nev);
deltaw=TM(ncj*1); *deltaw=0;
daux=TM(nad*nev);
//delwa=TM(3*nai*nah); *deltaw=0;
//delwap=TM(3*nai*nah); *delwap=0;
delwb=TM(3*(nah+1)*nad);  *delwb=0;
delwbp=TM(3*(nah+1)*nad); *delwbp=0;
fi=TM(nah*nev);
x_cij=TM(nah*nai*nev);
```

```
n_eucl=TM(nah*nev);
n_euclb=TM(nah*nev);
jac=TM(nlj*ncj);
jtj=TM(ncj*ncj);
ji=TM(ncj*ncj);
irp=TMI(ncj);
j_1=TM(ncj*ncj);
jtjt=TM(ncj*nlj);
mii=TM(ncj*ncj);
ex=TM(nlj*1);
delm2=TM(nah*nad);
delm3=TM(nad*nad);
derk2=TM(nah*nah);
wdel3=TM(nad*nad);
cij=TM(nah*nai*nev);


if(met==2)
{
dhn_1=TM(ncj*ncj);
hn1_1=TM(ncj*ncj);
g1=TM(ncj*1);
g0=TM(ncj*1);
dg=TM(ncj*1);
hn_1=TM(ncj*ncj);
s=TM(ncj*1);
lamb=TM(ncj*ncj);
irl=TMI(ncj);
r1=TM(ncj*1);
r2=TM(ncj*1);
r3=TM(ncj*ncj);
r4=TM(ncj*ncj);
r5=TM(ncj*ncj);
}

//printf("%d",fprintf(fp,"%d %d %d %d",nai,nah,nad,nev);
//VALORES RANDOMICOS PARA W(I,J,Z)
//VALORES DE wb
//randomize();
srand(5);
y=0;

for(k=0,i=0;i<nah+1;i++)
    for(j=0;j<nad;j++)
      {
        y=(rand()%10000);
        y/=10000;
        if(y>-0.5 && y<0.5)
        WB(i,j)=y;
        if(y<-0.5)
        WB(i,j)=y+0.5;
        if(y>0.5)
        WB(i,j)=y-0.5;
//      printf("WB(%d,%d)=%f",i,j,WB(i,j));
//      printf("\n");
//      tt=getche();
      }
//tt=getche();

for(i=0;i<nai;i++)
    {
    for(j=0;j<nev;j++)
       {
       I(i,j)=(I[i][j]);
       IL(i,j)=I(i,j);
       ILmin(i,j)=I(i,j);
```

```
//     printf("I(%d,%d):%f",i,j,I(i,j));
//     printf(" IL(%d,%d):%f",i,j,IL(i,j));
//     printf(" ILmin(%d,%d):%f\n",i,j,ILmin(i,j));
       }
//     tt=getche();
//     printf("\n");
     }

//NORMALIZACAO DOS VALORES DE ENTRADA

if(nev>1)
{//1
for(i=0;i<nai;i++)
{//2
    for(j=1;j<nev;j++)
       {
       ant=IL(i,j); mat=IL(i,j-1);
       if(fabs(IL(i,j))<fabs(IL(i,j-1)))
       {
       IL(i,j)=IL(i,j-1);
       }
       else IL(i,j)=IL(i,j);
       {
       IL(i,0)=IL(i,j);
       }
       }
       mat=IL(i,0);
//     printf("IL(%d,0):%f\n",i,IL(i,0));
}//2
}//1

for(j=0,i=0;i<nai;i++)
   {
   ILD(i,j)=IL(i,j);
//   printf("ILD(%d,0):%f\n",i,ILD(i,0));
   }

////////////////////////////////////////////////////////////////////////
if(nev>1)
{
for(i=0;i<nai;i++)
{
    for(j=1;j<nev;j++)
       {
       ant=ILmin(i,j); mat=ILmin(i,j-1);
       if(fabs(ILmin(i,j))>fabs(ILmin(i,j-1)))
       ILmin(i,j)=ILmin(i,j-1);
       else ILmin(i,j)=ILmin(i,j);
       ILmin(i,0)=ILmin(i,j);
       }
       mat=ILmin(i,0);
//     printf("ILmin(%d,0)=%f\n",i,ILmin(i,0));

}
}
////////////////////////////////////////////////////////////////////////

for(i=0;i<nai;i++)
{
//     {
    if((IL(i,0)-ILmin(i,0)))
    {
//     if(IL(i,0)>1)
    COEF(i,0)=1/(IL(i,0)-ILmin(i,0));
//     else
//     COEF(i,0)=1;
```

```
    }
    else
    COEF(i,0)=1/IL(i,0);
//    printf("COEF(%d,0):%f\n",i,COEF(i,0));
}

if(nai>1)
{
aux=0;
for(j=0,i=1;i<nai;i++)
{
    if(fabs(IL(i,j))<fabs(IL(i-1,j)))
       {
      ant=IL(i,j); mat=IL(i-1,j);
      IL(i,j)=IL(i-1,j);
       }
    else IL(i,j)=IL(i,j);
        IM=IL(i,j);
}
}
else IM=IL(0,0);

if(fabs(IM>1))
{
for(i=0;i<nai;i++)
    for(j=0;j<nev;j++)
       {
//      printf("I(%d,%d)=%f",i,j,I(i,j));
//      I(i,j)/=IM;
//      printf("  COEF(%d,0):%f",i,COEF(i,0));
        I(i,j)=COEF(i,0)*(I(i,j)-ILmin(i,0));
//      printf("  I(%d,%d):%f\n",i,j,I(i,j));
//      tt=getche();
       }
}

if(!IM)
printf("NAO E POSSIVEL TREINAR A REDE COM ESTES VALORES\n");

//DETERINA□ÇO DOS CENTROS DA CAMADA OCULTA

//CASO SIMPLES CADA CENTRO COMO VALOR DE ENTRADA(nah=nev)

if(nev<10)
{
for(k=0,i=0;i<nah;i++)
    {
     for(j=0;j<nai;j++)
        {
         mat=I(j,k);
         Cij(i,j,0)=I(j,k);
//       printf("Cij(%d,%d,%d)=%f\n",i,j,0,Cij(i,j,0));
        }
        k++;
//      printf("\n");
     }
}
if(nev>9)
{//ABRE A ATRIBUI□ÇO DO CENTRO

for(i=0;i<nai;i++)
     Cij(0,i,0)=I(i,0);

nah_cent=1;
ki=1;
Dmax(nev-1,0)=0;
```

```
Dmaximo_atual=0;

do
{
Dmaximo_anterior=Dmaximo_atual;
Dmaximo_atual=0;

for(kev=0;kev<nev;kev++)
{ //ABRE O LOOPING DE LEITURA DAS AMOSTRAS PARA ATRIBUI□ÇO DOS CENTROS
for(i=0;i<nah_cent;i++)
    for(j=0;j<nai;j++)
       {      X_Cij(i,j,kev)=I(j,kev)-Cij(i,j,0);
//     printf("I(%d,%d):%f""  ""Cij(%d,%d,0):%f"" ""X(%d,%d,%d):%f\n",j,kev,
//     I(j,kev),i,j,Cij(i,j,0),i,j,kev,X_Cij(i,j,kev));
//     tt=getche();
       }

//CµLCULO DE N_Euc (NORMA EUCLIDIANA)

for(aux=0,i=0;i<nah_cent;i++)
       {
    for(aux=0,j=0;j<nai;j++)
        aux+=(X_Cij(i,j,kev)*X_Cij(i,j,kev));
        N_Eucl(i,kev)=aux;
        N_EuclB(i,kev)=aux;
//      printf("N_Eucl(%d,%d)=%f\n",i,kev,N_Eucl(i,kev));
//tt=getche();
       }
}//FECHA PARA A DISTRIBUI□ÇO DOS CENTROS

if(nah_cent>1)
{
for(j=0;j<nev;j++)
    {
    for(i=1;i<nah_cent;i++)
       {
          ant=N_Eucl(i,j); mat=N_Eucl(i-1,j);
          if(i==1)
          N_ant=N_Eucl(i-1,j);
          N_atual=N_Eucl(i,j);
          if(fabs(N_atual)>fabs(N_ant))
          {
          Dmin(i,j)=N_ant; //N_Eucl(i-1,j);
          Dminb(i,j)=Dmin(i,j);
          N_ant=Dmin(i,j);
          }
          else
          {
          Dmin(i,j)=N_Eucl(i,j);
          Dminb(i,j)=Dmin(i,j);
          N_ant=Dmin(i,j);
          }
//        printf("Dmin(%d,%d)=%f\n",i,j,Dmin(i,j));
       }
    }
}

//PARA A DETERMINA□ÇO DO SEGUNDO CENTRO
if(nah_cent==1) //INÖCIO DO nah_cent==1
{
for(i=0;i<nah_cent;i++)
    {
     for(j=1;j<nev;j++)
        {
          ant=N_EuclB(i,j-1); mat=N_EuclB(i,j);
          if(fabs(N_EuclB(i,j))<fabs(N_EuclB(i,j-1)))
```

