DETECÇÃO E DIAGNÓSTICO DE FALHAS EM ROBÔS MANIPULADORES VIA REDES NEURAIS ARTIFICIAIS

Renato Tinós

Dissertação apresentada à Escola de Engenharia de São Carlos da Universidade de São Paulo, como parte dos requisitos para obtenção do título de Mestre em Engenharia Elétrica

Orientador:
Prof. Dr. Marco Henrique Terra

São Carlos
1999

Dedico este trabalho aos meus pais, Denir e Elisabet, por terem me permitido chegar até aqui e pelos exemplos de perseverança e simplicidade.

Agradecimentos

Ao Prof. Dr. Marco Henrique Terra pela orientação, amizade e pela confiança depositada para a realização deste trabalho.

À Lúcia Maria Santos pela dedicação, paciência e, sobretudo, companheirismo durante a realização deste trabalho. O caminho seria muito mais tortuoso sem você.

A todos os amigos que me acompanharam nesta jornada e ainda estão presentes. Além disto, agradeço em especial aos colegas Ricardo Sovat, Maria Cristina, Guilherme Barreto, Arthur Plínio, Marcelo Rosa e Hélio D'arbo pelas valiosas discussões e sugestões.

Aos Professores Aluízio F. R. Araújo da EESC - USP, Marcel Bergerman do CTI - Campinas, Oswaldo L. V. Costa da Escola Politécnica - USP e Walmir M. Caminhas da UFMG pelas críticas e sugestões.

À minha família, em especial, ao Fábio, à Adriana, ao Luís Filipe e à Marisa.

Aos professores e funcionários do Depto. de Engenharia Elétrica que sempre estiveram dispostos a colaborar.

Ao CNPq pelo suporte financeiro fornecido durante a realização deste trabalho, sem o qual o mesmo não seria possível.

Sumário

Lista de Abreviaturas

DDF	- Detecção e Diagnóstico de Falhas
IA	- Inteligência Artificial
RNA	- Redes Neurais Artificiais
MLP	- *MultiLayer Perceptron*
RBF	- *Radial Basis Function*
GCV	- *Generalized Cross Validation*
GRR	- *Global Ridge Regression*
LRR	- *Local Ridge Regression*
FS	- *Forward Selection*
MAOK	- Mapa Auto-Organizável de Kohonen

Notação Geral

a, B	letras em itálico representam escalares.
a, **b**	letras minúsculas em negrito representam vetores: $\mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix}$.
A, **B**	letras maiúsculas em negrito representam matrizes.
$\mathbf{I}_n$	matriz identidade (n x n).
$\mathbf{A}^{\mathrm{T}}$	transposta de **A**.
$\mathbf{A}^{-1}$	inversa de **A**.
exp	função exponencial: exp (a) = e^{a} .
$\hat{a}$	valor estimado de a.
$\|\mathbf{a}\|$	norma de **a**.
$\langle a \rangle$	expectância de a.
$\dot{a}$	primeira derivada de a.
$\ddot{a}$	segunda derivada de a.
diag (**A**)	matriz diagonal de **A**.
tr (**A**)	traço da matriz **A**.

Símbolos Principais

Detecção e Diagnóstico de Falhas

$\mathbf{d}$	- vetor de distúrbios externos não-correlacionados
$\mathbf{f}(.)$, $\mathbf{g}(.)$	- funções não-lineares do sistema livre de falhas
t	- índice do tempo
$\mathbf{x}$	- vetor de estados do sistema
$\mathbf{u}$	- vetor de controle
$\mathbf{y}$	- saída da planta
Δt	- período amostral
$\boldsymbol{\phi}(.)$	- função de falha
$\hat{\boldsymbol{\phi}}(.)$	- vetor de resíduos

Redes Neurais Artificiais

$\boldsymbol{\xi}$	- vetor de entradas da RNA
$\hat{\boldsymbol{\psi}}$	- vetor de saídas da RNA
$\boldsymbol{\psi}$	- vetor de saídas desejadas da RNA
h_a	- ativação do neurônio a
$\mathbf{H}$	- matriz de projeto
φ_a	- função de ativação do neurônio a
ω_{ba}	- peso entre a saída do neurônio a e a entrada do neurônio b
e	- erro instantâneo
E	- erro quadrático instantâneo
$E_{médio}$	- erro médio quadrático
C	- função de custo
η	- taxa de aprendizagem
δ_a	- fator de ajuste do neurônio a
$\boldsymbol{\mu}_a$	- centro da unidade radial a
$\mathbf{R}$	- matriz que determina o tamanho dos campos receptivos

λ - parâmetro de regularização

$\mathbf{\Lambda}$ - matriz dos parâmetros de regularização individuais

$\mathbf{P}$ - matriz de projeção

$\mathbf{A}^{-1}$ - matriz de variância

$\hat{\sigma}_{GCV}$ - erro GCV

α - função de decaimento que define a taxa de aprendizagem (MAOK)

β - define o espalhamento do ajuste (MAOK)

σ - função de decaimento que define o espalhamento (MAOK)

Robô Manipulador

$\boldsymbol{\theta}$ - vetor de posições das juntas

$\boldsymbol{\tau}$ - vetor de torques

$\mathbf{M}$ - matriz de inércia

$\mathbf{v}$ - vetor dos termos centrífugos e de Coriolis

$\mathbf{g}$ - vetor dos termos gravitacionais

$\mathbf{z}$ - vetor dos termos de fricção

n_g - número de graus de liberdade do manipulador

RESUMO

TINÓS, R. (1999). *Detecção e diagnóstico de falhas em robôs manipuladores via redes neurais artificiais*. São Carlos, 1999. 117p. Dissertação (Mestrado) - Escola de Engenharia de São Carlos, Universidade de São Paulo.

Neste trabalho, um novo enfoque para detecção e diagnóstico de falhas (DDF) em robôs manipuladores é apresentado. Um robô com falhas pode causar sérios danos e pode colocar em risco o pessoal presente no ambiente de trabalho. Geralmente, os pesquisadores têm proposto esquemas de DDF baseados no modelo matemático do sistema. Contudo, erros de modelagem podem ocultar os efeitos das falhas e podem ser uma fonte de alarmes falsos. Aqui, duas redes neurais artificiais são utilizadas em um sistema de DDF para robôs manipuladores. Um *Perceptron* Multicamadas treinado por retropropagação do erro é usado para reproduzir o comportamento dinâmico do manipulador. As saídas do *Perceptron* são comparadas com as variáveis medidas, gerando o vetor de resíduos. Em seguida, uma rede com função de base radial é usada para classificar os resíduos, gerando a isolação das falhas. Quatro algoritmos diferentes são empregados para treinar esta rede. O primeiro utiliza regularização para reduzir a flexibilidade do modelo. O segundo emprega regularização também, mas ao invés de um único termo de penalidade, cada unidade radial tem um regularização individual. O terceiro algoritmo emprega seleção de subconjuntos para selecionar as unidades radiais a partir dos padrões de treinamento. O quarto emprega o Mapa Auto-Organizável de Kohonen para fixar os centros das unidades radiais próximos aos centros dos aglomerados de padrões. Simulações usando um manipulador com dois graus de liberdade e um Puma 560 são apresentadas, demostrando que o sistema consegue detectar e diagnosticar corretamente falhas que ocorrem em conjuntos de padrões não-treinados.

Palavras-chave: Detecção e diagnóstico de falhas, Robôs manipuladores, Redes neurais artificiais, *Perceptron* multicamadas, Rede com função de base radial, Mapa auto-organizável de Kohonen.

ABSTRACT

TINÓS, R. (1999). *Fault detection and diagnosis in robotic manipulators via artificial neural networks*. São Carlos, 1999. 117p. Dissertação (Mestrado) - Escola de Engenharia de São Carlos, Universidade de São Paulo.

In this work, a new approach for fault detection and diagnosis in robotic manipulators is presented. A faulty robot could cause serious damages and put in risk the people involved. Usually, researchers have proposed fault detection and diagnosis schemes based on the mathematical model of the system. However, modeling errors could obscure the fault effects and could be a false alarm source. In this work, two artificial neural networks are employed in a fault detection and diagnosis system to robotic manipulators. A Multilayer Perceptron trained with Backpropagation algorithm is employed to reproduce the robotic manipulator dynamical behavior. The Perceptron outputs are compared with the real measurements, generating the residual vector. A Radial Basis Function Network is utilized to classify the residual vector, generating the fault isolation. Four different algorithms have been employed to train this Network. The first utilizes regularization to reduce the flexibility of the model. The second employs regularization too, but instead of only one penalty term, each radial unit has a individual penalty term. The third employs subset selection to choose the radial units from the training patterns. The forth algorithm employs the Kohonen's Self-organizing Map to fix the radial unit center near to the cluster centers. Simulations employing a two link manipulator and a Puma 560 manipulator are presented, demonstrating that the system can detect and isolate correctly faults that occur in nontrained pattern sets.

Keywords: Fault detection and diagnosis, Robotic manipulators, Artificial neural networks, Multilayer perceptrons, Radial basis function networks, Kohonen's self-organizing map.

Introdução

Com a crescente automatização dos processos industriais, mecanismos que garantam maior segurança e confiabilidade aos equipamentos estão sendo cada vez mais exigidos. É crescente a busca pela minimização das perdas econômicas causadas durante os processos de produção. Entre outras exigências, deseja-se que os componentes inclusos nos processos de produção tenham um bom desempenho e sejam livres de falhas. Eventuais falhas em componentes de um sistema dinâmico inserido no processo industrial podem acarretar perdas de desempenho não aceitáveis, bem como por em risco os equipamentos e o pessoal envolvido. Portanto, sistemas de detecção e diagnóstico de falhas se tornam cada vez mais importantes.

Particularmente na área de robótica, com o crescente uso de robôs em áreas como exploração espacial, medicina e ambientes hostis (por exemplo, instalações nucleares), não somente a detecção e o diagnóstico de falhas se torna importante, como também os sistemas tolerantes a falhas.

Uma falha em um sistema dinâmico pode ser entendida como qualquer tipo de mau funcionamento em seus componentes que leve a uma perda de desempenho não aceitável na realização de determinadas tarefas. Falhas podem ocorrer de modo abrupto, as quais geralmente causam a parada do equipamento afetado, e de modo lento. Estas ocorrem de modo gradual, ocasionando perda de desempenho no sistema e fadiga nos componentes sobrecarregados. Muitas vezes os efeitos destas falhas são encobertos pela ação dos controladores envolvidos, sendo, portanto, difíceis de serem detectadas.

Uma alternativa para a minimização dos efeitos desagradáveis causados pelas falhas é a utilização da chamada redundância física (inclusão de equipamentos, tais como sensores e atuadores redundantes). Entretanto, muitas vezes essa é uma alternativa bastante custosa e pode se tornar inviável, como por exemplo, no caso de falta de espaço físico para a instalação do componente redundante.

Por outro lado, pode-se utilizar sistemas de detecção e diagnóstico de falhas que não utilizam componentes físicos adicionais. As diversas técnicas de detecção e diagnóstico de falhas utilizam processamento de informações das variáveis medidas do sistema sob condições operacionais bem caracterizadas. As técnicas mais utilizadas são aquelas que utilizam o modelo matemático do sistema para reproduzir o seu comportamento dinâmico livre de falhas. Para um determinado vetor das variáveis medidas de entrada do sistema, o modelo matemático deverá gerar um vetor de saídas para ser comparado com o vetor das variáveis medidas da saída do sistema. A diferença entre estes dois vetores é chamada de resíduo. Este resíduo deve ser diferente de zero na ocorrência de uma falha e trará as informações necessárias para a sua identificação.

Contudo, erros de modelagem obscurecem os efeitos das falhas e são uma fonte de alarmes falsos. Técnicas que buscam robustez em relação aos erros de modelagem na geração de resíduos são cada vez mais estudadas. Outro problema, provavelmente mais sério, é que muitos dos sistemas reais não podem ser modelados com precisão suficiente.

Um outro enfoque é aquele em que se empregam técnicas de Inteligência Artificial. Dentro deste enfoque, podem-se destacar os métodos que utilizam sistemas baseados em conhecimento, a lógica nebulosa (*fuzzy logic*) e as redes neurais artificiais. Estas, em particular, têm sido exaustivamente empregadas em uma infinidade de áreas distintas.

O objetivo deste trabalho é propor um sistema de detecção e diagnóstico de falhas baseado em redes neurais artificiais para robôs manipuladores. Na arquitetura do sistema proposto, são utilizadas duas redes neurais artificiais: a primeira tem por objetivo reproduzir o comportamento dinâmico do manipulador e, a segunda tem por função classificar os sinais de resíduo produzidos pela diferença entre as saídas da primeira rede e as variáveis medidas. Dois tipos diferentes de redes neurais artificias são utilizados. Um *perceptron* multicamadas é utilizado para efetuar o mapeamento dinâmico do sistema e uma rede RBF (*radial basis function*) é empregada para a classificação dos padrões faltosos. Com o objetivo de se fazer comparações, esta rede é treinada por quatro métodos diferentes. Um manipulador planar com dois elos rígidos e um manipulador Puma 560 são simulados para testes do sistema de detecção e diagnóstico de falhas via redes neurais artificiais.

Neste trabalho, apresenta-se no Capítulo 1 uma visão geral sobre o problema da detecção e diagnóstico de falhas em sistemas dinâmicos. Primeiramente o conceito de geração de resíduos é apresentado. Tal conceito é utilizado neste trabalho e pode ser empregado em métodos baseados nos modelos matemáticos.

No Capítulo 2, uma introdução ao tema das redes neurais artificiais, enfatizando suas aplicações em mapeamento de funções e classificação de padrões, é feita. Duas redes são apresentadas: o *perceptron* multicamadas com aprendizado por retropropagação do erro e a rede *radial basis function* (RBF). Discute-se, também, a motivação de se empregar tais redes para o sistema de detecção e diagnóstico de falhas. Ainda neste Capítulo, apresenta-se os 4 algoritmos utilizados neste trabalho para o treinamento das redes RBF.