```
          {
          Dmax(j,ki)=N_EuclB(i,j-1);
          N_EuclB(i,j)=N_EuclB(i,j-1);
          }
          else if(fabs(N_EuclB(i,j))>fabs(N_EuclB(i,j-1)))
          {
          ant=N_EuclB(i,j-1); mat=N_EuclB(i,j);
          N_EuclB(i,j)=N_EuclB(i,j);
          Dmax(j,ki)=N_EuclB(i,j);
          }
          Dmaximo_atual=Dmax(j,ki);
//        printf("Dmax(%d,%d)=%f\n",j,ki,Dmax(j,ki));
        }
      }

for(i=0;i<nev;i++)
    {
     ant=Dmax(nev-1,ki); mat=N_Eucl(nah_cent-1,i);
     if((Dmax(nev-1,ki))==(N_Eucl(nah_cent-1,i)))
     {
     sinn[nah_cent]=i;
     goto fim1;
     }
    }
     fim1:;
     mat=sinn[nah_cent];

for(i=0;i<nai;i++)
    {
      mat=I(i,sinn[nah_cent]);
    Cij(nah_cent,i,0)=I(i,sinn[nah_cent]);
//  printf("  Cij(%d,%d,0,)=%f",nah_cent,i,Cij(nah_cent,i,0));
    }
}//FIM DO nah_cent==1

if(nah_cent>1)
{
  for(j=1;j<nev;j++)
      {
       ant=Dminb(nah_cent-1,j-1);  mat=Dminb(nah_cent-1,j);
       if((Dminb(nah_cent-1,j))<(Dminb(nah_cent-1,j-1)))
       Dminb(nah_cent-1,j)=Dminb(nah_cent-1,j-1);
       else
       Dminb(nah_cent-1,j)=Dmin(nah_cent-1,j);
//       printf("Dminb(%d,%d):%f\n",nah_cent-1,j,Dminb(nah_cent-1,j));
      }
Dmaximo_atual=Dminb(nah_cent-1,j-1);

for(i=0;i<nev;i++)
    {
     ant=Dminb(nah_cent-1,nev-1); mat=Dmin(nah_cent-1,i);
     if((Dminb(nah_cent-1,nev-1))==(Dmin(nah_cent-1,i)))
     {
     sinnt[nah_cent]=i;
     goto fim2;
     }
    }
     fim2:;
     mat=sinnt[nah_cent];

for(i=0;i<nai;i++)
    {
    Cij(nah_cent,i,0)=I(i,sinnt[nah_cent]);
//    printf("  Cij(%d,%d,0,)=%f\n",nah_cent,i,Cij(nah_cent,i,0));
    }
}
```

```
//tt=getche();

nah_cent++;
ant=Dmaximo_anterior;
mat=Dmaximo_atual;

max_centros++;

if (Dmaximo_atual==0) //PARA EVITAR REPETIR O PRIMEIRO
goto inicio;

}
while(((Dmaximo_atual)>(0.0125*Dmaximo_anterior)) && (max_centros<100));
}//FECHA A ATRIBUI□ÇO DO CENTRO
inicio:;

/*
//NORMALIZACAO DOS VALORES DE ENTRADA

if(nev>1)
{//1
for(i=0;i<nai;i++)
{//2
    for(j=1;j<nev;j++)
       {
       ant=IL(i,j); mat=IL(i,j-1);
       if(fabs(IL(i,j))<fabs(IL(i,j-1)))
       {
       IL(i,j)=IL(i,j-1);
       }
       else IL(i,j)=IL(i,j);
       {
       IL(i,0)=IL(i,j);
       }
       }
       mat=IL(i,0);
//     printf("IL(%d,0):%f\n",i,IL(i,0));
}//2
}//1

for(j=0,i=0;i<nai;i++)
   {
   ILD(i,j)=IL(i,j);
//   printf("ILD(%d,0):%f\n",i,ILD(i,0));
   }

////////////////////////////////////////////////////////////////////////
if(nev>1)
{
for(i=0;i<nai;i++)
{
    for(j=1;j<nev;j++)
       {
       ant=ILmin(i,j); mat=ILmin(i,j-1);
       if(fabs(ILmin(i,j))>fabs(ILmin(i,j-1)))
       ILmin(i,j)=ILmin(i,j-1);
       else ILmin(i,j)=ILmin(i,j);
       ILmin(i,0)=ILmin(i,j);
       }
       mat=ILmin(i,0);
//     printf("ILmin(%d,0)=%f\n",i,ILmin(i,0));

}
}
////////////////////////////////////////////////////////////////////////
```

```
for(i=0;i<nai;i++)
{
//     {
     if((IL(i,0)-ILmin(i,0)))
     {
//     if(IL(i,0)>1)
     COEF(i,0)=1/(IL(i,0)-ILmin(i,0));
//     else
//     COEF(i,0)=1;
     }
     else
     COEF(i,0)=1/IL(i,0);
//     printf("COEF(%d,0):%f\n",i,COEF(i,0));
}

if(nai>1)
{
aux=0;
for(j=0,i=1;i<nai;i++)
{
     if(fabs(IL(i,j))<fabs(IL(i-1,j)))
        {
        ant=IL(i,j); mat=IL(i-1,j);
        IL(i,j)=IL(i-1,j);
        }
     else IL(i,j)=IL(i,j);
         IM=IL(i,j);
}
}
else IM=IL(0,0);

if(fabs(IM>1))
{
for(i=0;i<nai;i++)
     for(j=0;j<nev;j++)
        {
//      printf("I(%d,%d)=%f",i,j,I(i,j));
//      I(i,j)/=IM;
//      printf("  COEF(%d,0):%f",i,COEF(i,0));
        I(i,j)=COEF(i,0)*(I(i,j)-ILmin(i,0));
//      printf("  I(%d,%d):%f\n",i,j,I(i,j));
//      tt=getche();
        }
}

if(!IM)
printf("NAO E POSSIVEL TREINAR A REDE COM ESTES VALORES\n");
*/
// NORMALIZACAO DOS VALORES DE SAIDA

for(i=0;i<nad;i++)
{
     for(j=0;j<nev;j++)
        {
        D(i,j)=D[i][j];
        DL(i,j)=D(i,j);
        DLmin(i,j)=D(i,j);
//      printf("  DL(%d,%d):%f",i,j,DL(i,j));
        }
//      printf("\n");
}

if(nev>1)
{ //1
aux=0;
```

```
for(i=0;i<nad;i++)
    {//2
    for(j=1;j<nev;j++)
       {//3
        ant=DL(i,j); mat=DL(i,j-1);
       if(fabs(DL(i,j))<fabs(DL(i,j-1)))
          {//4
           DL(i,j)=DL(i,j-1);
              DLL(i,j)=DL(i,j-1);
          }//4
          else
          {
              DL(i,j)=DL(i,j);
              DLL(i,j)=DL(i,j-1);
          }
          DL(i,0)=DL(i,j);
          DLL(i,0)=DLL(i,j);
       }//3
          mat=DL(i,0);
//        printf("DL(%d,0):%f\n",i,DL(i,0));
     }//2
}//1

if(nev>1)
{
for(i=0;i<nad;i++)
    {
     for(j=1;j<nev;j++)
        {
          ant=DLmin(i,j); mat=DLmin(i,j-1);
          if(fabs(DLmin(i,j))>fabs(DLmin(i,j-1)))
          DLmin(i,j)=DLmin(i,j-1);
          else
          DLmin(i,j)=DLmin(i,j);
          DLmin(i,0)=DLmin(i,j);
//        printf("DLmin(%d,0)=%f\n",i,DLmin(i,0));
        }
     }
}

for(i=0;i<nad;i++)
    {
     mat=DL(i,0)-DLmin(i,0);
     if((DL(i,0)-DLmin(i,0)))
        {
//       if(DL(i,0)>1)
         {
         ant=DL(i,0); mat=DLmin(i,0);
         COEF_D(i,0)=1/(DL(i,0)-DLmin(i,0));
         ant=COEF_D(i,0);
         }
//       else
//       COEF_D(i,0)=1;
        }
       else
        {
        COEF_D(i,0)=1/DL(i,0);
        mat=COEF_D(i,0);
        }
     }

if(nad>1)
{
for(j=0,i=1;i<nad;i++)
{
    if(fabs(DL(i,j))<fabs(DL(i-1,j)))
```

```
        {
        aux=DL(i-1,j);
        DL(i-1,j)=DL(i-1,j);
        DL(i,j)=aux;
         }
     else DL(i,j)=DL(i,j);
         DM=DL(i,j);
}
}
else DM=DL(0,0);

if(fabs(DM>1))
{
for(i=0;i<nad;i++)
     for(j=0;j<nev;j++)
        {
         printf("D(%d,%d)=%f",i,j,D(i,j));
         D(i,j)/=DM;
         printf("  DM=%f  D(%d,%d)=%f\n",DM,i,j,D(i,j));
        }
}

if(!DM)
printf("NAO E POSSIVEL TREINAR A REDE COM ESTES VALORES\n");
/*
for(i=0;i<nah_cent;i++)
      {
      for(j=0;j<nai;j++)
        printf("  Cij(%d,%d,%d):%f\n",i,j,0,Cij(i,j,0));
        printf("\n");
        tt=getche();
       }
tt=getche();
*/
//INICIO DO LOOPING************************************************

nah=nah_cent-1;
do
{ //ABRE O LOOPING
v=0;
k1=0;
for(kev=0;kev<nev;kev++)
{ //ABRE O LOOPING DE LEITURA DAS AMOSTRAS
for(i=0;i<nah;i++)
    {
     for(j=0;j<nai;j++)
        {
        X_Cij(i,j,kev)=I(j,kev)-Cij(i,j,0);
//      printf("I(%d,%d):%f""  ""Cij(%d,%d,%d):%f"" ""X(%d,%d,%d):%f\n",j,kev,
//      I(j,kev),i,j,0,Cij(i,j,0),i,j,kev,X_Cij(i,j,kev));
//      tt=getche();
        }
//      printf("\n");
     }
//CALCULO DE N_Euc (NORMA EUCLIDIANA)

for(i=0;i<nah;i++)
       {
     for(aux=0,j=0;j<nai;j++)
         aux+=(X_Cij(i,j,kev)*X_Cij(i,j,kev));
         N_Eucl(i,kev)=aux;
//       printf("N_Eucl(%d,%d)=%f\n",i,kev,N_Eucl(i,kev));
         }

//CALCULO DE Fi
```

```
//printf("\n\n");
for(i=0;i<nah;i++)
      {
//      printf("N_Eucl(%d,%d)=%f",i,kev,N_Eucl(i,kev));
        mat=pow(sigma,2);
        if(N_Eucl(i,kev)<(q*(2*pow(sigma,2)))) //modifica‡Æo de PLATT
//norm  Fi(i,kev)=exp(-N_Eucl(i,kev)/(2*pow(sigma,2)));
        Fi(i,kev)=(1-(N_Eucl(i,kev)/(q*(2*pow(sigma,2))))); //modif PLATT
        else  Fi(i,kev)=0; //modif PLATT
        mat=Fi(i,kev);
//      printf("  Fi(%d,%d)=%f",i,kev,Fi(i,kev));
//      tt=getche();
//      printf("\n");
      }
//tt=getche();

//CALCULO DE C
//printf("\n\n");
for(aux=0,i=0;i<nad;i++)
   {
    for(z=0;z<nah+1;z++)
          {
           if(!z)
             aux=WB(z,i)*1;
           else if(z)

            {
//printf("  WB1(%d,%d)=%f",z,i,WB(z,i));
//printf("  Fi(%d,%d)=%f",z-1,kev,Fi(z-1,kev));
            aux+=WB(z,i)*Fi(z-1,kev);
//          printf("  aux=%f\n",aux);
           }
          }
            X(2,i,kev)=aux;
            aux=0;
//          printf("\n");
       }

//printf("\n\n");
for(j=0,i=0;i<nad;i++)
///    for(j=0;j<nev;j++)
       {
        A(2,i,kev)=(1/(1+exp(-X(2,i,kev))));
//      printf("C(2,%d,%d)=%f\n",i,kev,A(2,i,kev));
//      t=getche();
       }

////////////////////////////////////////////////////////////////////////////

//printf("\n\n");
for(j=0,i=0;i<nad;i++)
///    for(j=0;j<ead;j++)
       {
        mat=A(2,i,kev); ant=D(i,kev);
        if(bom)
        {
        printf("A(2,%d,%d)=%f",i,kev,A(2,i,kev));
        printf("  D(%d,%d)=%f",i,kev,D(i,kev));
        }
        DIF1A(i,kev)=fabs(fabs(A(2,i,kev)*1)-fabs(D(i,kev)*1));
        Daux(i,kev)=fabs(fabs(A(2,i,kev)*1)-fabs(D(i,kev)*1));
        if(bom)
        printf("  DIF1A(%d,%d)=%f\n",i,kev,DIF1A(i,kev));
       }

aux=0;
```

```
if(nad>1)
{
//for(j=0;j<nad;j++)
{
for(i=1;i<nad;i++)
       if (DIF1A(i-1,kev)>DIF1A(i,kev))
            {
             aux=DIF1A(i,kev);
             DIF1A(i,kev)=(DIF1A(i-1,kev));
             DIF1A(i-1,kev)=aux;
             AAA(kev,kev)=DIF1A(i,kev);
            }
       else AAA(kev,kev)=DIF1A(i,kev);
}

for(i=0;i<nad;i++)
    {
     ant=AAA(kev,kev); mat=Daux(i,kev);
     if((AAA(kev,kev))==(Daux(i,kev)))
     marc_nad[kev]=i;
    }
}

else if(nad==1)
AAA(kev,kev)=DIF1A(0,kev);

if(bom)
printf("\n\nmaior diferenca relativa(%d,%d)=%f\n",kev,kev,AAA(kev,kev));
kl++;

if(kl==nev)
{
for(i=0;i<nev;i++)
    for(j=0;j<nev;j++)
       if(i==j)
           AAAaux(i,0)=AAA(i,j);
}

if(kl==nev)
{//1
aux=0;
for(i=1;i<nev;i++)
    for(j=1;j<nev;j++)
       if((i==j) && (AAA(i-1,j-1)>AAA(i,j)))
          {//2
           aux=AAA(i-1,j-1);
           AAA(i-1,j-1)=AAA(i,j);
           AAA(i,j)=aux;
           MAA=AAA(i,j);
          }//2
       else if((i==j) && ((AAA(i-1,j-1)<AAA(i,j)) || (AAA(i-1,j-1)==AAA(i,j))))
          {//3
           MAA=AAA(i,j);
          }//3

for(i=0;i<nev;i++)
{
//  printf("AAAaux(%d,0):%f\n",i,AAAaux(i,0));
   if(AAAaux(i,0)==MAA)
     marc_abs=i;
}

/*
for(i=0;i<nad;i++)
    {
    for(j=0;j<nev;j++)
```

```
        printf(" %f",Daux(i,j));
        printf("\n");
    }
/*
for(i=0;i<nad;i++)
   printf("COEF_D(%d):%f\n",i,COEF_D(i,0));
tt=getche();
*/

i=0;b=0;
i=marc_nad[marc_abs];
DMAX=MAA;//D[i][marc_abs];
ant=COEF_D(i,0);
mat=DLmin(i,0);
ant=DLL(i,0);
//MDA=fabs((fabs((DMAX/COEF_D(i,0))+DLmin(i,0)))-D[i][marc_abs]);
if(DM>1)
MDA=fabs((fabs((DMAX/COEF_D(i,0))+DLmin(i,0))));//-D[i][marc_abs]);
else if(DM<1)
MDA=DMAX/DLL(i,0);//D[i][marc_abs];
MDAp=MDA*100;////((fabs(MDA))/(DM-DLmin(marc_abs,0)));//D[i][marc_abs]);
//MDA=(fabs(MDA-fabs(D[i][marc_abs])));
b=i;

if(bom)
printf("\n\n\nmaior diferenca absoluta(%d,%d)=%f\n",i,kev,MAA);
}//1

////////////////////////////////////////////////////////////////////////////////

//tt=getche();

//CALCULO DOS TERMOS DE DECLINIO DO GRADIENTE
//for(kev=0;kev<nev;kev++)
//{//ABRE  SEGUNDO kev

//printf("\n\n");
for(j=0,i=0;i<nad;i++)          /////EAD PRECISA SER IGUAL A EAI......
///     for(j=0;j<ead;j++)
       {
//printf("D(%d,%d)=%f   A(2,%d,%d)=%f",i,kev,D(i,kev),i,kev,A(2,i,kev));
        Deltat3(3,i,kev)=D(i,kev)-A(2,i,kev);
        {
        mat=A(2,i,kev); ant=D(i,kev);
//DIF1(i,kev)=fabs(fabs(A(2,i,kev)*DM)-fabs(D(i,kev)*DM)); //
        DIF1(i,kev)=fabs(fabs(A(2,i,kev))-fabs(D(i,kev))); //
        Ep1+=DIF1(i,kev)*DIF1(i,kev);
        }
//printf("  X(2,%d,%d)=%f",i,kev,X(2,i,kev));
DER3(3,i)=(exp(-X(2,i,kev))/((1+exp(-X(2,i,kev)))*(1+exp(-X(2,i,kev)))));
//printf("   DER3(3,%d)=%f\n",i,DER3(3,i));
       }
       if(kev==nev-1)
        Ep1*=0.5;

//printf("\n");
if(met==1)
{
for(i=0;i<nad;i++)
///     for(j=0;j<nev;j++)
       {
//      printf("DER3(3,%d)=%f",i,DER3(3,i));
//      printf("  Deltat3(3,%d,%d)=%f",i,kev,Deltat3(3,i,kev));
        Delta3(3,i,kev)=Deltat3(3,i,kev)*DER3(3,i);
//      printf("  Delta3(3,%d,%d)=%f\n",i,kev,Delta3(3,i,kev));
       }
```

```
}

////////////////////////////////////////////////////////////
//PROCEDIMENTO PARA A MODIFICACAO DE MARQUARDT-LEVENBERG
if(!met || met==2)
{//ABRE O PROCEDIMENTO PARA ML/QN
if(v==nev-1)
{
for(i=0;i<nad;i++)
   {
    Ex(nad*v+i,0)=Deltat3(3,i,kev);
//    printf("  Ex(%d,%d)=%f\n",nad*v+i,0,Deltat3(3,i,kev));
   }
}

for(i=0;i<nad;i++)
    for(j=0;j<nad;j++)
       {
        if(i==j)
        {
        DEL3(i,j)=DER3(3,i);
//      printf("  DEL3(%d,%d)=%f",i,j,DEL3(i,j));
        }
        else
        {
        DEL3(i,j)=0;
//      printf("  DEL3(%d,%d)=%f",i,j,DEL3(i,j));
        }
//      printf("\n");
       }

for(i=0;i<nah;i++)
    {
    for(j=0;j<nah;j++)
       {
        if(i==j)
        {
        DERK2(i,j)=DER2(2,i);
//      printf("  DERK2(%d,%d)=%f",i,j,DERK2(i,j));
        }
        else
        {
        DERK2(i,j)=0;
//      printf("  DERK2(%d,%d)=%f",i,j,DERK2(i,j));
        }
        }
//     printf("\n");
     }