A seguir, no Capítulo 3, uma breve introdução aos robôs manipuladores é feita. São apresentadas também uma análise sobre as possíveis falhas que ocorrem nestes sistemas, os métodos utilizados para a detecção e diagnóstico de falhas e os sistemas com tolerância a falhas.

A arquitetura do sistema baseado em redes neurais artificiais utilizado para a detecção e diagnóstico de falhas é apresentada no Capítulo 4. O Capítulo 5 traz os resultados do sistema de detecção e isolação de falhas proposto utilizando redes neurais aplicado ao manipulador com 2 elos rígidos e ao manipulador Puma 560. Finalmente, o Capítulo 6 traz as discussões sobre o trabalho realizado.

Capítulo 1

Detecção e diagnóstico de falhas em sistemas dinâmicos

Qualquer tipo de alteração no sistema dinâmico que leve a uma perda de desempenho não aceitável na realização de determinadas tarefas pode ser entendida como falha. A categoria de falhas mais importante engloba as alterações na planta e o mau funcionamento de sensores, atuadores e controladores.

Falhas podem ocorrer de modo abrupto ou de modo gradual. Estas últimas são geralmente difíceis de serem detectadas pois seus efeitos não são observados em curto prazo. Além disso, muitas vezes tais efeitos são encobertos pela ação dos controladores. Um exemplo típico é a variação paramétrica em um sistema dinâmico. No entanto, tal qual as falhas que ocorrem de modo abrupto, essas falhas ocasionam perdas de desempenho não aceitáveis e muitas vezes colocam em risco os equipamentos, o ambiente de trabalho e o pessoal envolvido. Tais fatores podem explicar a crescente procura por sistemas que propiciam detecção e diagnóstico de falhas (DDF) de forma rápida e confiável.

Segundo a terminologia geralmente aceita, DDF consiste em [GERTLER, 1988]:

a) **Detecção da falha**, ou seja, a indicação de que algo errado está acontecendo no sistema;
b) **Isolação da falha**, ou seja, a determinação do tipo e local da falha, e
c) **Identificação da falha**, ou seja, determinação do tamanho da falha.[1]

Métodos de DDF podem ser empregados em sistemas de controle que propiciem tolerância a falhas. Tais sistemas de controle podem ser caracterizados por serem robustos e/ou reconfiguráveis [PATTON, 1997]. Um sistema de controle é dito robusto, se retém satisfatoriamente o desempenho na presença de erros de modelagem, ruídos e/ou falhas. O sistema de controle é dito reconfigurável, se a sua estrutura ou seus parâmetros puderem ser

[1]Neste trabalho, o termo DDF é empregado para designar a detecção e a isolação das falhas.

alterados em resposta a falhas. Neste caso, o sistema de controle deve detectar e diagnosticar a falha, modificando posteriormente suas leis para manter um desempenho aceitável [STENGEL, 1991].

Muitos sistemas empregam a chamada redundância física (ou de *hardware*) para obter tolerância a falhas. Na redundância física, alguns componentes do sistema de controle (sensores, atuadores e controladores) são duplicados, ou seja, existem dois componentes para desempenhar a mesma função. Os componentes redundantes são aqueles mais sujeitos a falhas e/ou mais cruciais em relação ao desempenho do sistema.

Uma derivação de tal método é a redundância física em paralelo, na qual dois ou mais conjuntos de sensores, atuadores e controladores, cada qual com capacidade individual de um controle satisfatório, são instalados para a execução da mesma tarefa. Um sistema gerenciador deve comparar os sinais de controle para detectar o conjunto faltoso. Com dois sinais redundantes, um sistema votante pode detectar a existência de um conjunto faltoso mas não pode identificá-lo. Com três sinais redundantes, um sistema votante pode detectar e isolar o conjunto faltoso, selecionando assim, qual não deve ser utilizado para o controle da planta.

Redundância física pode proteger o sistema contra falhas nos componentes do sistema de controle, mas não nos componentes da planta. Pode, também, ser uma alternativa bastante cara e muitas vezes inviável, como no caso de falta de espaço físico para a instalação dos componentes redundantes.

De outro lado surgem sistemas de DDF que não necessitam de instrumentação adicional na planta, utilizando-se do processamento de informação das variáveis medidas. Dentro deste enfoque, os métodos que empregam a chamada redundância analítica [CHOW & WILLSKY, 1984] são os mais conhecidos. Estes métodos utilizam o conceito de geração de resíduos, que será visto na Seção seguinte. Os sistemas de DDF via redundância analítica serão apresentados resumidamente na Seção 1.2. Será visto que erros de modelagem podem comprometer a eficiência de tais sistemas de DDF, levando os pesquisadores a buscarem técnicas robustas. Como será visto na Seção 1.3, técnicas de inteligência artificial podem tornar-se ferramentas interessantes para auxílio da resolução de tal problema.

1.1. CONCEITO DE GERAÇÃO DE RESÍDUOS

Um sistema dinâmico é geralmente constituído por atuadores, planta e sensores, como pode ser visto na Figura 1.

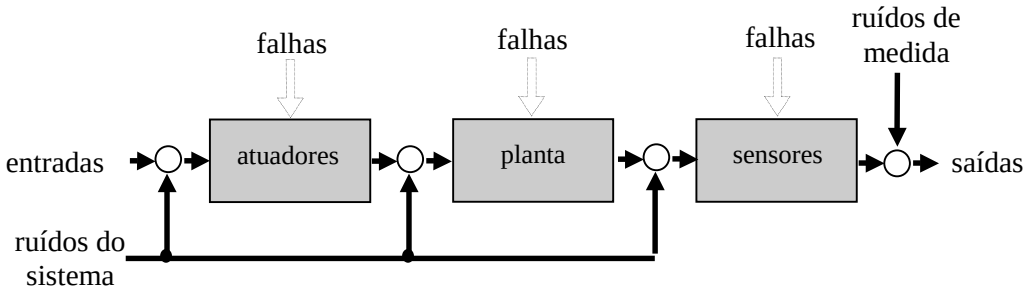

Figura 1. Representação do sistema dinâmico.

A equação de estados e a equação de saídas do sistema dinâmico livre de falhas são dadas por

$$\dot{\mathbf{x}}(t) = \mathbf{f}_t\left(\mathbf{x}(t), \mathbf{u}(t), \mathbf{d}(t)\right) \tag{1}$$

$$\mathbf{y}(t) = \mathbf{g}_t\left(\mathbf{x}(t), \mathbf{u}(t), \mathbf{d}(t)\right) \tag{2}$$

na qual $\mathbf{x}(t)$ é o vetor de estados no tempo t, $\mathbf{y}(t)$ é o vetor de saídas no tempo t, $\mathbf{u}(t)$ é o vetor de controle atual, $\mathbf{d}(t)$ é o vetor de distúrbios externos não-correlacionados e, $\mathbf{f}_t(.)$ e $\mathbf{g}_t(.)$ representam as funções não-lineares do sistema livre de falhas. Discretizando as equações acima, o sistema dinâmico livre de falhas pode ser representado por

$$\mathbf{x}(t + \Delta t) = \mathbf{f}\left(\mathbf{x}(t), \mathbf{u}(t), \mathbf{d}(t)\right) \tag{3}$$

$$\mathbf{y}(t) = \mathbf{g}\left(\mathbf{x}(t), \mathbf{u}(t), \mathbf{d}(t)\right) \tag{4}$$

nas quais Δt é o período amostral e, por abuso de notação, $\mathbf{x}$ e $\mathbf{y}$ representam respectivamente o vetor de estados e o vetor de saídas discretos. A Figura 2 apresenta a representação da equação de estados do sistema livre de falhas. Note que os ruídos do sistema e demais distúrbios externos não-correlacionados são representados pelo vetor $\mathbf{d}(t)$.

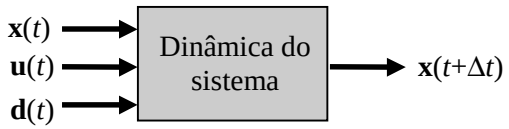

Figura 2. Representação da dinâmica do sistema livre de falhas.

Considerando agora os efeitos das falhas, as equações do sistema podem ser dadas por

$$\mathbf{x}_\phi\left(t+\Delta t\right)=\mathbf{f}_\phi\left(\mathbf{x}(t),\mathbf{u}(t),\mathbf{d}(t)\right) \quad (5)$$

$$\mathbf{y}_\phi\left(t\right)=\mathbf{g}_\phi\left(\mathbf{x}(t),\mathbf{u}(t),\mathbf{d}(t)\right) \quad (6)$$

nas quais $\mathbf{f}_\phi(.)$ e $\mathbf{g}_\phi(.)$ representam as funções não-lineares do sistema na presença de falhas. Note que as falhas podem ser, ou não, entradas aditivas do sistema (ou seja, dependentes apenas do tempo t). O vetor de falhas é aqui definido como a diferença do comportamento dinâmico do sistema na presença de falhas (Eq. 5) e do comportamento dinâmico do sistema livre de falhas (Eq. 3). Assim, o vetor de falhas é dado por

$$\boldsymbol{\phi}\left(t+\Delta t\right)=\mathbf{x}_\phi\left(t+\Delta t\right)-\mathbf{x}\left(t+\Delta t\right)=\mathbf{f}_\phi\left(\mathbf{x}(t),\mathbf{u}(t),\mathbf{d}(t)\right)-\mathbf{f}\left(\mathbf{x}(t),\mathbf{u}(t),\mathbf{d}(t)\right). \quad (7)$$

Note que na ausência de falhas no sistema, $\boldsymbol{\phi}(t+\Delta t)=\mathbf{0}$ e, em caso contrário, $\boldsymbol{\phi}(t+\Delta t)\neq\mathbf{0}$. Geralmente, para cada tipo de falha, $\boldsymbol{\phi}$ tem um comportamento peculiar, o qual é chamado de assinatura da falha. A Figura 3 ilustra o comportamento dinâmico de um sistema com dois estados, x_1 e x_2 , em que ocorre uma falha em t=2. Note que para t<2 o vetor de falhas é nulo.

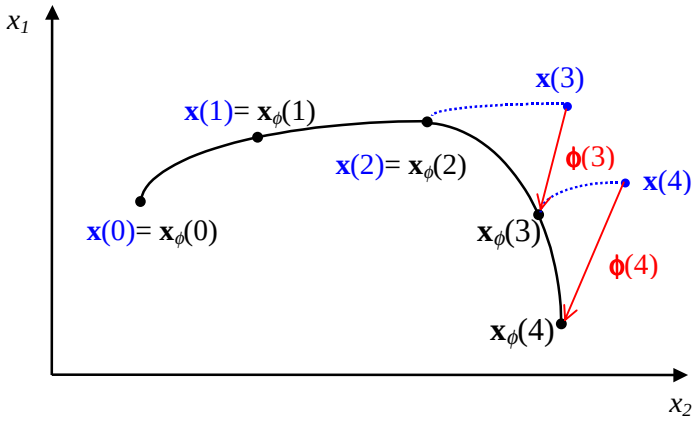

Figura 3. Vetor de falhas e estados de um sistema em que ocorre uma falha em *t*=2.

O princípio básico dos métodos de DDF que utilizam o conceito de geração de resíduos é o emprego de $\boldsymbol{\phi}$ para análise das falhas do sistema. Para isso, o comportamento dinâmico do sistema livre de falhas (dado pela Eq. 3) deve ser reproduzido por uma ferramenta qualquer. Como será visto a seguir, em redundância analítica, o comportamento dinâmico do sistema livre de falhas é reproduzido pelo modelo matemático. Considere que a estimativa do vetor de estados de um sistema livre de falhas dado por uma ferramenta qualquer seja

$$\hat{\mathbf{x}}(t+\Delta t) = \hat{\mathbf{f}}\left(\mathbf{x}(t), \mathbf{u}(t)\right) \tag{8}$$

na qual $\hat{\mathbf{f}}(.)$ é a função não-linear que representa o mapeamento entrada-saída da ferramenta utilizada. Aqui, o vetor de resíduos é definido como

$$\hat{\boldsymbol{\phi}}(t+\Delta t) = \mathbf{x}_{\phi}(t+\Delta t) - \hat{\mathbf{x}}(t+\Delta t) = \mathbf{f}_{\phi}\left(\mathbf{x}(t), \mathbf{u}(t), \mathbf{d}(t)\right) - \hat{\mathbf{f}}\left(\mathbf{x}(t), \mathbf{u}(t)\right). \tag{9}$$

Se a ferramenta utilizada representa exatamente o comportamento dinâmico do sistema livre de falhas e não existem distúrbios externos, o vetor de resíduos é exatamente igual ao vetor de falhas. No entanto, em sistemas reais, tal fato não ocorre. Em tais sistemas sempre haverá diferenças entre o vetor de resíduos e o vetor de falhas, devido aos distúrbios não-correlacionados e/ou erros no mapeamento do comportamento dinâmico. Estas diferenças, como será visto a seguir, podem prejudicar seriamente a detecção e o diagnóstico das falhas. A Figura 4 mostra o vetor de resíduos para um sistema em que ocorre uma falha em t=2. Note que existe um erro no mapeamento do sistema dinâmico.

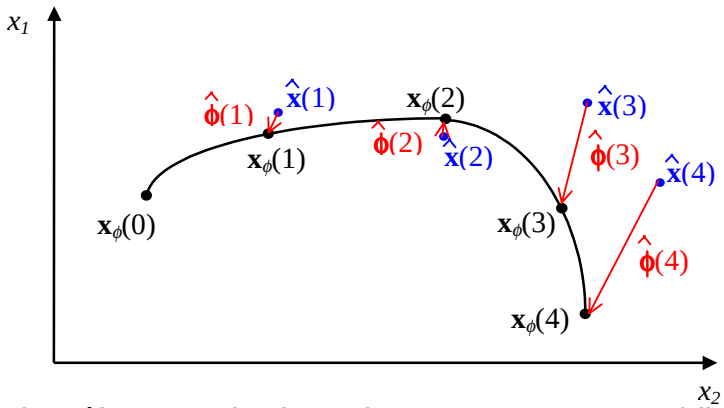

Figura 4. Vetor de resíduos e estados de um sistema em que ocorre uma falha em *t*=2.

Portanto, o procedimento para DDF pode ser descrito da seguinte forma: gera-se o vetor de resíduos comparando-se o vetor de estados medidos em (t+Δt) com a sua estimativa dada por uma ferramenta qualquer e, posteriormente, analisa-se o vetor de resíduos, indicando a falha ocorrida. Normalmente nesta etapa, uma falha é detectada quando uma banda pré-definida (que será chamada banda de detecção) sobre o vetor de resíduos é ultrapassada. Contudo, como as falhas e os erros de mapeamento geralmente são correlacionados com a dinâmica do sistema, o sistema de DDF pode não conseguir detectar uma falha ou gerar alarmes falsos.

Note que para a geração de resíduos, os estados estão sendo considerados mensuráveis. No entanto, se a saída do sistema possuir informação suficiente a respeito dos estados, uma equação de saída do tipo recursiva pode ser utilizada para a geração de resíduos. Assim, a equação do sistema que deverá ser reproduzida é dada por

$$\mathbf{y}(t+\Delta t) = \mathbf{l}\left(\mathbf{u}(t), \mathbf{d}(t), \mathbf{y}(t), \mathbf{y}(t-\Delta t), \cdots\right) \tag{10}$$

na qual $\mathbf{l}(.)$ representa a função não-linear do sistema livre de falhas.

1.2. DETECÇÃO E DIAGNÓSTICO DE FALHAS VIA REDUNDÂNCIA ANALÍTICA

Em contraste com a redundância física na qual medidas de diferentes sensores são confrontadas, na redundância analítica as medidas dos sensores são comparadas com os valores correspondentes analiticamente obtidos [GERTLER, 1988]. Usando-se o conceito de geração de resíduos, o modelo matemático é utilizado para a reprodução do comportamento dinâmico do sistema livre de falhas.

O preço a ser pago pelos benefícios do modelo matemático do sistema, além é claro do considerável esforço computacional, é a sensibilidade do sistema de DDF com respeito aos erros de modelagem que são inevitáveis na prática. Os erros de modelagem podem obscurecer os efeitos das falhas e tornarem-se uma fonte de alarmes falsos. Portanto, a sensibilidade aos erros de modelagem é o principal problema nas aplicações dos métodos de DDF que utilizam redundância analítica [FRANK, 1990].

Nos métodos que utilizam redundância analítica, para a detectabilidade e diferenciação da falha, deve-se ter: conhecimento do modelo nominal do sistema, conclusividade do comportamento faltoso, existência de relações redundantes, observação

do comportamento do sistema na presença da falha e invariância ou um mínimo de robustez com respeito a entradas desconhecidas.

A literatura traz uma grande variedade de métodos que utilizam redundância analítica. No entanto, eles basicamente podem ser englobados por dois conceitos básicos: os métodos que usam estimação de estado, nos quais se destacam o enfoque por paridade de estados e o enfoque por observadores dedicados, e os métodos que utilizam estimação paramétrica.

No enfoque por paridade de estados, a principal idéia é checar a paridade (consistência) do modelo matemático do sistema através das variáveis medidas. Quando uma banda de erros pré-definida (banda de detecção) é ultrapassada, indica-se que uma falha ocorreu. O problema de DDF empregando paridade de estados pode ser formulado como se segue: dadas q medidas redundantes das variáveis do sistema e as bandas de detecção que caracterizam um comportamento faltoso, ache uma estimativa $\hat{\mathbf{x}}$ da variável de processo a partir do mais consistente subconjunto de medidas e, identifique a medida faltosa pela verificação da paridade.

Basicamente, no enfoque por observadores dedicados, os resíduos são gerados reconstruindo-se os estados do sistema a partir das variáveis medidas (ou um subconjunto destas). Os estados são reconstruídos utilizando-se observadores via estimação de erros ou filtros de Kalman. A falha poderá ser detectada checando-se o incremento do resíduo causado pela falha. No caso simples, uma banda de detecção é usada para evitar alarmes falsos. Um caminho similar pode ser traçado para gerar resíduos usando observadores de ordem reduzida ou estimadores não-lineares. O esquema de geração de resíduos para a DDF utilizando-se um estimador de estados de ordem plena pode ser visto na Figura 5. Note que o resíduo é definido como a diferença entre as variáveis medidas e a saída estimada pelo modelo.

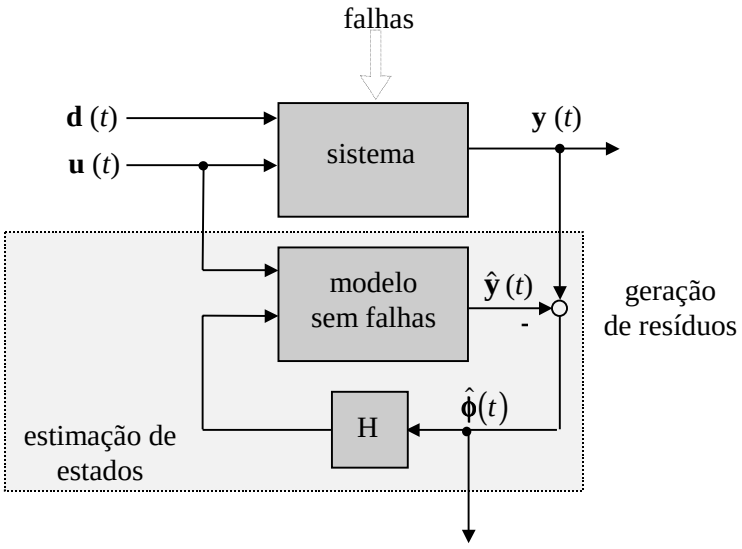

Figura 5. Geração de resíduos via observadores dedicados.

O enfoque por identificação paramétrica faz uso do fato de que as falhas nos sistemas dinâmicos são refletidas nos parâmetros físicos, como por exemplo fricção, massa, viscosidade, capacitância, etc. A idéia básica é detectar as falhas fazendo a estimativa paramétrica do modelo matemático atual e calculando os desvios em relação ao modelo do sistema sem falhas [ISERMANN, 1984]. O procedimento utilizado é ilustrado na Figura 6.

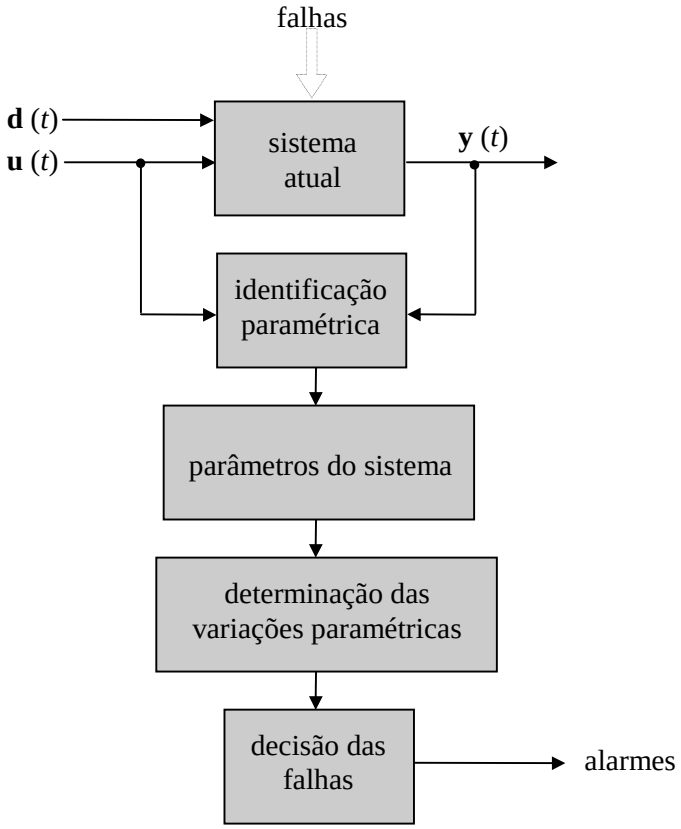

Figura 6. Detecção e diagnóstico de falhas via identificação paramétrica.

1.2.1. Robustez em sistemas de detecção e diagnóstico de falhas via redundância analítica

Em sistemas reais, os métodos de DDF baseados no modelo matemático operam sobre condições não-ideais, na presença de ruídos, distúrbios e erros de modelagem [GERTLER, 1997]. A consequência desse fato é que a diferença (resíduo) entre a variável medida e a saída do modelo matemático é diferente de zero na ausência de falhas. O que se deseja de um sistema de DDF via redundância analítica é que ele seja sensível às falhas e insensível às incertezas de modelagem e ruídos no sistema. Um método que atende a estes requisitos é dito robusto.

Para esconder os efeitos indesejáveis dos erros de modelagem e ruídos no sistema utiliza-se bandas de detecção no sinal de resíduo. O uso de bandas de detecção apresenta dois problemas principais: a sensibilidade do sistema de DDF fica reduzida e a determinação da largura das bandas é difícil de ser estabelecida, já que o resíduo varia com o sinal de entrada, com a magnitude e a natureza dos distúrbios no sistema. Escolhendo-se bandas de detecção fixas muito pequenas, aumenta-se o número de alarmes falsos. Por outro lado,

escolhendo bandas de detecção fixas muito grandes, reduz-se consideravelmente a sensibilidade do sistema de DDF.

Muitos trabalhos têm sido desenvolvidos em sistemas de DDF robustos [WILLSKY, 1976], [PATTON *et al*., 1989], [PATTON, 1994]. Várias técnicas robustas de projeto de estimadores de estado para geração de resíduos foram desenvolvidas [FRANK, 1987], [WATANABE & HIMMELBLAU, 1982], [CHOW & WILLSKY, 1984]. Mais recentemente, nesta mesma linha, pode-se citar os métodos que utilizam equações de paridade robustas [GERTLER, 1991], o enfoque por estimação robusta H_∞ [MANGOUBI *et al*., 1992], [SADRNIA *et al*., 1997 a,b], o uso de observadores de estados robustos baseados na teoria de modos deslizantes [CAMINHAS, 1997], entre outros.

Por outro lado, foram desenvolvidos vários enfoques que aumentam a robustez da DDF pela escolha apropriada de bandas de detecção ou fazendo-as adaptativas ao sinal de entrada do sistema, como foi proposto por CLARK no livro de PATTON *et al.* (1989). A análise de alguns dos mais recentes métodos de DDF podem ser encontrados no trabalho de GERTLER (1997).

1.3. TÉCNICAS DE INTELIGÊNCIA ARTIFICIAL APLICADAS EM SISTEMAS DE DETECÇÃO E DIAGNÓSTICO DE FALHAS

Basicamente, técnicas de Inteligência Artificial (IA) podem ser utilizadas em sistemas de DDF para três funções distintas. Na primeira, as técnicas de IA são usadas para a produção de classificadores baseados nas variáveis medidas do processo. Na segunda, o conceito de geração de resíduos é utilizado, sendo que as técnicas de IA são empregadas para reproduzir o comportamento dinâmico do sistema e/ou para a classificação do vetor de resíduos (neste caso, indiretamente, a técnica empregada produz bandas de detecção variáveis baseadas nas medidas do sistema). As técnicas de IA, ainda, podem ser usadas para produzir a estimativa paramétrica do sistema e, em um procedimento semelhante ao descrito anteriormente, detectar as falhas.

Dentre as ferramentas de IA utilizadas em sistemas de DDF, destacam-se três: os sistemas baseados em conhecimento (ou sistemas especialistas), a lógica nebulosa e as redes neurais artificiais.

1.3.1. Sistemas baseados em conhecimento

Também conhecidos como sistemas especialistas, os sistemas baseados em conhecimento são programas de computador que fazem uso de relações heurísticas e fatos (experiência) tal qual os especialistas humanos fazem. Sistemas especialistas oferecem um formalismo proveitoso para DDF porque eles podem considerar diversas fontes de dados e subproblemas de abstrações [FRANK, 1990]. Entre as funções de um sistema especialista, destacam-se para os sistemas de DDF:

a) Interpretação, ou seja, uma correta, consistente e completa análise de dados;
b) Diagnóstico e,
c) Monitoração, ou seja reconhecimento das condições de alarme.

O sistema especialista pode ser implementado através de um sistema baseado em regras, consistindo de um banco de dados, um banco de regras e um interpretador de regras [CHARNIAK & MCDERMOTT, 1985].

Atualmente, tais sistemas têm sido muito usados em conjunto com lógica nebulosa em sistemas de DDF [EVSUKOFF *et al.*, 1997], [CALADO & ROBERTS, 1997]. A lógica nebulosa é aplicada em sistemas onde as informações são vagas e imprecisas, como grande parte dos problemas de DDF.

1.3.2. Lógica nebulosa

Lógica nebulosa foi desenvolvida para prover algoritmos de processamento de informações que podem “raciocinar” sobre ou utilizar dados imprecisos [BROWN & HARRIS, 1994]. Para isso, utiliza informações linguísticas tal qual os seres humanos o fazem.

Pode ser utilizada para a construção de bandas de detecção baseadas nas variáveis do processo [SCHNEIDER & FRANK, 1996] ou para a classificação destas. Algumas vezes é utilizada em conjunto com as redes neurais artificiais [FÜSSEL *et al*., 1997], [CAMINHAS *et al*., 1996], [CAMINHAS, 1997] para as tarefas citadas anteriormente.

1.3.3. Redes neurais artificiais

São sistemas computacionais inspirados em certas características dos sistemas biológicos. Existe um grande número de tipos de redes neurais artificiais (RNA) que

basicamente diferem em suas arquiteturas e suas formas de aprendizado. Entre as diferentes arquiteturas, as mais utilizadas são as recorrentes e as com propagação para a frente (diretas). Entre as formas de aprendizado, sobressaem a supervisionada e a não-supervisionada. As RNA são aplicadas para a realização de inúmeras tarefas, como por exemplo, aproximação de funções (especialmente não-lineares), associação de padrões, reconhecimento de padrões, classificação e predição.