//printf("\n");
for(aux=0,i=0;i<nah;i++)
    {
    for(j=0;j<nad;j++)
       {
        for(z=0;z<nad;z++)
            {
             aux+=WB(i,z)*DEL3(z,j);
            }
             WDEL3(i,j)=aux;
             aux=0;
//           printf("  WDEL3(%d,%d)=%f",i,j,WDEL3(i,j));
       }
//           printf("\n");
     }

//printf("\n");
```

```
for(aux=0,i=0;i<nah;i++)
    {
    for(j=0;j<nad;j++)
       {
        for(z=0;z<nah;z++)
             {
              aux+=DERK2(i,z)*WDEL3(z,j);
             }
              DEL2(i,j)=aux;
              aux=0;
//            printf("  DEL2(%d,%d)=%f",i,j,DEL2(i,j));
       }
//            printf("\n");
    }
//PREENCHIMENTO DA MATRIZ JACOBIANO

if(!v)
{
//printf("\n");
      for(i=0;i<nlj;i++)
   {
    for(j=0;j<ncj;j++)
       {
        J(i,j)=0;
//      printf("  J(%d,%d)=%f",i,j,J(i,j));
       }
//     printf("\n");
    }
}

//printf("\n");
for(ki=0,kl=0,i=(nai*v);i<(nai*(v+1));i++)
    {
     for(k=0,j=0;j<nah*nai;j++)
        {
//       printf("DEL2(%d,%d)=%f",ki,kl,DEL2(ki,kl));
//       printf("  A(0,%d,%d)=%f",k,kev,A(0,k,kev));
         J(i,j)=DEL2(ki,kl)*A(0,k,kev);
//       printf("  J(%d,%d)=%f\n",i,j,J(i,j));
         ki++;
         if(ki==nah)
         {
         ki=0;
         k++;
         }
        }
        kl++;
    }

//printf("_______\n\n");
for(ki=0,kl=0,i=(nad*v);i<(nad*(v+1));i++)
    {
     for(k=0,j=(nai*nah);j<ncj;j++)
        {
//       printf("DEL3(%d,%d)=%f",ki,kl,DEL3(ki,kl));
//       printf("  A(1,%d,%d)=%f",k,kev,A(1,k,kev));
         J(i,j)=DEL3(ki,kl)*A(1,k,kev);
//       printf("  J(%d,%d)=%f\n",i,j,J(i,j));
         ki++;
         if(ki==nad)
         {
         ki=0;
         k++;
         }
        }
        kl++;
```

```
    }
v++;

if((v==nev) && (!met))
{//ABRE O IF v=nev
for(aux=0,i=0;i<ncj;i++)
   {
   for(j=0;j<ncj;j++)
       {
       for(z=0;z<nlj;z++)
           {
            aux+=J(z,i)*J(z,j);
           }
            JTJ(i,j)=aux;
//          printf("  JTJ(%d,%d)=%.12f\n",i,j,JTJ(i,j));
            aux=0;
       }
//          printf("\n");
   }

if(!met)
{
for(i=0;i<ncj;i++)
    {
     for(j=0;j<ncj;j++)
        {
         miI(i,j)=(i==j ? mi : 0);
//       printf("  miI(%d,%d)=%f",i,j,miI(i,j));
        }
//       printf("\n");
    }
}

if(!met)
{
for(i=0;i<ncj;i++)
   {
    for(j=0;j<ncj;j++)
       {
       JI(i,j)=JTJ(i,j)+miI(i,j);
//     printf("  JI(%d,%d)=%f",i,j,JI(i,j));
//     printf("  %f",JI(i,j));
       }
//     printf("\n");
   }
}

if(!met)
{//ABRE O !met===1
for(i=0;i<ncj;i++) irp[i]=i;

for(i=0;i<ncj;i++)
    for(j=0;j<ncj;j++)
       J_1(i,j)=(i==j ? 1 : 0);

//*********************************************************************
lu(ncj,irp,ji);
backs(ncj,ncj,irp,ji,j_1);
//*********************************************************************
//printf("VALORES DO J_1\n");
for(aux=0,i=0;i<ncj;i++)
   {
   for(j=0;j<nlj;j++)
       {
       for(z=0;z<ncj;z++)
           {
```

```
//            printf("  %f",J_1(i,z));
              aux+=J_1(i,z)*J(j,z);
             }
              JTJT(i,j)=aux;
//            printf("\n");
//            printf("  JTJT(%d,%d)=%f",i,j,JTJT(i,j));
              aux=0;
        }
//            printf("\n");
   }

for(aux=0,i=0;i<ncj;i++)
     {
     for(j=0;j<1;j++)
        {
         for(z=0;z<nlj;z++)
              {
               aux+=JTJT(i,z)*Ex(z,0);
              }
               Deltaw(i,j)=aux;
//             printf("  Deltaw(%d,%d)=%f",i,j,Deltaw(i,j));
               aux=0;
        }
//             printf("\n");
     }
}//FECHA O IF !met===1
}//FECHA O IF v=nev

//INICIO DO QUASE-NEWTON
if(met==2)
{//ABRE O QUASE-NEWTON

if(!iter)
{
for(i=0;i<ncj;i++)
   for(j=0;j<ncj;j++)
        {
       HN_1(i,j)=(i==j ? 1 : 0);
       DHN_1(i,j)=0;
       G0(i,0)=0;
        }
}

//DETERMINACAO DE H

for(i=0;i<ncj;i++)
     for(j=0;j<ncj;j++)
        {
         HN1_1(i,j)=DHN_1(i,j)+HN_1(i,j);
         HN_1(i,j)=HN1_1(i,j);
        }


for(aux=0,i=0;i<ncj;i++)
     for(j=0;j<1;j++)
        {
         for(z=0;z<nlj;z++)
              aux+=J(z,i)*Ex(z,j);
              G1(i,j)=aux;
              aux=0;
        }

for(i=0;i<ncj;i++)
     for(j=0;j<1;j++)
        {
         DG(i,j)=G1(i,j)-G0(i,j);
```

```
        G0(i,j)=G1(i,j);
       }

for(aux=0,i=0;i<ncj;i++)
   {
   for(j=0;j<1;j++)
       {
       for(z=0;z<ncj;z++)
           aux+=HN_1(i,z)*G1(z,j);
           S(i,j)=-aux;
//         printf("  S(%d,%d)=%f",i,j,S(i,j));
           aux=0;
       }
//         printf("\n");
   }
//tt=getche();

for(i=0;i<ncj;i++)
   {
    for(j=0;j<ncj;j++)
       {
        LAMB(i,j)=(i==j ? lambda : 0);
//      printf(" LAMB(%d,%d)=%f",i,j,LAMB(i,j));
       }
//      printf("\n");
    }
lambda-=0.1;

for(aux=0,i=0;i<ncj;i++)
    {
    for(j=0;j<1;j++)
       {
        for(z=0;z<ncj;z++)
            {
             aux+=LAMB(i,z)*S(z,j);
            }
             Deltaw(i,j)=aux;
             aux=0;
       }
    }

//DETERMINACAO DO DNH_1

for(aux=0,i=0;i<ncj;i++)
    for(j=0;j<1;j++)
       {
        for(z=0;z<ncj;z++)
            aux+=HN_1(i,z)*DG(z,j);
            R1(i,j)=aux;
            aux=0;
       }

for(i=0;i<ncj;i++)
    {
    for(j=0;j<1;j++)
       {
       R2(i,j)=Deltaw(i,j)-R1(i,j);
//     printf("  Deltaw(%d,%d)=%f",i,j,Deltaw(i,j));
       }
//     printf("\n");
    }

for(aux=0,i=0;i<ncj;i++)
    for(j=0;j<ncj;j++)
       {
        for(z=0;z<1;z++)
```

```
            aux+=R2(i,z)*Deltaw(j,z);
            R3(i,j)=aux;
       }
            aux=0;

for(aux=0,i=0;i<ncj;i++)
    for(j=0;j<ncj;j++)
       {
        for(z=0;z<1;z++)
            aux+=Deltaw(i,z)*R2(j,z);
            R4(i,j)=aux;
            aux=0;
       }

for(aux=0,i=0;i<1;i++)
    for(j=0;j<1;j++)
       {
        for(z=0;z<ncj;z++)
            aux+=DG(z,i)*Deltaw(z,j);
            den1=aux;
            aux=0;
       }

for(i=0;i<ncj;i++)
    for(j=0;j<ncj;j++)
        R4(i,j)/=den1;

for(aux=0,i=0;i<1;i++)
    for(j=0;j<1;j++)
       {
        for(z=0;z<ncj;z++)
             aux+=R2(z,i)*DG(z,j);
             num1=aux;
             aux=0;
       }

for(aux=0,i=0;i<ncj;i++)
    for(j=0;j<ncj;j++)
       {
        for(z=0;z<1;z++)
            aux+=Deltaw(i,z)*Deltaw(j,z);
            R5(i,j)=aux;
            aux=0;
       }

for(i=0;i<ncj;i++)
     for(j=0;j<ncj;j++)
         R5(i,j)*=num1;

for(i=0;i<ncj;i++)
    for(j=0;j<ncj;j++)
       R5(i,j)/=(den1*den1);

for(i=0;i<ncj;i++)
     for(j=0;j<ncj;j++)
        DHN_1(i,j)=(R3(i,j)+R4(i,j)-R5(i,j));