Em sistemas de DDF, as RNA têm sido empregadas nos últimos anos especialmente em sistemas estáticos e menos intensivamente em sistemas dinâmicos [KORBICZ, 1997]. Na maioria das aplicações, as RNA têm sido utilizadas como classificadores baseados nas variáveis medidas do processo (salienta-se que as entradas do sistema geralmente não são utilizadas). Em tais soluções, as RNA podem ser implementadas com aprendizado supervisionado ou não-supervisionado. No aprendizado supervisionado, um *perceptron* multicamadas (do inglês, *multilayer perceptron* - MLP) é geralmente empregado tendo como padrões de entrada, vetores *p*-dimensionais contendo as variáveis medidas do processo e, na saída, o estado das diferentes classes (diferentes falhas e sistema em operação normal). As diferentes classes devem ter regiões separáveis no espaço de entradas *p*-dimensional. Para cada padrão de entrada, um vetor *q*-dimensional descrevendo os estados das diferentes classes deverá ser utilizado para o aprendizado da RNA. Entretanto, em algumas aplicações não se conhece profundamente o comportamento das variáveis medidas quando submetidas às falhas. Neste caso, devem ser empregadas RNA com aprendizado não-supervisionado. Uma modelo típico de tais redes é a dos mapas auto-organizáveis de Kohonen [KOHONEN, 1995]. Em tais redes, os diferentes padrões são separados em aglomerados (*clusters*) que posteriormente devem ser associados às diferentes falhas e à operação normal.

No entanto, em sistemas dinâmicos tais procedimentos geralmente não são válidos pois as variáveis de saída medidas comumente sofrem os efeitos das entradas do sistema. Isto ocorre especialmente em sistemas dinâmicos não-lineares [KORBICZ, 1997]. Um enfoque que surgiu para superar tais dificuldades é o baseado no conceito de geração de resíduos. Neste enfoque, o modelo matemático ou uma RNA é utilizada para a reprodução do comportamento dinâmico do sistema livre de falhas. As saídas do modelo matemático ou da RNA são comparadas com os valores das variáveis medidas do sistema, gerando assim o vetor de resíduos, que é aplicado em uma RNA que tem por objetivo a classificação das falhas. Esta é a arquitetura utilizada para a DDF neste trabalho e será discutida com maiores detalhes no Capítulo 4. No Capítulo seguinte, as RNA utilizadas neste trabalho serão apresentadas.

Capítulo 2

As redes neurais artificiais

As redes neurais artificiais (RNA) têm sido aplicadas na solução de uma infinidade de problemas. O adjetivo “neural” é usado porque muito da inspiração de tais redes vem da neurociência [HERTZ *et al*., 1991]. As RNA empregam como unidade de processamento fundamental o neurônio artificial, inspirado no funcionamento básico dos neurônios biológicos.

Neste trabalho, as RNA são utilizadas para duas tarefas distintas: aproximação de funções não-lineares e classificação de padrões. O *perceptron* multicamadas (*multilayer perceptron* - MLP) com aprendizado por retropropagação do erro (*backpropagation*) é, atualmente, a RNA mais utilizada para a aproximação de funções não-lineares contínuas. Para a classificação de padrões em DDF, no entanto, esta RNA apresenta alguns problemas. Tais problemas levam à utilização de outra RNA conhecida como rede de função de base radial (*radial basis function network* - rede RBF). Neste trabalho, com o objetivo de comparação, as redes RBF empregadas utilizam quatro diferentes métodos de treinamento. Os dois primeiros utilizam regularização para penalizar os pesos grandes e, assim, gerar um efeito de suavidade na função de saída da rede. O terceiro método utiliza seleção de subconjuntos para escolher os centros das unidades radiais a partir dos padrões do conjunto de treinamento. O quarto algoritmo emprega o mapa auto-organizável de Kohonen para selecionar os centros das unidades radiais em posições próximas aos centros dos aglomerados de padrões (*clusters*) pertencentes a uma mesma classe.

2.1. *PERCEPTRON* MULTICAMADAS COM TREINAMENTO POR RETROPROPAGAÇÃO DO ERRO

O *perceptron* multicamadas pode ser visto como um veículo prático para realizar mapeamentos de funções não-lineares de maneira geral [HAYKIN, 1994]. A relação entrada/saída do MLP define um mapeamento de um espaço de entrada Euclidiano *p*-dimensional para um espaço de saída Euclidiano *q*-dimensional continuamente diferenciável. Um MLP com apenas uma camada escondida pode ser visto na Figura 7.

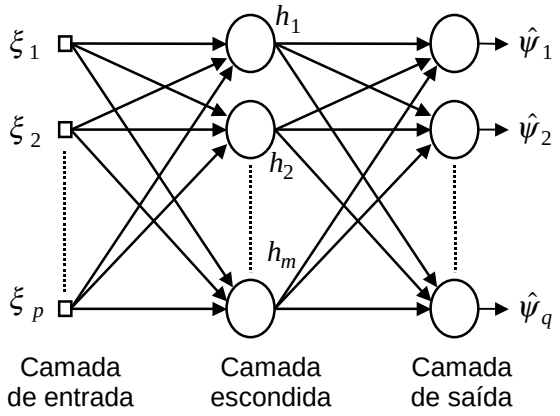

Figura 7. *Perceptron* multicamadas com uma única camada escondida.

Para um MLP com uma única camada escondida, apresentando-se o *n*-ésimo padrão de entrada $\boldsymbol{\xi}(n) = [\xi_1(n)\ \xi_2(n)\ \ ...\ \xi_p(n)]^T$, a ativação do neurônio de saída *k* e a ativação dos neurônios *j* da camada escondida, respectivamente, são dadas por

$$\hat{\psi}_k(n) = \varphi_k\left(\sum_{j=0}^{m} \omega_{kj}(n)\, h_j(n)\right), \tag{11}$$

$$h_j(n) = \varphi_j\left(\sum_{i=0}^{p} \omega_{ji}(n)\xi_i(n)\right) \tag{12}$$

nas quais φ_a é a função de ativação não-linear do neurônio *a*, ω_{cb} é o peso entre a saída do neurônio *b* (camada anterior) e a entrada do neurônio *c* (camada posterior), $i = 1, ..., p$ é o índice dos neurônios da camada de entrada, $j = 1, ..., m$ é o índice dos neurônios da camada escondida e $k = 1, ..., q$ é o índice dos neurônios da camada de saída.

Uma função de ativação não-linear comumente empregada no MLP é a sigmoidal, que é dada por

$$\varphi_a(v_a) = \frac{1}{1+\exp(-v_a)} \tag{13}$$

na qual v_a é o nível de ativação interna no neurônio *a*.

A seguir, mostra-se o algoritmo para treinamento do MLP por retropropagação do erro. Apresentando-se o *n*-ésimo padrão de treinamento (n = 1,...,n_p) na entrada do MLP, o erro instantâneo na saída *k* é dado por

$$e_k(n) = \psi_k(n) - \hat{\psi}_k(n) \tag{14}$$

na qual $\psi_k(n)$ é a variável que deve ser estimada pela saída da RNA $\hat{\psi}_k(n)$. A soma dos erros quadráticos instantâneos nas saídas do MLP para o padrão *n* é dada por

$$E(n) = \frac{1}{2}\sum_{k=1}^{q} e_k^2(n) \tag{15}$$

e o erro médio quadrático sobre o conjunto de treinamento é dado por

$$E_{medio} = \frac{1}{n_p}\sum_{n=1}^{n_p} E(n). \tag{16}$$

Utilizando-se a lei *delta* [HERTZ *et al*., 1991], as conexões entre a camada escondida e a camada de saída são ajustadas por

$$\Delta\omega_{kj}(n) = -\eta(n)\frac{\partial E(n)}{\partial \omega_{kj}(n)} \tag{17}$$

na qual η é a taxa de aprendizagem e $\Delta\omega_{kj}$ é a correção aplicada ao peso ω_{kj}. Da Equação anterior e da Eq. (15), chega-se a lei de ajuste dos pesos

$$\Delta\omega_{kj}(n) = \eta(n)\delta_k(n)h_j(n) \tag{18}$$

na qual

$$\delta_k(n) = e_k(n)\varphi_k'\left(v_k(n)\right). \tag{19}$$

Do mesmo modo, para as conexões entre a camada de entrada e a camada escondida, tem-se que o ajuste de pesos é dado por

$$\Delta\omega_{ji}(n) = \eta(n)\delta_j(n)\xi_i(n) \tag{20}$$

na qual

$$\delta_j(n) = \varphi_j'\left(v_j(n)\right)\sum_{k=1}^{q}\delta_k(n)\omega_{kj}(n). \tag{21}$$

A definição da taxa de aprendizagem tem um papel importante no treinamento por retropropagação do erro. Se a taxa é muito baixa, o algoritmo demorará para convergir. Se, por outro lado, a taxa é muito alta, o algoritmo pode se tornar instável. Um método simples de aumentar a velocidade de convergência e evitar a instabilidade é modificar a lei de ajuste adicionando um termo de *momentum* que é proporcional ao ajuste de pesos anterior [HAYKIN, 1994].

Neste trabalho, o MLP tem apenas uma camada escondida. O seguinte teorema estabelece que uma RNA direta (*feedforward*) com uma única camada escondida e com um número suficiente de unidades escondidas é capaz de aproximar qualquer função contínua *f*: $\Re^p \rightarrow \Re^q$ com qualquer acuracidade desejada.

Teorema A1: [CYBENKO, 1989] Seja φ qualquer função de ativação contínua. Então, dada qualquer função contínua com valores reais $f(.)$, em um subespaço compacto $s \subset \Re^{n_p}$ e $\varepsilon > 0$, existem vetores $\boldsymbol{\omega}_1$, $\boldsymbol{\omega}_2$,..., $\boldsymbol{\omega}_m$, $\boldsymbol{\alpha}$, $\boldsymbol{\theta}$ e uma função parametrizada $G(\,.\,, \boldsymbol{\omega}, \boldsymbol{\alpha}, \boldsymbol{\theta}):\rightarrow\Re$ tal que

$$\left|G(\boldsymbol{\xi}, \boldsymbol{\omega}, \boldsymbol{\alpha}, \boldsymbol{\theta}) - f(\boldsymbol{\xi})\right| < \varepsilon \tag{22}$$

$$G(\boldsymbol{\xi},\boldsymbol{\omega},\boldsymbol{\alpha},\boldsymbol{\theta}) = \sum_{j=1}^{m} \alpha_j \varphi\left(\boldsymbol{\omega}_j^{\mathrm{T}}\boldsymbol{\xi} + \theta_j\right) \tag{23}$$

na qual $\boldsymbol{\omega}_j \in \Re^p$, $\boldsymbol{\xi} \in \Re^p$, $\alpha_j \in \Re$, $\theta_j \in \Re$, $\boldsymbol{\omega}_j = [\omega_{j1} \ldots \omega_{jp}]^{\mathrm{T}}$, $\boldsymbol{\alpha} = [\alpha_1 \ldots \alpha_m]^{\mathrm{T}}$ e $\boldsymbol{\theta} = [\theta_1 \ldots \theta_m]^{\mathrm{T}}$.

Este teorema pode ser interpretado da seguinte maneira: uma falha na função de mapeamento do MLP é resultado de uma escolha de parâmetros inadequada ou de um insuficiente número de unidades escondidas [EFRATI, 1997].

Neste trabalho, o MLP com treinamento por retropropagação do erro será utilizado para aproximar a função dinâmica do sistema. Inicialmente, esta RNA também deveria ser utilizada para o problema de classificação do vetor de resíduos.

Para o problema de classificação, o MLP produz bordas de decisão que separam os padrões das diferentes classes. Bordas de decisão são superfícies (ou linhas para o caso de dois neurônios na entrada) no espaço de entradas onde a saída dos dois (ou mais) neurônios com maior ativação na última camada são iguais para um mesmo padrão. As bordas de decisão produzidas pelo MLP com treinamento por retropopagação são posicionadas muito perto da superfície que separa os padrões de treinamento pertencentes às diferentes classes. Esta característica do MLP pode gerar má-classificação dos padrões não-treinados em problemas (como por exemplo em DDF) em que as superfícies que separam as diferentes classes são difíceis de serem estipuladas através de um conjunto limitado de padrões. Este problema poderia ser evitado se as bordas de decisão estivessem em posições mais conservadoras. Além disso, nas áreas do espaço de entradas não ocupadas pelos padrões do conjunto de treinamento a classificação é arbitrária já que os pontos de saída desejados somente refletem as ativações da rede para os padrões empregados no treinamento [LEONARD & KRAMER, 1991]. Portanto, sob certas condições, o MLP com treinamento por retropropagação pode produzir bordas de decisão que são não-intuitivas e não-robustas.

Para ilustrar tais características será apresentado um exemplo adaptado de LEONARD & KRAMER (1991). O seguinte problema de DDF é uma versão simplificada de muitos dos problemas encontrados em processos reais, em que a RNA é utilizada para a classificação das falhas (observe que o conceito de geração de resíduos não é empregado).

Exemplo 1: Considere que o vetor de estados de um processo em condição de operação normal seja $\mathbf{x}_0$ e as falhas sejam representadas por mudanças em um conjunto de parâmetros **b**. Quando não existem falhas, o conjunto de parâmetros **b** deve ter seus valores numéricos

próximos de zero. O efeito da variação destes parâmetros sobre o vetor de estados **x** é assumido como linear

$$x_i = x_{0i} + \sum_{j=1}^{n_b} a_{ij} b_j + c_i \,, \qquad i = 1,\ldots,n_x \tag{24}$$

na qual **a** é o vetor responsável pela direção do efeito de **b** em **x** e, **c** é um vetor de ruídos de medida Gaussiano.

Assumindo n_x=2 e n_b=2, as classes são definidas como segue

- Classe 1 (operação normal): $| b_1 | < 0{,}1$ e $| b_2 | < 0{,}1$;
- Classe 2 (falha 1): $| b_1 | > 0{,}1$ e,
- Classe 3 (falha 2): $| b_2 | > 0{,}1$.

Para os padrões com falha 1, considera-se $\mathbf{a}_{1j} = [1\ -1]^T$ e, para os padrões com falha 2, $\mathbf{a}_{2j} = [1\ 1]^T$. A variância do ruído de medida é definida como 0,15. Utilizando a Eq. (24) são gerados 202 padrões que podem ser vistos na Figura 8.