}//FECHA O QUASE-NEWTON
}//FECHA O PROCEDIMENTO PARA ML/QN
////////////////////////////////////////////////////////////////

if(((met==0 || met==2) && v==nev) || met==1)
{//ABRE O if3


//PROCEDIMENTO PARA O GDR
```

```
//CALCULO DAS VARIACOES PONDERAIS

//Caso B
if(!iter)
{
//printf("\n\n");
///for(z=0;z<ead;z++)
for(z=0,i=0;i<nah+1;i++)
    {
     for(j=0;j<nad;j++)
        {
         Deltawb(1,i,j)=0;
         Deltawbp(1,i,j)=0;
//printf("  Deltawbp(1,%d,%d)=%f",i,j,Deltawbp(1,i,j));
        }
//       printf("\n");
     }
}
//printf("\n\n");

if(met==1)
{//ABRE O METODO PARA B_1
///for(z=0;z<ead;z++)
    for(z=0,i=0;i<nah+1;i++)
       {
       for(j=0;j<nad;j++)
          {
           if(!i)
           Deltawb(1,i,j)=Delta3(3,j,kev)*1;
           else if(i)
             {
//printf("Fi(%d,%d)=%f",i-1,kev,Fi(i-1,kev));
//printf("   Delta3(3,%d,%d)=%f",j,kev,Delta3(3,j,kev));
            Deltawb(1,i,j)=(ETA*Delta3(3,j,kev)*Fi(i-1,kev));
            ant=Deltawbp(1,i,j);
            Deltawb(1,i,j)+=(ALFA*Deltawbp(1,i,j));
            Deltawbp(1,i,j)=Deltawb(1,i,j);
//printf("   Deltawb(1,%d,%d)=%f\n",i,j,Deltawb(1,i,j));
             }
          }
       }
}//FECHA O METODO PARA B_1

if(!met || met==2)
{//ABRE O METODO PARA B_0
for(i=0;i<nah;i++)
    {
    for(j=0;j<nad;j++)
       {
        Deltawb(1,i,j)=Deltaw((nai*nah)+nad*i+j,0);
//      printf(" Deltawb(1,%d,%d)=%f",i,j,Deltawb(1,i,j));
        Deltawbp(1,i,j)=Deltawb(1,i,j);
       }
//     printf("\n");
     }
}//FECHA O METODO PARA B_0

//ATUALIZACAO DOS PESOS

//printf("\n");
//CASO B
if(!iter) // || met==0)// && v==nev)
{
//printf("\n");
//for(z=0;z<ead;z++)
    for(z=0,i=0;i<nah+1;i++)
```

```
        {
       for(j=0;j<nad;j++)
            {
             WBP(i,j)=WB(i,j);
//           printf("  WBP(%d,%d)=%f",i,j,WBP(i,j));
            }
//          printf("\n");
        }
}
//printf("\n\nVALORES DE WB CORRIGIDOS\n\n");
///for(z=0;z<nev;z++)
//if((met==0 && v==nev) || met==1)

for(z=0,i=0;i<nah+1;i++)
    {
    for(j=0;j<nad;j++)
       {
//printf("WBP(%d,%d)=%f",i,j,WBP(i,j));
//printf("  Deltawb(1,%d,%d)=%f",i,j,Deltawb(1,i,j));
        t=WBP(i,j);
        WB(i,j)=t+Deltawb(1,i,j);
        if(met==1)
        WBP(i,j)=WB(i,j);
//      if(MAA*DM<0.09000001 && MAA*DM>0)
//      {
//printf("  WB2(%d,%d)=%f",i,j,WB(i,j));
//       tt=getche();
//      }
       }
     }

//}//FECHA O SEGUNDO kev.
//}//FECHA O FOR DO LOOPING DAS AMOSTRAS

//for(kev=0;kev<nev;kev++)
//{//ABRE O ERRO
//DETERMINACAO DO ERRO
//printf("\n\n");
////////////////////////////////////////////////////////////////////
if(!met || met==2)
{//ABRE O met 2
for(kev=0;kev<nev;kev++)
{//ABRE O KEV 2
for(i=0;i<nai;i++)
///     for(j=0;j<nev;j++)
        {
       mat=Th(i,0);
       X(0,i,kev)=I(i,kev)-0;//Th(i,0);
       mat=X(0,i,0);
        }

//CALCULO DE A

//printf("\n\n");
for(i=0;i<nai;i++)
///      for(j=0;j<eai;j++)
       {
//      printf("X(0,%d,%d)=%f",i,kev,X(0,i,kev));
        A(0,i,kev)=(1/(1+exp(-X(0,i,kev))));
        mat=A(0,i,kev);
//      printf("  A(0,%d,%d)=%f",i,kev,A(0,i,kev));
//      printf("\n");
       }
//tt=getche();

//CALCULO DE B
```

```
//printf("\n");
///for(j=0;j<eai;j++)
    for(aux=0,z=0;z<nah;z++)
       {
        for(i=0;i<nai;i++)
                  {
//printf("  WA(%d,%d,%d)=%f\n",i,z,kev,WA(i,z));
//printf("  A(0,%d,%d)=%f",i,kev,A(0,i,kev));
                   aux+=WA(i,z)*A(0,i,kev);
//                 printf("  aux=%f\n",aux);
                  }
                 mat=Th(z,1);
                 X(1,z,kev)=aux+1;//Th(z,1);
                 mat=X(1,z,kev);
//               printf("X(1,%d,%d)=%f",z,kev,X(1,z,kev));
//               printf("\n");
                 aux=0;
         }
//tt=getche();

//clncjcr();
//printf("\n\n");
for(i=0;i<nah;i++)
///     for(j=0;j<eai;j++)
       {
//     printf("X(1,%d,%d)=%f",i,kev,X(1,i,kev));
       A(1,i,kev)=(1/(1+exp(-X(1,i,kev))));
       mat=A(1,i,kev);
//     printf("   B(%d,%d,%d)=%f",k,i,kev,A(1,i,kev));
//     printf("\n");
       }
//tt=getche();

//CALCULO DE C

aux=0;
//clrscr();
/*
for(j=0,i=0;i<nad;i++)
   {
    for(z=0;z<nah;z++)
       {
//      printf("  WB(%d,%d)=%f",z,i,WB(z,i));
       }
//     printf("\n");
    }
*/
//printf("\n\n");
///for(j=0;j<ead;j++)
for(aux=0,i=0;i<nad;i++)
   {
    for(z=0;z<nah;z++)
             {
//printf("WB(%d,%d)=%f",z,i,WB(z,i));
//printf("  A(1,%d,%d)=%f",z,kev,A(1,z,kev));
             aux+=WB(z,i)*A(1,z,kev);
//           printf("  aux=%f\n",aux);
            }
             mat=Th(i,2);
             X(2,i,kev)=aux+1;//Th(i,2);
             mat=X(2,i,kev);
             printf("X(2,%d,%d)=%f",i,kev,X(2,i,kev));
             aux=0;
//           printf("\n");
       }
```

```
//tt=getche();
//printf("\n\n");
for(j=0,i=0;i<nad;i++)
//     for(j=0;j<nev;j++)
       {
        A(2,i,kev)=(1/(1+exp(-X(2,i,kev))));
        mat=A(2,i,kev);
//      printf("C(2,%d,%d)=%f\n",i,kev,A(2,i,kev));
       }

for(j=0,i=0;i<nad;i++)
///     for(j=0;j<ead;j++)
        if(kev==nev-1)
       {
//      printf("A(2,%d,%d)=%f",i,kev,A(2,i,kev));
//      printf("  D(%d,%d)=%f",i,kev,D(i,kev));
        mat=A(2,i,kev); ant=D(i,kev);
        DIF2(i,kev)=fabs(fabs(A(2,i,kev)*DM)-fabs(D(i,kev)*DM));
        Ep2+=DIF2(i,kev)*DIF2(i,kev);
        printf("  DIF2(%d,%d)=%f\n",i,kev,DIF2(i,kev));
       }
        if(kev==nev-1)
        Ep2*=0.5;
}//FECHA O KEV 2
}//FECHA O met 2
////////////////////////////////////////////////////////////////////////

}//FECHA O LOOPING DAS AMOSTRAS

if((DM>1) && (kev==(nev)))
{
if(!met || met==2)
Ep2=(sqrt(Ep2)/(nad*nev))*DM;
Ep1=(sqrt(Ep1)/(nad*nev))*DM;
}

if((DM<1) && (kev==(nev)))
{
if(!met || met==2)
Ep2=sqrt(Ep2)/(nad*nev);
Ep1=sqrt(Ep1)/(nad*nev);
}

if((Ep2>Ep1) && (met==0 || met==2) && (kev==nev))
mi*=BETA;
if((Ep2<Ep1) && (met==0 || met==2) && (kev==nev))
mi/=BETA;

if((met==0 || met==2) && v==nev)
{//ABRE O ITER
for(z=0,i=0;i<nai;i++)
    {
    for(j=0;j<nah;j++)
       {
//printf("  WAP(%d,%d)=%f",i,j,WAP(i,j));
//printf("  WA(%d,%d)=%f",i,j,WA(i,j));
//printf("  Deltawa(0,%d,%d)=%f",i,j,Deltawa(0,i,j));
        if(Ep2>Ep1)
        {
//      t=WAP(i,j);
        WA(i,j)-=Deltawa(0,i,j);
////    printf("  WA(-%d,%d)=%f\n",i,j,WA(i,j));
        }
        if(Ep2<Ep1)
        WAP(i,j)=WA(i,j);
```