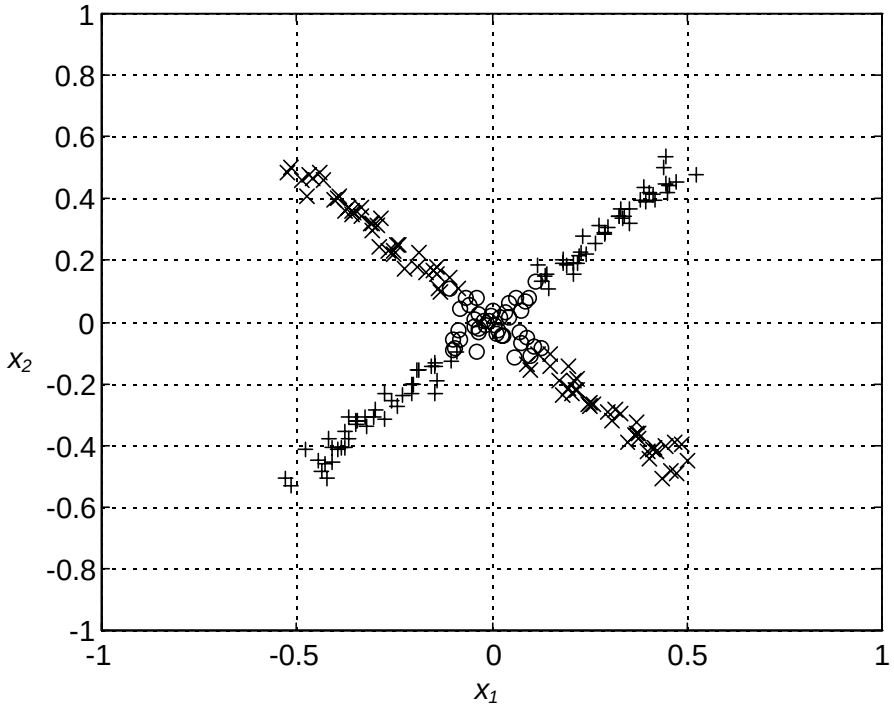

Figura 8. Padrões utilizados no exemplo 1: "o" operação normal, "x" falha 1, "+" falha 2.

Os padrões vistos na Figura 8 foram empregados para o treinamento de um MLP com aprendizado por retropropagação do erro (veja parâmetros no Apêndice A1). O MLP tem

dois neurônios na camada de entrada (estados x_1 e x_2), 6 neurônios na camada escondida e 3 neurônios na camada de saída (3 classes). A separação do espaço de entradas de acordo com a classificação feita pelo MLP após o treinamento (400 épocas) pode ser vista na Figura 9. Salienta-se que esta separação é dependente da arquitetura e do treinamento do MLP. No entanto, de modo geral, esta caso particular exemplifica bem o problema de classificação do MLP.

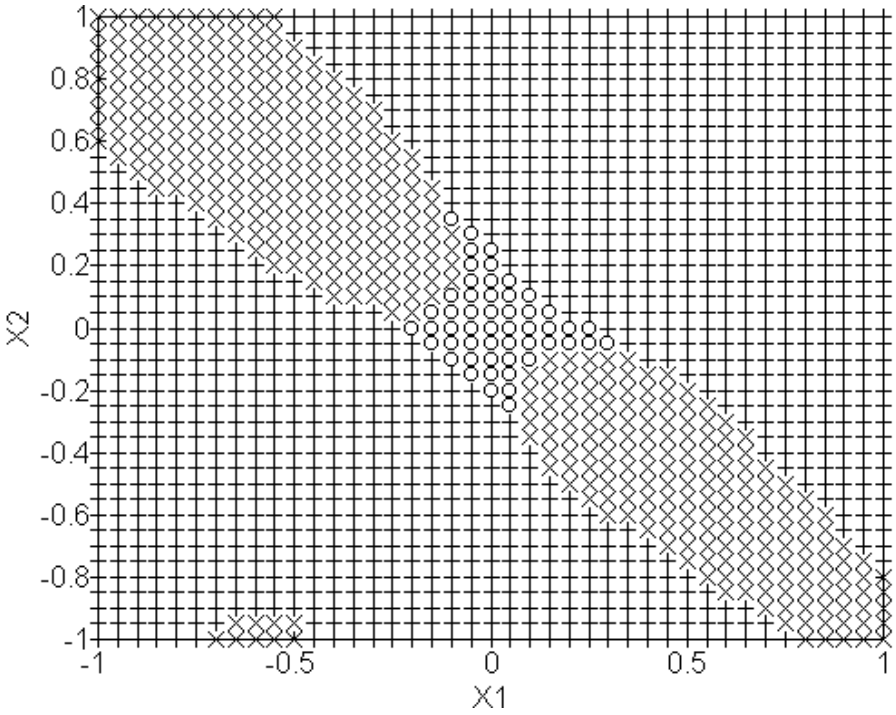

Figura 9. Separação do espaço de entradas do exemplo 1 de acordo com a classificação feita pelo MLP treinado por retropropagação do erro: "o" operação normal, "x" falha 1, "+" falha 2.

Observe que a região classificada como falha 1 é menor que a região classificada como falha 2 sem que os padrões do conjunto de treinamento indicassem tal comportamento. Isto ocorre porque a classificação das regiões reflete as ativações do MLP para os padrões utilizados no treinamento. Veja que para o conjunto de treinamento a classificação é satisfatória. No entanto, tal característica pode gerar má-classificação para problemas de DDF em que o tamanho do conjunto de treinamento geralmente é limitado. Note também que existe uma região no quadrante inferior esquerdo que foi definida como falha 1, apesar de não haver padrões de treinamento localizados nesta área que indiquem tal comportamento.

Estes problemas podem ser superados quando se utilizam redes RBF, nas quais a classificação é feita de acordo com a distância entre o padrão a ser classificado e os centros das unidades radiais que representam as diferentes classes.

2.2. REDE COM FUNÇÃO DE BASE RADIAL (REDE RBF)

Funções de base radial são, simplesmente, uma classe de funções. MOODY & DARKEN (1989) propuseram seu uso em um novo tipo de RNA chamada rede com função de base radial (RBF). Esta RNA é inspirada em neurônios que têm ativações localmente sintonizadas ou neurônios seletivos, que respondem para determinadas faixas de sinais de entrada e são encontrados em diversas partes do corpo humano e de outros animais. Em princípio, as redes RBF podem ser multicamadas e terem funções de ativação na saída não-lineares. Contudo, redes RBF têm tradicionalmente sido associadas com funções radiais em uma única camada escondida e funções de saída lineares [ORR, 1996 a].

A rede RBF utilizada neste trabalho tem três camadas distintas. Na primeira, os padrões de entrada são apresentados. Não existem pesos entre a primeira e a segunda camadas, sendo os padrões de entrada repassados para os neurônios da segunda camada (camada escondida). As unidades desta camada têm função de ativação radial. Cada neurônio j da camada escondida (chamado unidade radial j) é responsável pela criação de um campo receptivo no espaço de entradas centrado em um vetor $\boldsymbol{\mu}_j$, chamado de centro da unidade radial. A unidade radial j tem ativação de acordo com a distância entre o vetor de entrada e o centro da unidade radial. Quanto mais próximos forem os dois vetores, maior será a ativação do neurônio (Figura 10). Entre a segunda e a terceira camadas existem pesos e a terceira camada apresenta ativação linear. A k-ésima (k = 1,...,q) saída da rede RBF para o n-ésimo (n = 1,...,n_p) vetor de entrada $\boldsymbol{\xi}_n$, é dada por

$$\hat{\psi}_k(n) = \sum_{j=0}^{m} \omega_{kj} h_j(n) \tag{25}$$

na qual h_j é a ativação da unidade radial j da camada escondida e ω_{kj} é o peso entre a unidade radial j e o neurônio de saída k. A ativação das unidades da camada escondida é definida por uma função radial. Uma função radial de ativação comum é a Gaussiana. Aplicando tal função na unidade radial j, a ativação desta unidade para o n-ésimo vetor de entrada é dada por

$$h_j(n) = \exp\left(-\frac{\left\| \boldsymbol{\xi}(n) - \boldsymbol{\mu}_j \right\|^2}{2\rho^2} \right) \tag{26}$$

na qual ρ determina o tamanho do campo receptivo da unidade radial j e || . || define a norma do vetor (geralmente é a Euclidiana).

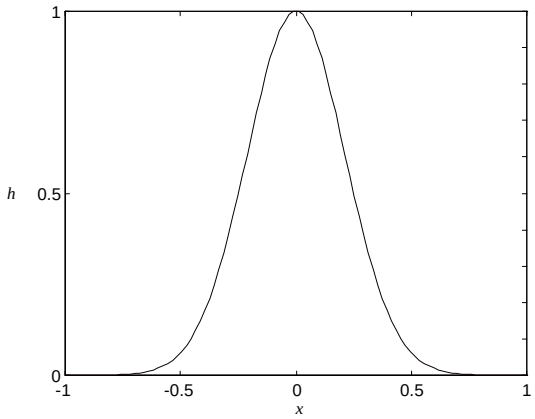

Figura 10. Resposta da função Gaussiana com o centro em 0 (μ = 0) e ρ = 0,3. Note que a ativação máxima ocorre em $x = \mu$.

Neste trabalho, a função de ativação utilizada nas unidades radiais é a função de Cauchy. Esta função foi escolhida por apresentar um decaimento mais suave conforme os padrões se distanciam do centro da unidade radial. Assim, mesmo os padrões nas regiões do espaço de entradas que não são ocupadas pelos padrões de treinamento serão classificados de acordo com a distância até o centro mais próximo, sem a necessidade de se aumentar o tamanho do campo receptivo. Aqui, a função de Cauchy é dada por

$$h_j(n) = \frac{1}{1 + \left\| \mathbf{R}^{-1}\left(\boldsymbol{\xi}(n) - \boldsymbol{\mu}_j\right) \right\|^2} \tag{27}$$

na qual **R** é uma matriz diagonal formada pelos parâmetros individuais que definem o tamanho do campo receptivo em cada dimensão do espaço de entradas, ou seja

$$\mathbf{R} = \begin{bmatrix} \rho_1 & 0 & 0 & 0 \\ 0 & \rho_2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \rho_p \end{bmatrix}. \tag{28}$$

Desta forma, o campo receptivo não precisa necessariamente ter o mesmo tamanho em todas as dimensões do espaço de entradas. Portanto, se uma determinada variável do

vetor de entrada tem um poder discriminatório menor, o tamanho do campo receptivo em sua dimensão pode ser maior, priorizando-se, assim, as outras variáveis.

Para os problemas de DDF, a rede RBF pode produzir bordas de decisão que são mais robustas e intuitivas que as do MLP, já que a classificação é feita considerando-se a proximidade entre o padrão a ser classificado e os centros das unidades radiais. É claro que este resultado é dependente da escolha adequada das unidades radiais e de seus parâmetros Por exemplo, treinando uma rede RBF (veja parâmetros no Apêndice A1) com os padrões apresentados no exemplo 1, a classificação do espaço de entradas é muito mais interessante do que a feita pelo MLP com treinamento por retropropagação do erro (Figura 11).

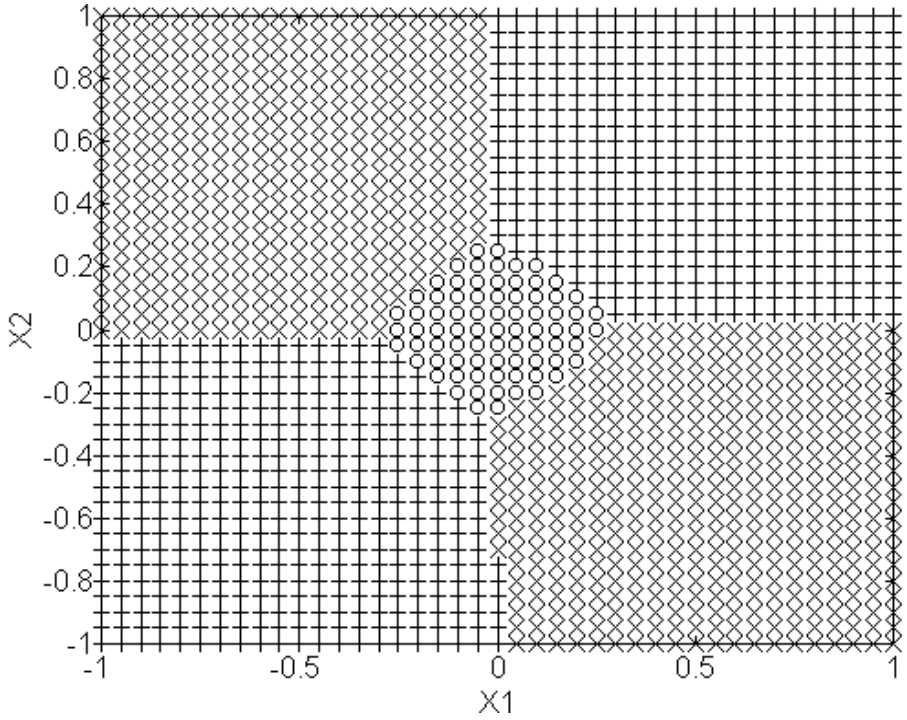

Figura 11. Separação do espaço de entradas do exemplo 1 de acordo com a classificação feita pela rede RBF: "$_o$" operação normal, "$_x$" falha 1, "$_+$" falha 2.

As redes RBF ainda apresentam outras vantagens sobre o MLP, tais como inexistência de mínimos locais no cálculo dos pesos e um menor tempo de treinamento [LOONEY, 1997]. Entre as desvantagens, pode-se citar: as redes RBF ocupam mais memória devido ao número alto de unidades escondidas (unidades radiais), a velocidade de operação pode ser menor devido ao maior número de unidades escondidas e a escolha das unidades radiais e de seus parâmetros é subótima. Neste ponto, uma questão que pode surgir é: por que não utilizar também a rede RBF para a aproximação da função dinâmica do sistema no problema de DDF? Realmente, a rede RBF apresenta bons resultados no

mapeamento de sistemas não-lineares com poucas variáveis medidas [NELLES & ISERMANN, 1995]. Contudo, se a dimensão do espaço de entradas é grande, um número de unidades radiais bastante alto é requerido para tratar a complexidade do problema [WARWICK & CRADDOCK, 1996]. Isto pode resultar em uma baixa generalização e em um esforço computacional extremamente alto durante a fase de treinamento da rede RBF.

Agora, será considerado o problema do treinamento da rede RBF. Se a rede RBF tem apenas uma camada escondida e as unidades radiais são fixadas (isto é, o número de unidades radiais e seus parâmetros são constantes), então esta RNA pode ser vista como um modelo linear. Neste caso, o treinamento pode ser feito de maneira desacoplada: primeiro determina-se as unidades radiais e seus parâmetros e depois calcula-se os pesos da segunda camada de acordo com as ativações das unidades radiais e com as saídas desejadas. Assim, evita-se o uso de algoritmos de otimização não-lineares, como os do tipo gradiente descendente, que apresentam um esforço computacional relativamente alto para cálculo dos pesos. Outra vantagem é que evita-se os mínimos locais no cálculo dos pesos.

A determinação do número de unidades radiais e seus parâmetros é a primeira etapa do treinamento. O método RBF original utiliza como centros todos os padrões de treinamento. No entanto, como o número de padrões é muito grande em DDF, este método é raramente utilizado. Além disso, utilizando-se um número grande de unidades radiais pode haver problemas de sobreajuste (*overfitting*), ou seja, a classificação será sensível à escolha do conjunto de treinamento, apresentando uma baixa generalização (maus resultados para padrões não-apresentados).

Neste trabalho, utilizou-se quatro diferentes métodos para a escolha das unidades radiais. Os três primeiros escolhem as unidades radiais a partir dos padrões de treinamento, sendo que os dois primeiros utilizam regularização e o terceiro utiliza seleção de subconjuntos. O quarto emprega o mapa auto-organizável de Kohonen para posicionar os centros das unidades radiais perto dos centros dos aglomerados de padrões (*clusters*) das diferentes classes. Antes da descrição destes métodos, o problema da generalização para RNA será descrito.

Para atingir a melhor generalização, a complexidade do modelo (RNA) precisa ser otimizada [BISHOP, 1995]. Para uma melhor compreensão, o erro de generalização será decomposto na soma da variância com o *bias* ao quadrado. Um modelo que é muito simples, ou muito inflexível, terá um *bias* grande, enquanto que um modelo muito flexível em relação a um particular conjunto de padrões terá uma grande variância. Considerando que $\hat{\psi}(\xi)$ é a saída do modelo de predição (neste caso é a saída da rede RBF) da variável medida

$\psi(\boldsymbol{\xi})$ para um vetor de entrada $\boldsymbol{\xi}$, o erro médio quadrático sobre os padrões do espaço de entradas é dado por

$$E_{medio} = \left\langle \left(\psi(\boldsymbol{\xi}) - \hat{\psi}(\boldsymbol{\xi})\right)^2 \right\rangle \qquad (29)$$

na qual a expectância, que é indicada por < . >, é considerada sobre os padrões do espaço de entradas. Separando a equação acima em duas componentes [GERMAN *et al*., 1992]

$$E_{medio} = \left(\psi\ (\boldsymbol{\xi}) - \left\langle \hat{\psi}\ (\boldsymbol{\xi})\right\rangle\right)^2 + \left\langle \left(\hat{\psi}\ (\boldsymbol{\xi}) - \left\langle \hat{\psi}\ (\boldsymbol{\xi})\right\rangle\right)^2 \right\rangle \qquad (30)$$

na qual a primeira parte desta equação é o *bias* ao quadrado e a segunda parte é a variância. O *bias* indica a diferença entre a saída desejada e a média das saídas do modelo. A variância indica a sensibilidade às peculiaridades (tal qual ruídos e escolha dos padrões) de cada conjunto particular de treinamento. A melhor generalização ocorre quando existe o melhor compromisso entre as conflitantes necessidades de *bias* pequeno e variância pequena. Isto pode ser obtido controlando a flexibilidade do modelo. Um modo de controlar a flexibilidade da rede RBF é utilizar a técnica da regularização (como nos dois primeiros métodos de treinamento), conhecida de Regressão Linear, para diminuir a sensibilidade do modelo em relação ao conjunto de treinamento e, assim, diminuir a variância. Em um outro enfoque, a flexibilidade pode ser controlada variando-se o número de parâmetros adaptativos da rede RBF (como em seleção de subconjuntos). Finalmente, a flexibilidade pode ser controlada escolhendo os centros das unidades radiais como uma média dos padrões pertencentes a aglomerados de uma mesma classe (como empregando o mapa auto-organizável de Kohonen). Assim, diminui-se a variância e o número de parâmetros adaptativos da rede RBF.

2.2.1. Treinamento da rede RBF via regularização

Por volta dos anos 50, o russo Andre Tikhonov desenvolveu a técnica matemática conhecida como regularização para a solução de problemas mal definidos [ORR, 1996 a]. Estes são problemas nos quais não há uma única solução porque não existem informações suficientes no problema. No entanto, o trabalho de Tikhonov só ficou amplamente

conhecido no Ocidente depois da publicação de seu livro em 1977 [TIKHONOV & ARSENIN, 1977].

Enquanto isso, dois estatísticos americanos, Arthur Hoerl e Robert Kennard, desenvolveram [HOERL & KENNARD, 1970] o conceito de *ridge regression*, um método para a solução de problemas de regressão linear mal condicionados. Mal condicionamento significa dificuldades em resolver a matriz inversa necessária para a obtenção da matriz de variância. *Ridge regression* utiliza o fato de que uma matriz quadrada singular pode se tornar não-singular adicionando uma constante na diagonal da matriz [RAWLINGS, 1988]. Se por exemplo, $\mathbf{A}^T\mathbf{A}$ é singular, então ($\mathbf{A}^T\mathbf{A} + k\,\mathbf{I}$) é não-singular, na qual k é uma constante positiva pequena. O problema de mal condicionamento é um tipo de problema de regressão mal definido no sentido de Tikhonov e o método de Hoerl e Kennard é de fato uma forma de regularização conhecida como regularização de ordem zero [PRESS *et al.*, 1992].

Se na Eq. (30), $\langle \hat{\psi}(\boldsymbol{\xi}) \rangle = \psi(\boldsymbol{\xi})$ para todo $\boldsymbol{\xi}$, o *bias* é igual a zero. No entanto, se a variância for grande, o erro médio quadrático ainda poderá ser grande. Este será o caso se a saída do modelo de predição é altamente sensível às peculiaridades (tais como ruído e escolha dos padrões) de um conjunto de treinamento particular e esta sensibilidade causa problemas de regressão mal definidos no sentido de Tikhonov. Contudo, a variância pode ser significativamente reduzida introduzindo-se uma pequena quantidade de *bias* na função de custo a ser minimizada.

Introduzir um termo de *bias* é equivalente a restringir o domínio das funções as quais o modelo pode reproduzir. Tipicamente isto é alcançado removendo graus de liberdade. Como exemplo, pode-se diminuir a ordem de um polinômio ou reduzir o número de pesos na RNA. *Ridge regression* não remove explicitamente os graus de liberdade mas, ao invés disso, diminui o número efetivo de parâmetros. Como resultado tem-se uma perda de flexibilidade que faz o modelo menos sensível.

Um método conveniente de se restringir a flexibilidade de modelos lineares é aumentar a soma dos erros quadráticos com um termo que penaliza os pesos grandes. Quando as RNA se tornaram populares na década de 1980, uma das técnicas que surgiram para “podar” conexões sem importância na rede foi a de decaimento de pesos. Logo foi reconhecido que *ridge regression* e decaimento de pesos são equivalentes pois acrescentam o mesmo termo de penalidade à soma dos erros quadráticos.

Neste trabalho, utilizou-se *ridge regression* de duas formas: a primeira, a qual será chamada *global ridge regression*, utiliza um único termo de penalidade (ou regularização)

para todos os pesos, e a segunda, *local ridge regression*, utiliza temos de regularização individuais para os vários pesos.

2.2.1.1 *Global ridge regression* (GRR):

No método *global ridge regression* (GRR), um único termo de regularização é utilizado para todos os pesos. A função de custo (para o neurônio de saída *k*) é dada pela soma dos erros quadráticos mais o termo de regularização aplicado aos pesos

$$C_k = \sum_{n=1}^{n_p} \left(\psi_k\ (n) - \hat{\psi}_k\ (n)\right)^2 + \lambda \sum_{j=1}^{m} \omega\ _{kj}^{2} \tag{31}$$

na qual λ é o parâmetro de regularização (ou penalidade) que controla o balanço entre ajustar a função e evitar a penalização, ψ é a saída desejada e *m*, neste caso, é igual a n_p pois todos os padrões do conjunto de treinamento são escolhidos como centros das unidades radiais. O *bias* introduzido favorece as soluções envolvendo pesos pequenos e o efeito de suavidade da função de saída, já que pesos grandes são usualmente requeridos para produzir funções de saída com variações rápidas (bruscas).

Como, após a determinação das ativações das unidades radiais para o conjunto de treinamento, o modelo pode ser visto como linear, pode-se calcular, minimizando a Eq. (31), a matriz de pesos ótima (veja Apêndice A2)

$$\hat{\mathbf{\Omega}} = \left(\mathbf{H}^{\mathrm{T}}\mathbf{H} + \lambda\ \mathbf{I}_m\right)^{-1} \mathbf{H}^{\mathrm{T}}\mathbf{\Psi} \tag{32}$$

na qual a matriz identidade **I** tem dimensão *m* (número de unidades radiais que, aqui, é igual ao número de padrões), a matriz de pesos ótima é formada por $\hat{\mathbf{\Omega}} = \left[\hat{\boldsymbol{\omega}}_1 \quad \hat{\boldsymbol{\omega}}_2 \quad \ldots \quad \hat{\boldsymbol{\omega}}_q\right]$ na qual $\hat{\boldsymbol{\omega}}_k = \left[\hat{\omega}_{k1} \quad \hat{\omega}_{k2} \quad \ldots \quad \hat{\omega}_{km}\right]^{\mathrm{T}}$, a matriz de saída desejada é formada por $\mathbf{\Psi} = \left[\boldsymbol{\psi}_1 \quad \boldsymbol{\psi}_2 \quad \ldots \quad \boldsymbol{\psi}_q\right]$ na qual $\boldsymbol{\psi}_k = \left[\psi_k(1) \quad \psi_k(2) \quad \ldots \quad \psi_k\left(n_p\right)\right]^{\mathrm{T}}$ e, **H** é a matriz de projeto formada pelas ativações h_j das unidades radiais para os diferentes padrões de treinamento $\boldsymbol{\xi}_n$, $\mathbf{H} = \left[\mathbf{h}_1 \quad \mathbf{h}_2 \quad \ldots \quad \mathbf{h}_m\right]$ na qual $\mathbf{h}_j = \left[h_j(1) \quad h_j(2) \quad \ldots \quad h_j\left(n_p\right)\right]^{\mathrm{T}}$. Ou seja

$$\mathbf{H} = \begin{bmatrix} h_1(1) & h_2(1) & \cdots & h_m(1) \\ h_1(2) & h_2(2) & \cdots & h_m(2) \\ \vdots & \vdots & \ddots & \vdots \\ h_1(n_p) & h_2(n_p) & \cdots & h_m(n_p) \end{bmatrix}.$$

Um problema que surge é a escolha do parâmetro de regularização λ. Se um valor de λ é escolhido muito pequeno, significa que existirá uma pequena penalização, o que pode gerar uma baixa generalização já que o modelo se ajustará firmemente aos dados de treinamento (*overfitting*). Por outro lado, se um valor de λ muito grande é escolhido, existirá uma grande penalização, significando que o modelo pode se ajustar mal aos dados de treinamento. Critérios para seleção de modelos podem ser utilizados para encontrar um valor de λ o mais próximo possível do valor ótimo. O valor escolhido é aquele associado com o menor erro de predição. O erro de predição é a estimativa do desempenho do modelo treinado para padrões desconhecidos (não-treinados). Existem vários critérios para estimar o erro de predição (ou métodos para seleção de modelos), entre eles: *leave-one-out cross-validation*, *generalized cross-validation*, erro de predição final e critério de informação Bayesiano.

Neste trabalho, o critério *generalized cross-validation* (GCV) é utilizado na escolha do parâmetro de regularização pois, para este fim, a sua forma de otimização é simples [GOLUB *et al*., 1979]. Para a utilização do critério GCV, a matriz de projeção precisa ser calculada. Esta é uma matriz quadrada que projeta vetores de um espaço n_p-dimensional (n_p é o número de padrões de treinamento) perpendicular para um subespaço m-dimensional (m é o número de unidades radiais) gerado pelo modelo e representa o erro entre a saída desejada (que é um vetor n_p-dimensional para cada neurônio de saída) e a predição (vetor m dimensional) pelo princípio dos mínimos quadráticos sem a penalização dos pesos (veja Apêndice A3). Uma observação importante é que em *ridge regression*, a matriz de projeção, embora continuará a ser chamada assim, não é exatamente a projeção dos vetores do espaço n_p-dimensional perpendicular para um subespaço m-dimensional, já que a penalização diminuiu o número efetivo de parâmetros (m). A matriz de projeção dada pelo princípio do mínimo quadrático é

$$\mathbf{P} = \mathbf{I}_{n_p} - \mathbf{H}\mathbf{A}^{-1}\mathbf{H}^{\mathrm{T}} \tag{33}$$

na qual $\mathbf{A}^{-1}$ é a matriz de variância e a matriz identidade $\mathbf{I}$ tem dimensão n_p (número de padrões). A matriz de variância é dada por:

$$\mathbf{A}^{-1} = \left(\mathbf{H}^{\mathrm{T}}\mathbf{H} + \boldsymbol{\Lambda}\right)^{-1} \tag{34}$$

na qual $\boldsymbol{\Lambda}$ é uma matriz quadrada com dimensão m com os parâmetros de regularização na diagonal principal e zero nos elementos restantes. A predição da variância do erro pelo critério GCV (ou erro GCV), considerando-se que a rede RBF tem apenas uma saída (q=1), é dada por (Apêndice A4)

$$\hat{\sigma}^2_{\mathrm{GCV}} = \frac{n_p \boldsymbol{\psi}^{\mathrm{T}}\mathbf{P}^2\boldsymbol{\psi}}{\left(\mathrm{tr}\left(\mathbf{P}\right)\right)^2} \tag{35}$$

na qual tr ($\mathbf{P}$) denota o traço da matriz de projeção.