```
//printf("  WA(%d,%d)=%f\n",i,j,WA(i,j));
      }
    }
//tt=getche();
for(z=0,i=0;i<nah;i++)
    {
    for(j=0;j<nad;j++)
       {
//printf("WBP(%d,%d)=%f",i,j,WBP(i,j));
//printf("  Deltawb(1,%d,%d)=%f",i,j,Deltawb(1,i,j));
        if(Ep2>Ep1)
        {
//      t=WBP(i,j);
        WB(i,j)-=Deltawb(1,i,j);
////    printf("  WB(-%d,%d)=%f\n",i,j,WB(i,j));
        }
        if(Ep2<Ep1)
        {
        WBP(i,j)=WB(i,j);
//printf("  WB(%d,%d)=%f\n",i,j,WB(i,j));
        }
       }
     }
//tt=getche();
}//FECHA O ITER

}//FECHA O if3
//mat=Ep2;
//}//PROVISORIO PARA FECHAR O LOOPING DAS AMOSTRAS
/*
if(DM>1 && Ep2<Ep1)
{
Ep2=(sqrt(Ep2)/(nad*nev))*DM;
if(!met)
Ep1=(sqrt(Ep1)/(nad*nev))*DM;
}
if(DM<1 && Ep2<Ep1)
{
Ep2=sqrt(Ep2)/(nad*nev);
if(!met)
Ep1=sqrt(Ep1)/(nad*nev);
}
*/
/*
if(((Ep2<Ep1) && (met==0)) || (met==1))
//if((Ep2<Ep1) && (met==2))
{
if(met==1)
{
if(cont<100)
{
printf("BP(i:%d)>DQ(%d.0%d):%0.10f",nah,(cont1-1),cont,Ep1);//Ep1 em MAA
if(Ept<Ep1)
printf(" *");
//printf("  MDA:%0.10f",MAA);
printf("  MDA:%0.10f",MAA*DM);
//if(D(ptx,pty))
printf("  PA(%.2f%)",MAA*100);
//else
//printf("  PA(-)");
if(MAAp<MAA)
printf(" *");
printf("\n");
}
else if(cont>99)
{
```

```
printf("BP(i:%d)>DQ(%d.%d):%0.10f",nah,(cont1-1),cont,Ep1);//Ep1 em MAA
if(Ept<Ep1)
printf(" *");
//printf("  MDA:%0.10f",MAA);
printf("  MDA:%0.10f",MAA*DM);
printf("  PA(%.2f%)",MAA*100);
if(MAAp<MAA)
printf(" *");
printf("\n");
}
}
if(!met)
{
printf("MARQUARDT-LEVENBERG>>DIFER(%d.%d)=%0.10f",(cont1-1),cont,Ep2);
printf("  mi=%.12f\n",mi);
}
if(met==2)
{
printf("QUASE-NEWTON>>>DIFER(%d.%d)=%0.10f",(cont1-1),cont,Ep2);
}
cont++;

if(cont==999)
{
cont=0;
cont1++;
}
iter++;
}

if((Ep2<Ep1) && met==2)
{
printf("QUASE-NEWTON>>>DIFER(%d.%d)=%0.10f",(cont1-1),cont,Ep2);
cont++;
if(cont==999)
{
cont=0;
cont1++;
}
}

v=0;
if(met==1)
{
Ept=Ep1;
MAAp=MAA;
}
if((!met) || (met==2))
Ept=Ep2;
Ep1=0;
Ep2=0;
ptx=0;
ptx=MAA*DM;
//if(ptx==0.08999999)
//tt=getche();
/*
if(MAA*DM<0.4)
{//AAAA
for(EPF=0,k=0;k<nev;k++)
{
for(i=0;i<nad;i++)
///     for(j=0;j<nev;j++)
        {
//       A(2,i,j)=(1/(1+exp(-X(2,i,j))));
         ant=A(2,i,k); mat=DM;
//       A(2,i,k)*=DM;
```

```
        printf("A(2,%d,%d)=%f",i,k,A(2,i,k));
        printf("  D(%d,%d)=%f",i,k,D(i,k));
        printf("  DIF=%f\n",fabs(fabs(A(2,i,k))-fabs(D(i,k))));
//      printf("  C(%d)=%f\n",k,A(2,i,k)*DM);
EPF+=(fabs(fabs(A(2,i,k)*DM)-fabs(D(i,k)*DM)))*(fabs(fabs(A(2,i,k)*DM)-
fabs(D(i,k)*DM)));
//      printf("DM=%f",DM);
       }
        printf("\n");
        tt=getche();
}
}//AAAA
*/

if((Ep2<Ep1) && ((met==0) || (met==1)))
//if((Ep2<Ep1) && (met==2))
{ //1
if(met==1)
{//2
if(cont<100)
{//3
printf("BP(i:%d)>DQ(%d.0%d):%0.10f",nah,(cont1-1),cont,Ep1);//Ep1 em MAA
if(Ept<Ep1)
printf(" *");
//printf("  MDA:%0.10f",MAA);
printf("  MDA[%d::%d]:%0.10f",b+1,marc_abs+1,DMAX);
//if(D(ptx,pty))
printf("  PA(%.2f%)",MDAp);
//else
//printf("  PA(-)");
if(MAAp<MDAp)
printf(" *");
printf("\n");
}//3
else if(cont>99)
{//4
printf("BP(i:%d)>DQ(%d.%d):%0.10f",nah,(cont1-1),cont,Ep1);//Ep1 em MAA
if(Ept<Ep1)
printf(" *");
//printf("  MDA:%0.10f",MAA);
printf("  MDA[%d::%d]:%0.10f",b+1,marc_abs+1,DMAX);
printf("  PA(%.2f%)",MDAp);
if(MAAp<MDAp)
printf(" *");
printf("\n");
}//4
}//2
if(!met && (Ep2<Ep1aux))
{//5
printf("MARQUARDT-LEVENBERG>>DIFER(%d.%d)=%0.10f",(cont1-1),cont,Ep2);
printf("  mi=%.12f\n",mi);
Ep1aux=Ep2aux;
Ep2aux=1;
}//5
if(met==2)
{//6
printf("QUASE-NEWTON>>>DIFER(%d.%d)=%0.10f",(cont1-1),cont,Ep2);
}//6
cont++;

if(cont==999)
{//7
cont=0;
cont1++;
}//7
iter++;
```

```
}//1

if((Ep2<Ep1) && met==2)
{
printf("QUASE-NEWTON>>>DIFER(%d. %d)=%0.10f",(cont1-1),cont,Ep2);
cont++;
if(cont==999)
{
cont=0;
cont1++;
}
}

v=0;
if(met==1)
{
Ept=Ep1;
MAAp=MDAp;//MAA;
}
if((!met) || (met==2))
Ept=Ep2;
Ep1=0;
Ep2=0;
ptx=0;
if(met==1)
ptx=MDAp;//MAA*DM;
if(!met)
ptx=Ept;

}//FECHA O do
while(((Ept>tol) && (MDAp>0.05)) && cont1<30000);//MAA*DM
//while (cont1<100 && MAA*DM>0.09);

printf("\n\nVALORES DE SAIDA\n\n");
//printf("MAIOR DIFERENCA ABSOLUTA=%f\n\n",Ept);

for(EPF=0,k=0;k<nev;k++)
{
for(i=0;i<nad;i++)
///     for(j=0;j<nev;j++)
       {
//      A(2,i,j)=(1/(1+exp(-X(2,i,j))));
        ant=A(2,i,k); mat=DM;
//      A(2,i,k)*=DM;
        printf("A(2,%d,%d)=%f",i,k,A(2,i,k));
        printf("  D(%d,%d)=%f",i,k,D(i,k));
        if(fabs((D(i,k)-A(2,i,k))<0.999*DMAX)) // && fabs(D(i,k)-A(2,i,k)>0.00001))
        printf("  DM= OK!       ");
else if(fabs(((D(i,k)-A(2,i,k))>0.999*DMAX) || ((D(i,k)-A(2,i,k))==DMAX)))
        printf("  DM=%f(*)",fabs(D(i,k)-A(2,i,k)));
if (DM>1)
        printf("  C(%d)=%f\n",k,A(2,i,k)*DM);
else if(DM<1 || DM==1)
     printf("  C(%d)=%f\n",k,A(2,i,k));
EPF+=(fabs(fabs(A(2,i,k)*DM)-fabs(D(i,k)*DM)))*(fabs(fabs(A(2,i,k)*DM)-
fabs(D(i,k)*DM)));
//      printf("DM=%f",DM);
       }
        printf("\n");
        tt=getche();
}

printf("ULTIMA DIFERENCA QUADRATICA:%f\n",EPF*0.5);
printf("ALFA:%f     ETA:%f     TOL:%f",ALFA,ETA,tol);
tt=getche();
```