Como em regularização os critérios para seleção de modelos dependem não-linearmente de λ, algum tipo de otimização não-linear é necessária. Qualquer das técnicas de otimização não-linear, como por exemplo o método de Newton, podem ser usadas. Alternativamente [ORR, 1995 a], pode-se explorar o fato de que quando a derivada do erro GCV é fixada em zero, a equação resultante pode ser manipulada de tal modo que somente $\hat{\lambda}$ aparece no lado direito da equação (Apêndice A5). Considerando-se que a rede RBF tem apenas uma saída, então

$$\hat{\lambda} = \frac{\boldsymbol{\psi}^{\mathrm{T}}\mathbf{P}^2\boldsymbol{\psi}\,\mathrm{tr}\left(\mathbf{A}^{-1} - \hat{\lambda}\,\mathbf{A}^{-2}\right)}{\hat{\boldsymbol{\omega}}^{\mathrm{T}}\mathbf{A}^{-1}\hat{\boldsymbol{\omega}}\,\mathrm{tr}\left(\mathbf{P}\right)}. \tag{36}$$

Esta não é uma solução, e sim uma fórmula de reestimação já que o lado direito da equação depende de $\hat{\lambda}$. Portanto, um valor inicial de $\hat{\lambda}$ tem que ser escolhido e usado no lado direito da equação acima. Isto leva a uma nova estimativa e o processo pode ser repetido até que algum critério de convergência seja alcançado.

2.2.1.2. *Local ridge regression* (LRR):

Ao invés de todos os pesos serem igualmente tratados através de um único termo de regularização, eles podem ser tratados separadamente através de vários termos, cada um associado a uma unidade radial. Assim, a nova função de custo para o neurônio de saída *k* fica

$$C_k = \sum_{n=1}^{n_p} \left(\psi_k (n) - \hat{\psi}_k (n) \right)^2 + \sum_{j=1}^{m} \lambda_j \, \omega_{kj}^2 \, . \qquad (37)$$

A nova matriz de pesos ótima é dada por (veja Apêndice A2)

$$\hat{\mathbf{\Omega}} = \left(\mathbf{H}^{\mathrm{T}} \mathbf{H} + \mathbf{\Lambda} \right)^{-1} \mathbf{H}^{\mathrm{T}} \mathbf{\Psi} \qquad (38)$$

na qual $\mathbf{\Lambda}$ é uma matriz diagonal formada pelos parâmetros de regularização λ_j.

Em geral, não existe nada de local nesta forma de decaimento de pesos. No entanto, se forem usadas funções de base locais como as funções radiais, então, a suavidade produzida por esta forma de *ridge regression* é controlada de maneira local pelos parâmetros de regularização individuais [ORR, 1995 b]. Pode-se, portanto, adaptar a suavidade da função às condições locais. Este é o porquê do nome *local ridge regression* (LRR). O método GRR tem dificuldades em reproduzir funções que têm suavidade significativamente diferente nas diversas partes do espaço de entradas.

Como no método GRR, os parâmetros de regularização devem ser otimizados através de um critério para seleção de modelos. O critério de seleção depende principalmente da matriz de projeção (Apêndice A3) e, portanto, será preciso deduzir a sua dependência aos parâmetros de regularização individuais. O critério de seleção de modelo que será utilizado é o GCV.

Em contraste com o método GRR, para o método LRR existe uma solução para os valores ótimos dos parâmetros de regularização e, portanto, a reestimação não é necessária. O problema é que existem *m*-1 outros parâmetros para otimizar e cada vez que um parâmetro é otimizado, os valores ótimos dos outros parâmetros mudam. Como solução para este problema todos os parâmetros são otimizados juntos como em um tipo de reestimação, fazendo um por vez e repetindo até que o erro GCV convirja.

Quando o valor calculado $\lambda_j = \infty$, a *j*-ésima unidade radial pode ser removida, pois, se for calculada a matriz de projeção com $\lambda_j = \infty$, esta será igual a matriz de projeção sem a unidade radial *j*. Na prática, especialmente se a rede é inicialmente muito flexível (alta variância e/ou pequeno *bias*), vários valores ótimos dos parâmetros de regularização serão iguais a ∞ e LRR pode ser usado para "podar" unidades radiais desnecessárias. O Apêndice A6 traz o método para o cálculo dos valores ótimos dos parâmetros de regularização.

Como qualquer tipo de otimização não-linear, o algoritmo pode cair em um mínimo local dependendo das condições iniciais. É aconselhável, portanto, que valores iniciais dos parâmetros de regularização sejam escolhidos (por exemplo, iniciar com o parâmetro global calculado pelo método GRR), ao invés de iniciá-los com valores aleatórios.

2.2.2. Treinamento da rede RBF via *forward selection* (FS)

Uma alternativa aos métodos que controlam o balanço entre *bias* e a variância (como os métodos que utilizam regularização) é comparar os modelos formados por diferentes subconjuntos de unidades radiais subtraídos de um mesmo conjunto de treinamento fixo. Deste modo, a flexibilidade é variada mudando-se o número de parâmetros adaptativos da rede RBF (como os pesos e tamanho do campo receptivo das unidades radiais). Isto é chamado de seleção de subconjuntos em estatística [RAWLINGS, 1988]. Seleção de subconjuntos é normalmente utilizada para a identificação de subconjuntos de funções fixadas de variáveis independentes que podem modelar a maior variação das variáveis dependentes. Neste sentido, achar o melhor subconjunto é intratável, sendo necessário a utilização de procedimentos heurísticos. Um destes procedimentos heurísticos chama-se *forward selection* (FS).

CHEN *et al.* (1991) utilizaram FS para escolher os centros das unidades radiais em redes RBF a partir dos padrões do conjunto de treinamento. FS começa com um conjunto vazio e adiciona uma unidade radial por vez - aquela que mais reduz a soma dos erros quadráticos [MILLER, 1990]. Ou seja, considerando-se apenas uma saída (*q*=1), a unidade radial selecionada é aquela cuja diferença abaixo (veja Apêndice A7) é maior

$$\hat{C}_M - \hat{C}_{M+1} = \frac{\left(\boldsymbol{\psi}^{\mathrm{T}}\mathbf{P}_M\mathbf{h}_n\right)^2}{\mathbf{h}_n^{\mathrm{T}}\mathbf{P}_M\mathbf{h}_n} \qquad (39)$$

na qual M = 1, ..., m (m é o número de unidades radiais escolhidas pelo método) é o número de unidades radiais escolhidas até o momento, $\hat{C}_M$ é a soma dos erros quadráticos antes de a n-ésima (n = 1, ..., n_p) unidade radial ser acrescentada, $\hat{C}_{M+1}$ é a soma dos erros quadráticos depois de a n-ésima unidade radial ser acrescentada, $\mathbf{h}_n$ é a n-ésima coluna da matriz de projeto **H** original, ou seja, o vetor com as n_p ativações da unidade radial n e $\mathbf{P}_M$ é a matriz de projeção antes de a n-ésima unidade radial ser acrescentada.

Se a n-ésima unidade radial é acrescentada, então o vetor $\mathbf{h}_n$ é inserido na última coluna da nova matriz de projeto $\mathbf{H}_M$. Ao final, a nova matriz de projeto terá m colunas. O algoritmo FS pode se tornar mais eficiente usando-se uma técnica conhecida como *orthogonal least squares* (OLS) [CHEN *et al*., 1991], que garante que cada novo vetor $\mathbf{h}_n$ correspondente ao centro adicionado à matriz de projeto seja ortogonal às colunas anteriores. Isto simplifica a função de custo e torna o algoritmo mais rápido. Originalmente, a seleção de centros deve parar quando um *threshold* na variância da soma dos erros quadráticos é ultrapassado. Um método mais eficiente é empregar um critério que estima o erro de predição, tal qual o GCV, para finalizar a seleção das unidades radiais [ORR, 1995 a]. Apesar da soma dos erros quadráticos $\hat{C}$ nunca aumentar quando uma nova unidade radial for acrescentada, o erro GCV eventualmente irá parar de decrescer e começará a crescer, indicando um sobreajuste. Este é o ponto em que o algoritmo deve parar a seleção. A matriz de pesos ótima é dada por (veja Apêndice A2)

$$\hat{\mathbf{\Omega}} = \left(\mathbf{H}^{\mathrm{T}}\,\mathbf{H}\right)^{-1}\mathbf{H}^{\mathrm{T}}\mathbf{\Psi} \tag{40}$$

na qual a matriz de projeção **H** é formada pelas ativações das m unidades radiais selecionadas apresentando-se os padrões do conjunto de treinamento.

2.2.3. Treinamento da rede RBF via mapa auto-organizável de Kohonen (MAOK)

O último método para treinamento da rede RBF utiliza o mapa auto-organizável de Kohonen (MAOK) para a seleção das unidades radiais da rede RBF. O uso do MAOK para o treinamento de redes RBF não é um enfoque novo. Em OJALA & VUORIMAA (1995) por exemplo, o MAOK é usado para a seleção inicial dos centros das unidades radiais e, então, um algoritmo *Learning Vector Quantization* (LVQ2.1) modificado [KOHONEN, 1995] é utilizado para sintonizar todos os parâmetros da rede (como os centros e os pesos). Neste trabalho, algumas mudanças serão feitas para adequar o MAOK para o treinamento da rede RBF aplicada no problema de DDF. Apesar deste algoritmo ser não-supervisionado (ou seja, não necessita do conhecimento das saídas desejadas), ele será aplicado em um problema supervisionado. Assim, aproveitando as características do problema, o conjunto de treinamento será separado de acordo com as diferentes classes. Este procedimento é adotado para evitar que padrões de treinamento pertencentes à diferentes classes sejam sintonizados em uma mesma unidade radial. Portanto, o algoritmo descrito abaixo deve ser repetido para cada classe.

Inicialmente, todos os padrões de treinamento são considerados como centros das unidades radiais. Utilizando os padrões pertencentes a uma determinada classe, calcula-se as ativações das unidades radias (Eq. 27) para cada padrão de treinamento (da mesma classe) e a unidade com a maior ativação é selecionada de acordo com

$$h_c(t) = \max_j \left\{ h_j(t) \right\} \tag{41}$$

na qual c é a unidade radial selecionada, j=1,...,m_k (m_k é o número de padrões da classe k), k=1,...,q (q é o número de classes) e t=1,...,$t_{\max}$ é o índice do tempo discreto (t_{max} deve ser múltiplo de m_k). Em cada amostra t, um padrão diferente do subconjunto de treinamento deve ser apresentado (para que o algoritmo apresente uma convergência conveniente, o subconjunto de treinamento deve ser apresentado diversas vezes). O passo seguinte é atualizar as posições dos centros das unidades radiais de acordo com

$$\boldsymbol{\mu}_j(t+1) = \boldsymbol{\mu}_j(t) + \alpha(t)\beta(t)\left[\boldsymbol{\xi}(t) - \boldsymbol{\mu}_j(t)\right] \tag{42}$$

na qual $\alpha(t)$ é uma função de decaimento no tempo que define a taxa de aprendizagem e $\beta(t)$ é uma função da distância vetorial entre o centro da unidade radial j ($\boldsymbol{\mu}_j$) e o centro da unidade radial selecionada ($\boldsymbol{\mu}_c$). Aqui, esta função é dada por

$$\beta(t) = \begin{cases} \dfrac{1}{1+\left\|\mathbf{R}^{-1}\left(\boldsymbol{\mu}_c - \boldsymbol{\mu}_j\right)\right\|^2}, & \text{se } \left\|\mathbf{R}^{-1}\left(\boldsymbol{\mu}_c - \boldsymbol{\mu}_j\right)\right\| < \sigma(t), \\ 0, & \text{se } \left\|\mathbf{R}^{-1}\left(\boldsymbol{\mu}_c - \boldsymbol{\mu}_j\right)\right\| \geq \sigma(t) \end{cases} \tag{43}$$

na qual a função de espalhamento $\sigma(t)$ decai com o tempo e define o tamanho da vizinhança ao redor do centro da unidade radial selecionada c ($\boldsymbol{\mu}_c$). Se o número de iterações (t_{max}) é suficientemente grande e os parâmetros de treinamento são apropriadamente escolhidos, os centros das unidades radiais em um mesmo aglomerado (*cluster*) deverão se mover até as proximidades do centro do aglomerado (ou seja, no local em que a média das ativações das unidades radiais para todos os padrões de um mesmo aglomerado seja a maior). Após o treinamento, vários centros de um mesmo aglomerado estarão na mesma posição ou em posições muito próximas. Pode-se, portanto, juntar estas unidades radiais em uma única, já que as ativações de duas unidades radiais com o mesmo centro são iguais. Agindo desta forma, a complexidade da rede RBF diminui pois o número de parâmetros adaptativos (pesos) decresce. Este procedimento também é importante para evitar problemas de singularidade na matriz usada para determinar os pesos ótimos (Eq. 40). Se existem dois centros muito próximos, a inversa da matriz da Eq. 40 sofre problemas de mal condicionamento. A união das unidades radiais com centros muito próximos é feita verificando se a norma da distância entre duas unidades radiais é muito pequena. Se a resposta for afirmativa, uma unidade radial é retirada.

Repetindo o procedimento descrito acima para todas as classes, as unidades radiais de cada subconjunto são agrupadas em uma única rede RBF e a matriz de pesos ótima é calculada através da Eq. (40). A seguir, um exemplo simples será apresentado para demostrar a eficiência deste método.