```
printf("%d",fprintf(fp,"%d %d %d %d",nai,nah,nad,nev));

printf("%d",fprintf(fp,"  %f",DM));

printf("%d",fprintf(fp,"  %f",IM));

for(i=0;i<nai;i++)
printf("%d",fprintf(fp,"  %f",COEF(i,0)));


for(j=0,i=0;i<nai;i++)
printf("%d",fprintf(fp,"  %f",ILD(i,j)));
for(j=0,i=0;i<nai;i++)
printf("%d",fprintf(fp,"  %f",ILmin(i,j)));
/*
printf("%d",fprintf(fp,"\n"));
///for(k=0;k<nev;k++)
    for(k=0,i=0;i<nai;i++)
       {
       for(j=0;j<nah;j++)
          {
printf("%d",fprintf(fp,"  %f",WA(i,j)));
          }
printf("%d",fprintf(fp,"\n"));
       }
*/
///for(k=0;k<nev;k++)
for(k=0,i=0;i<nah+1;i++)
       {
       for(j=0;j<nad;j++)
          {
printf("%d",fprintf(fp,"  %f",WB(i,j)));
          }
printf("%d",fprintf(fp,"\n"));
       }

for(i=0;i<nah;i++)
    {
    for(j=0;j<nai;j++)
       {
//printf("Cij(%d,%d):  %f",i,j,Cij(i,j,0));
printf("%d",fprintf(fp,"  %f",Cij(i,j,0)));
       }
//printf("\n");
printf("%d",fprintf(fp,"\n"));
    }

//tt=getche();

fclose(fp);

free(ii);free(il);free(d);free(dl);free(x);free(th);free(a);
free(aaa);free(wa);free(wap);free(wb);free(wbp);free(dif1);
free(dif2);free(der2);free(der3);free(del2);free(del3);free(delt2);
free(delt3);free(delt1);free(deltaw);free(delwa);free(delwap);free(delwb);
free(delwbp);free(jac);free(jtj);free(ji);free(irp);free(j_1);free(jtjt);
free(mii);free(ex);free(delm2);free(delm3);free(derk2);free(wdel3);

if(met==2)
{
free(dhn_1);free(hn1_1);free(g1);free(g0);free(dg);free(hn_1);free(s);
free(lamb);free(irl);free(r1);free(r2);free(r3);free(r4);free(r5);
}
}//FECHA A SUBROTINA DE TREINAMENTO

void ler_trein(void)
```

```
{ //INICIO DA SUBROTINA DE LEITURA DE ARQUIVO DE DADOS

FILE *fp1;
char s[20];

printf("\nQUAL O NOME DO ARQUIVO DE ENTRADA:");
gets(s);

if((fp1=fopen(s,"r"))==NULL)
{
printf("NAO E POSSIVEL ABRIR ARQUIVO\n");
exit(1);
}

fscanf(fp1,"%d%*c%d%*c%d%*c%f%*c", &nai,&nad,&nev,&tol);

for(i=0;i<nai;i++)
    for(j=0;j<nev;j++)
       {
        fscanf(fp1,"%f%*c",&I[i][j]);
//      printf("%f\n",I[i][j]);
//      scanf("%c");
       }

for(i=0;i<nad;i++)
    for(j=0;j<nev;j++)
       {
        fscanf(fp1,"%f%*c",&D[i][j]);
//      printf("%f\n",D[i][j]);
//      scanf("%c");
       }
} //FIM DA SUBROTINA DE LEITURA DE ARQUIVO DE DADOS

void ler_inf(void)
{//INICIO DA FUNCAO DE LEITURA DE ARQUIVO
FILE *fp3;
char s[20];

//float WA[20][20];
//float WB[20][20];
//float I[20][20];
//float IL[20][1];
//float ILmin[20][1];

wa=TM(nai*nah);
wb=TM(nah*nad);

printf("\nQUAL O ARQUIVO DE PESOS TREINADOS:");
gets(s);

if((fp3=fopen(s,"r"))==NULL)
{
printf("NAO E POSSIVEL ABRIR ARQUIVO\n");
exit(1);
}
fscanf(fp3,"%d%*c%d%*c%d%*c%d%*c%f%*c%f*c",&nai,&nah,&nad,&nev,&DM,&IM);

for(i=0;i<nai;i++)
    {
    fscanf(fp3,"%f%*c",&COEF[i][0]);
//    printf("COEF(%d,0):%f\n",i,COEF[i][0]);
    }

for(j=0,i=0;i<nai;i++)
       {
```

```
      fscanf(fp3,"%f%*c",&IL[i][j]);
//      printf("IL(%d,%d):%f\n",i,j,IL[i][j]);
      }

for(j=0,i=0;i<nai;i++)
      {
      fscanf(fp3,"%f%*c",&ILmin[i][j]);
//      printf("ILmin(%d,%d):%f\n",i,j,ILmin[i][j]);
      }

/*for(i=0;i<nai;i++)
  {
    for(j=0;j<nah;j++)
    {
fscanf(fp3,"%f%*c",&WA[i][j]);
//printf("  WA(%d,%d):%f",i,j,WA[i][j]);
    }
//printf("\n");
  }
*/
for(i=0;i<nah+1;i++)
   {
    for(j=0;j<nad;j++)
    {
fscanf(fp3,"%f%*c",&WB[i][j]);
//printf("  WB(%d,%d):%f",i,j,WB[i][j]);
    }
//    printf("\n");
   }

for(i=0;i<nah;i++)
   {
    for(j=0;j<nai;j++)
    {
fscanf(fp3,"%f%*c",&Cij[i][j]);
//printf("  Cij(%d,%d):%f",i,j,Cij[i][j]);
    }
//    printf("\n");
   }

}//FIM DA FUNCAO DE LEITURA DE ARQUIVO

char menu(void)
{ //INICIO DA SUBROTINA DE OPCOES
char ch;
do
{
//clrscr();
printf("******************************************************\n");
printf("INICIO DO PROGRAMA- ESCOLHA UMA OPCAO ABAIXO\n");
printf("******************************************************\n\n\n");
printf("Tecle (D) para determinar a saida de uma rede\n");
printf("Tecle (L) para ler um arquivo de dados para treinamento\n");
printf("Tecle (I) para ler um arquivo de dados para inferenciamento\n");
printf("Tecle (T) para treinar uma rede nova\n");
printf("Tecle (S) para sair do programa\n");
ch=toupper(getche());
}
while(ch!='D'&&ch!='L'&&ch!='I'&&ch!='T');
printf("\n");
return ch;
} //FIM DA SUBROTINA DE OPCOES

////////////////////////////////////////////////////////////////////////
//SUBROTINAS
////////////////////////////////////////////////////////////////////////
```

```
int lu(int rf,int *irf,float *p1f)

{
  int r1 = rf-1;
  register   int i,j,k;
  float a1,a2;double fabs();

  for (irf[r1]=r1,k=0,  i=0  ; i<r1 ; i++) {
       for ( j = ( k=i)  +1 ; j<rf ; j++)
              if (fabs( P1f(j,i)) > fabs( P1f(k,i))) k=j;
      if ( k != i ) for ( j=i ; j<rf ; j++) {
              a1= P1f(i,j) ; P1f(i,j) = P1f(k,j) ; P1f(k,j) = a1 ;
          }
  if ( !(a2 = P1f(i,i), a2 )) return (-1);

  for( irf[i]=k, j=i+1; j<rf; j++)
     {
     a1=  ( P1f(j,i) /= -a2 );
     for ( k = i+1 ; k<rf ; k++) P1f(j,k) += a1 * P1f(i,k) ;
     }
  }
   if( !P1f(r1,r1) ) return (-1) ;
  return(0);
}

void backs (int rf ,int rs, int *irf, float *p1f, float *uf )
{
  register int i, j, k;
  int r1 = rf- 1;
  float a1,mat,ant;
  for ( i = 0; i < r1; i++ ) {
       if ( ( j = irf[i] ) != i ) for ( k = 0; k < rs; k++ ){
             a1 = Uf(i,k); Uf(i,k) = Uf(j,k); Uf(j,k) = a1;
              }
       for ( j = i + 1; j < rf; j++ )
             for ( k = 0; k < rs; k++ ) Uf(j,k) += P1f(j,i) * Uf(i,k);
      }
        ant=P1f(r1,r1);
        mat=Uf(r1,k);
        for ( k = 0; k < rs; k++ ) Uf(r1,k) /= P1f(r1,r1);
             for ( i = r1 - 1; i >= 0; i-- ) {
       for ( a1 = P1f(i,i), j = i + 1; j < rf; j++ )
              for ( k = 0; k < rs; k++ ) Uf(i,k) -= P1f(i,j) * Uf(j,k);
       for ( k = 0; k < rs; k++ )
                                  {
                                 Uf(i,k) /= a1;
                                 mat=Uf(i,k);
                                }
      }mat+=mat;ant+=ant;
    }

 void determinar(void)
 {//INICIO DA FUNCAO DE DETERMINACAO DOS VALORES INFERENCIADOS
 FILE *fp2;
 char v[20];
 int bin,nmn;

 nmn=nai+nah+nad;
 x=TMD(3*nmn*nev); //TMD
 a=TMD(3*nmn*nev);//TMD
 th=TM(3*nai);
 ii=TM(nai*nev);
 //wad=TM(nai*nah);
 wbd=TMD((nah+1)*nad);
 x_cij=TM(nah*nai*nev);
```

```
n_eucl=TM(nah*nev);
cij=TM(nah*nai*nev);
fi=TM(nah*nev);

sigma=0.0125;

printf("QUAL O ARQUIVO COM OS DADOS A SEREM AJUSTADOS:");
gets(v);
//clrscr();
if((fp2=fopen(v,"r"))==NULL)
{
printf("NAO E POSSIVEL ABRIR O ARQUIVO\n");
exit(1);
}

for(j=0,i=0;i<nai;i++)
//    for(j=0;j<nev;j++)
       {
fscanf(fp2,"%f%*c",&ID[i][0]);
//printf("  ID(%d,%d)=%f\n",i,j,ID[i][j]);
       }
fscanf(fp2,"%d%*c",&bin);

//CALCULO DOS VALORES DE SAIDA
/*
printf("\n");
for(i=0;i<nai;i++)
{
    for(j=0;j<nah;j++)
       {
//     WAD(i,j)=WA[i][j];
       COEF_D(i,0)=COEF[i][0];
//     printf("WA(%d,%d)=%f\n",i,j,WAD(i,j));
       }
//     printf("COEF_D(%d,0):%f\n",i,COEF_D(i,0));
}
*/
for(j=0,i=0;i<nai;i++)
    {
//     printf("ID(%d,0):%f""  ""IL(%d,0):%f",i,ID[i][j],i,IL[i][j]);
//     printf("ILmin(%d,0):%f\n",i,ILmin[i][0]);
//     printf("COEF(%d,0)=%f",i,COEF_D(i,0));
     if(IM>1)
     I(i,j)=(COEF[i][0]*(ID[i][0]-ILmin[i][0]));
        else if(IM<1 || IM==1)
     I(i,j)=ID[i][0];
//     printf("  I(%d,0):%f\n",i,I(i,0));
//     tt=getche();
    }

//printf("\n");
for(i=0;i<nah+1;i++)
    for(j=0;j<nad;j++)
       {
        WBD(i,j)=WB[i][j];
//      printf("WB(%d,%d)=%f\n",i,j,WBD(i,j));
       }

for(i=0;i<nah;i++)
    for(j=0;j<nai;j++)
       {
        Cij(i,j,0)=Cij[i][j];
//printf("Cij(%d,%d,0)=%f\n",i,j,Cij(i,j,0));
       }

for(i=0;i<nah;i++)
```

```
   {
    for(j=0;j<nai;j++)
       {
       X_Cij(i,j,0)=I(j,0)-Cij(i,j,0);
//     printf("I(%d,%d):%f""  ""Cij(%d,%d,%d):%f"" ""X(%d,%d,%d):%f\n",j,0,
//     I(j,0),i,j,0,Cij(i,j,0),i,j,0,X_Cij(i,j,0));
//     tt=getche();
       }
//     printf("\n");
    }
//CALCULO DE N_Euc (NORMA EUCLIDIANA)

for(i=0;i<nah;i++)
       {
    for(aux=0,j=0;j<nai;j++)
        aux+=(X_Cij(i,j,0)*X_Cij(i,j,0));
        N_Eucl(i,0)=aux;
//      printf("N_Eucl(%d,%d)=%f\n",i,0,N_Eucl(i,0));
        }

//CALCULO DE Fi

//printf("\n\n");
for(i=0;i<nah;i++)
       {
//      printf("N_Eucl(%d,%d)=%f",i,kev,N_Eucl(i,kev));
        mat=pow(sigma,2);
        Fi(i,0)=exp(-N_Eucl(i,0)/(2*pow(sigma,2)));
        mat=Fi(i,0);
//      printf("  Fi(%d,%d)=%f",i,0,Fi(i,0));
//      tt=getche();
//      printf("\n");
       }
//tt=getche();

//CALCULO DE C
//printf("\n\n");
aux=0;
for(i=0;i<nad;i++)
   {
    for(z=0;z<nah+1;z++)
          {
           if(!z)
             aux=WBD(z,i)*1;
           else if(z)

            {
//printf("  WBl(%d,%d)=%f",z,i,WBD(z,i));
//printf("  Fi(%d,%d)=%f",z-1,0,Fi(z-1,0));
            aux+=WBD(z,i)*Fi(z-1,0);
//          printf("  aux=%f\n",aux);
           }
          }
            X(2,i,0)=aux;
//          printf("X(2,%d,0):%f",i,X(2,i,0));
            aux=0;
//          printf("\n");
       }

//printf("\n\n");
for(j=0,i=0;i<nad;i++)
///    for(j=0;j<nev;j++)
       {
        A(2,i,0)=(1/(1+exp(-X(2,i,0))));
        printf("C(2,%d,%d)=%f\n",i,0,A(2,i,0));
        tt=getche();
```

```
      }
////////////////////////////////////////////////////////////////////////////////

//printf("\n\n");

/*
for(j=0,i=0;i<nad;i++)
///     for(j=0;j<ead;j++)
       {
        mat=A(2,i,0); ant=D(i,0);
        if(bom)
        {
        printf("A(2,%d,%d)=%f",i,0,A(2,i,0));
        printf("  D(%d,%d)=%f",i,0,D(i,0));
        }
        DIF1A(i,0)=fabs(fabs(A(2,i,0)*1)-fabs(D(i,0)*1));
        Daux(i,0)=fabs(fabs(A(2,i,0)*1)-fabs(D(i,0)*1));
        if(bom)
        printf("  DIF1A(%d,%d)=%f\n",i,0,DIF1A(i,0));
       }


*/
/*
for(j=0,i=0;i<nai;i++)
    {
//      printf("ID(%d,0):%f""  ""IL(%d,0):%f",i,ID[i][j],i,IL[i][j]);
//      printf("ILmin(%d,0):%f\n",i,ILmin[i][0]);
//      printf("COEF(%d,0)=%f",i,COEF_D(i,0));
     I(i,j)=(COEF[i][0]*(ID[i][0]-ILmin[i][0]));
//      printf("  I(%d,0):%f\n",i,I(i,0));
//      tt=getche();
    }

for(j=0;j<nai;j++)
     Th(j,0)=0;
for(j=0;j<nah;j++)
     Th(j,1)=1;
for(j=0;j<nad;j++)
     Th(j,2)=1;

for(j=0,i=0;i<nai;i++)
//    for(j=0;j<nev;j++)
       {
       X(0,i,j)=I(i,j)-Th(i,0);
//printf("X(0,%d,%d)=%f\n",i,j,X(0,i,j));
       }
//CALCULO DE A

//clrscr();
for(j=0,i=0;i<nai;i++)
//     for(j=0;j<nev;j++)
       {
        A(0,i,j)=(1/(1+exp(-X(0,i,j))));
//      printf("A(0,%d,%d)=%f",i,j,A(0,i,j));
//      printf("\n");
       }
//tt=getche();

//CALCULO DE B

//printf("\n");
//for(j=0;j<nev;j++)
    for(aux=0,j=0,z=0;z<nah;z++)
       {
```

```
        for(i=0;i<nai;i++)
                    {
//printf("WA(%d,%d)=%f",i,z,WAD(i,z));
//printf("  A(0,%d,%d)=%f",i,j,A(0,i,j));
                     aux+=WAD(i,z)*A(0,i,j);
//                   printf("  aux=%f\n",aux);
                    }

                   X(1,z,j)=aux+Th(z,1);
//                 printf("X(1,%d,%d)=%f",z,j,X(1,z,j));
//                 printf("\n");
                   aux=0;
          }
//tt=getche();

//clrscr();
for(j=0,i=0;i<nah;i++)
//     for(j=0;j<nev;j++)
       {
//     printf("X(1,%d,%d)=%f",i,j,X(1,i,j));
       A(1,i,j)=(1/(1+exp(-X(1,i,j))));
//     printf("   B(1,%d,%d)=%f",i,j,A(1,i,j));
//     printf("\n");
       }
//tt=getche();

//CALCULO DE C

aux=0;
//clrscr();

for(j=0,i=0;i<nai;i++)
   {
    for(z=0;z<nah;z++)
       {
        printf("  WA(%d,%d):%f",i,z,WAD(i,z));
       }
       printf("\n");
    }


for(j=0,i=0;i<nad;i++)
   {
    for(z=0;z<nah;z++)
       {
        printf("WB(%d,%d)=%f",z,i,WBD(z,i));
       }
       printf("\n");
    }


//printf("\n\n");
//for(j=0;j<nev;j++)
for(aux=0,j=0,i=0;i<nad;i++)
   {
    for(z=0;z<nah;z++)
            {
//printf("WBD(%d,%d)=%f",z,i,WBD(z,i));
//printf("  A(1,%d,%d)=%f",z,j,A(1,z,j));
             aux+=WBD(z,i)*A(1,z,j);
//           printf("  aux=%f\n",aux);
            }
             X(2,i,0)=aux+Th(i,2);
//           printf("X(2,%d,0)=%f",i,X(2,i,0));
             aux=0;
//           printf("\n");
```

```
      }

//printf("\n\n");
for(j=0,i=0;i<nad;i++)
//    for(j=0;j<nev;j++)
      {
       A(2,i,j)=(1/(1+exp(-X(2,i,j))));
//     printf("C(%d)=%f\n",i,A(2,i,j));
      }
*/

printf("\n\n\n\n");
if(bin)
{
for(j=0,i=0;i<nad;i++)
{
if(A(2,i,j)<0.3)
A(2,i,j)=0;
if(A(2,i,j)>0.7)
A(2,i,j)=1;
if(A(2,i,j)==1)
printf("R(%d)=%f*\n",i,A(2,i,j));
else if(A(2,i,j)!=1)
printf("R(%d)=%f\n",i,A(2,i,j));
}
}
if(!bin)
for(j=0,i=0;i<nad;i++)
printf("R(%d)=%f\n",i,A(2,i,j));

tt=getche();
}//FIM DA FUNCAO DE DETERMINACAO DOS VALORES INFERENCIADOS
```