Exemplo 2: Considere que se deseja classificar padrões inseridos em um espaço bidimensional em duas classes. Para a construção do classificador (treinamento) são disponíveis 20 padrões de cada classe. Os padrões da classe 1 são gerados aleatoriamente (distribuição uniforme) com média=$[0{,}5\ 0{,}5]^T$ e variância=$[0{,}01\ 0{,}09]^T$. Os padrões da classe

2 são gerados aleatoriamente (distribuição uniforme) com média=[0 0]^T e variância=[0,01 0,09]^T. Dois métodos para treinamento da rede RBF são utilizados: o algoritmo MAOK e o algoritmo FS utilizando um simples *threshold* na variância do erro como critério de parada. O tamanho do campo receptivo (diagonal da matriz **R**) da rede RBF é escolhido como [0,2 0,6]^T. O algoritmo FS seleciona 2 unidades radiais centradas em [0,578 0,589]^T e [0,047 0,083]^T, como pode ser visto na Figura 12. Note que tais centros são escolhidos a partir das posições dos padrões de treinamento. O algoritmo MAOK seleciona 2 unidades radiais centradas em [0,513 0,560]^T e [0,015 0,039]^T, como pode ser visto na Figura 13. Tais posições são obtidas através da convergência dos padrões em direção aos centros de cada aglomerado.

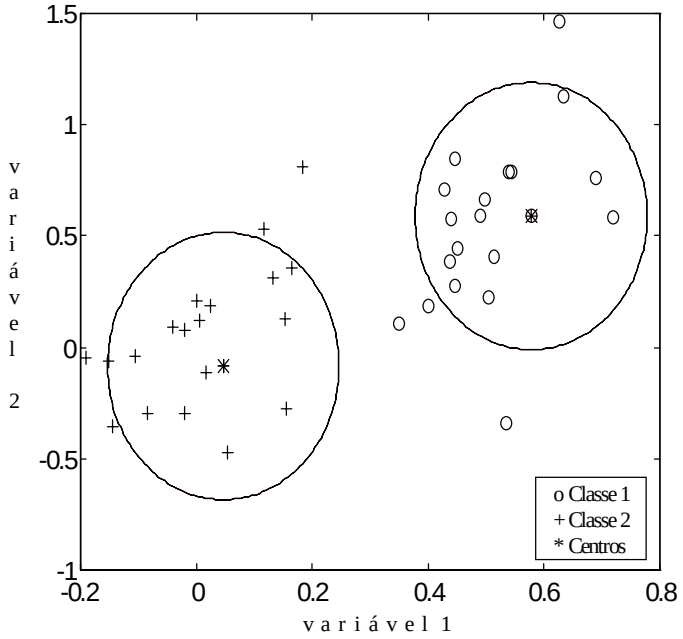

Figura 12. Padrões de treinamento e campos receptivos criados pelas unidades radiais de uma rede RBF treinada pelo algoritmo FS.

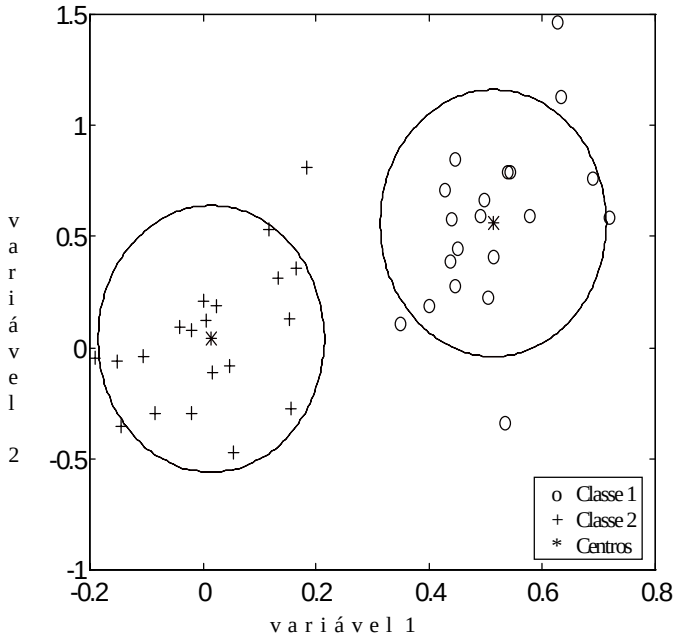

Figura 13. Padrões de treinamento e campos receptivos criados pelas unidades radiais de uma rede RBF treinada pelo algoritmo MAOK.

A seguir, foram gerados 200 padrões com as mesmas características anteriores para teste das redes RBF treinadas pelos dois métodos. As Figuras 14 e 15 mostram os padrões de treinamento e os campos receptivos formados pelas redes RBF treinadas pelos dois algoritmos.

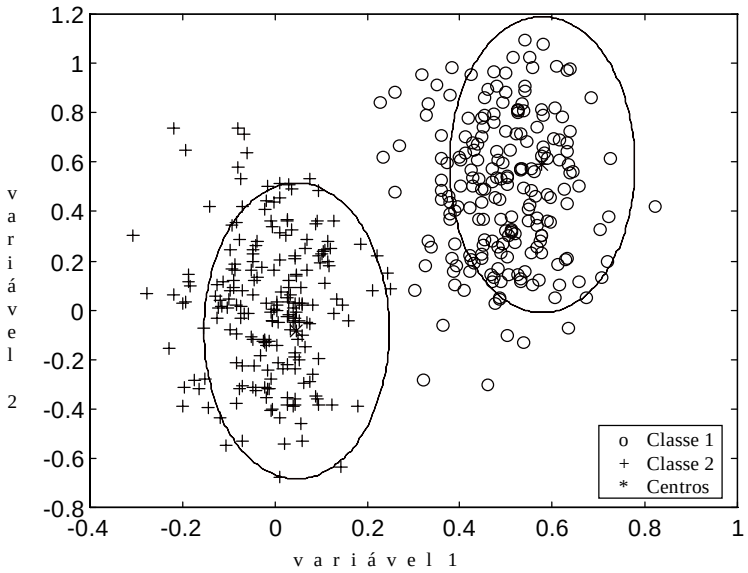

Figura 14. Padrões de teste e campos receptivos criados pelas unidades radiais de uma rede RBF treinada pelo algoritmo FS.

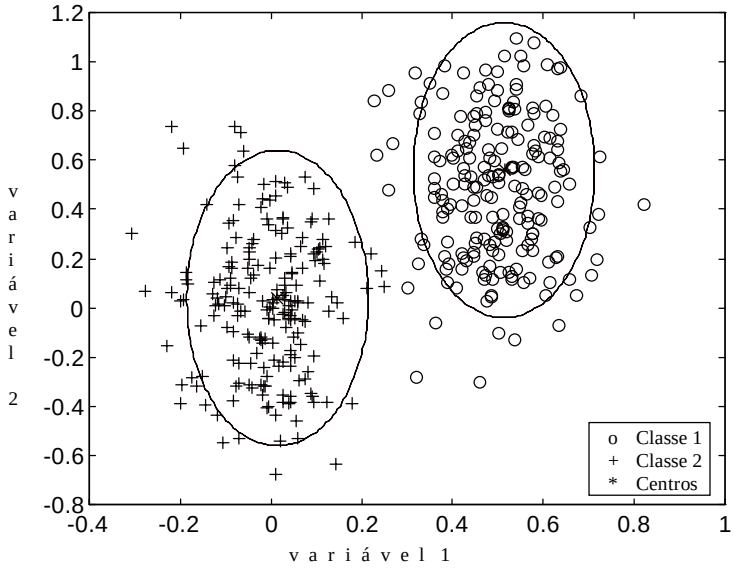

Figura 15. Padrões de teste e campos receptivos criados pelas unidades radiais de uma rede RBF treinada pelo algoritmo MAOK.

Capítulo 3

O robô manipulador

Robôs manipuladores (ou robôs industriais) estão sendo cada vez mais utilizados em sistemas de engenharia. Entre as diversas aplicações, pode-se citar o seu uso cada vez mais frequente em processos de produção, exploração espacial, medicina e ambientes hostis (tais como usinas nucleares e fundo do mar).

Um robô manipulador é definido como uma máquina com características significantes de versatilidade e flexibilidade [SCIAVICCO & SICILIANO, 1996]. De acordo com uma definição amplamente aceita feita pelo *Robot Institute of America*, datada de 1980, "um robô é um manipulador multifuncional projetado para mover materiais, partes, ferramentas ou artifícios especializados através de movimentos programados variáveis para o desempenho de uma variedade de tarefas". Um robô manipulador é constituído por

- manipulador (ou estrutura mecânica): constituído por um conjunto de elos conectados através de articulações (ou juntas), um pulso e um efetuador projetado para realizar a tarefa requerida pelo robô,
- atuadores: responsáveis pela movimentação do manipulador através de forças aplicadas direta ou indiretamente nas juntas (através, por exemplo, de engrenagens ou correias); as forças são aplicadas por motores elétricos, pneumáticos ou hidráulicos,
- sensores: indicam o *status* do manipulador e,
- sistema de controle (computador): permite o controle e a supervisão do manipulador.

As juntas de um manipulador podem ser rotativas, que permitem movimentos de rotação das diferentes partes do robô (elos, punho ou efetuador), ou prismáticas, que permitem movimentos de translação. Geralmente, sensores de velocidade e/ou posição são colocados nas juntas do robô.

Diferente do braço e da mão humana, que juntos possuem cerca de 30 graus de liberdade, muitos robôs trabalham em três dimensões com apenas 6 graus de liberdade, o número mínimo de graus necessários para posicionar e orientar o efetuador em qualquer lugar de um espaço tridimensional [VISINSKY *et al*., 1994]. Robôs com um número maior de juntas podem ser configurados de diferentes modos para realizar o mesmo posicionamento de um objeto ligado ao efetuador.

A Figura 16 mostra um robô manipulador planar com dois elos rígidos. Este manipulador será posteriormente utilizado para a simulação do sistema de DDF. Os ângulos θ_1 e θ_2 indicam a posição, respectivamente, da junta 1 e da junta 2. Os tamanhos dos elos são representados por l_1 e l_2. A seguir, o modelo matemático do robô manipulador será descrito. Na seção 3.2, será comentado o problema das falhas em tais sistemas e na seção 3.3 serão apresentados alguns métodos para DDF em robôs manipuladores. Considerando que a falha já foi detectada e isolada por um sistema de DDF, em alguns casos, pode-se reconfigurar a ação de controle para permitir tolerância a falha. Alguns mecanismos para reconfiguração de controle após falha em robôs manipuladores serão discutidas na seção 3.4.

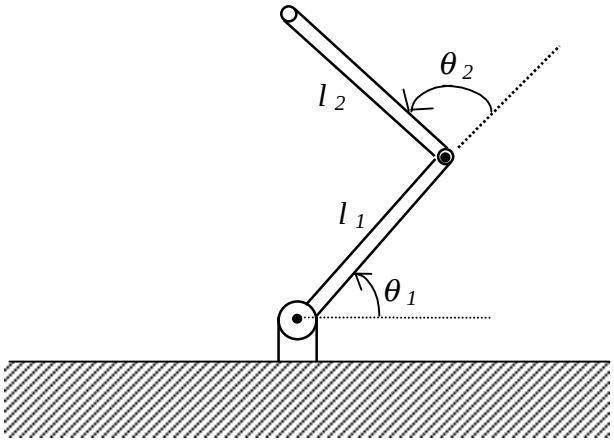

Figura 16. Robô manipulador com dois elos rígidos e juntas rotativas.

3.1. MODELO MATEMÁTICO DO ROBÔ MANIPULADOR SEM FALHAS

Considere um robô manipulador rígido livre de falhas com atuadores em cada grau de liberdade. A sua dinâmica é dada por [CRAIG, 1988]

$$\boldsymbol{\tau} = \mathbf{M}(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + \mathbf{v}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) + \mathbf{g}(\boldsymbol{\theta}) + \mathbf{z}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, t) + \mathbf{d}(t) \qquad (44)$$

na qual $\boldsymbol{\theta}$ é o vetor dos ângulos das juntas (n_g -dimensional), n_g é o número de graus de liberdade, t é o índice do tempo, $\boldsymbol{\tau}$ é o vetor dos torques (n_g -dimensional), $\mathbf{M}$ é a matriz de inércia (n_g x n_g), $\mathbf{v}$ é o vetor dos termos centrífugos e de Coriolis, $\mathbf{g}$ é o vetor dos termos gravitacionais, $\mathbf{z}$ é o vetor dos termos de fricção dinâmica e estática e $\mathbf{d}$ é o vetor dos distúrbios externos não-correlacionados.

A matriz de inércia é simétrica e positiva definida para qualquer robô manipulador. Utilizando-se o equacionamento anterior, tem-se a seguinte equação de estados para o manipulador livre de falhas

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\boldsymbol{\theta}} \\ \ddot{\boldsymbol{\theta}} \end{bmatrix} = \begin{bmatrix} \dot{\boldsymbol{\theta}} \\ \mathbf{M}(\boldsymbol{\theta})^{-1}\left[\boldsymbol{\tau} - \mathbf{v}(\boldsymbol{\theta},\dot{\boldsymbol{\theta}}) - \mathbf{g}(\boldsymbol{\theta}) - \mathbf{z}(\boldsymbol{\theta},\dot{\boldsymbol{\theta}},t) - \mathbf{d}(t)\right] \end{bmatrix} \tag{45}$$

na qual o vetor de estados $\mathbf{x} = \begin{bmatrix} \boldsymbol{\theta} & \dot{\boldsymbol{\theta}} \end{bmatrix}^{\mathrm{T}}$ ou seja $\mathbf{x} = \begin{bmatrix} \theta_1 & \cdots & \theta_{n_g} & \dot{\theta}_1 & \cdots & \dot{\theta}_{n_g} \end{bmatrix}^{\mathrm{T}}$.

Exemplificando o sistema acima para um robô planar com dois elos rígidos se movendo no plano vertical, como na Figura 16, tem-se

$$\mathbf{M}(\boldsymbol{\theta}) = \begin{bmatrix} m_2 l_2^2 + 2m_2 l_1 l_2 c_2 + l_1^2 (m_1 + m_2) & m_2 l_2^2 + m_2 l_1 l_2 c_2 \\ m_2 l_2^2 + m_2 l_1 l_2 c_2 & m_2 l_2^2 \end{bmatrix},$$

$$\mathbf{v}(\boldsymbol{\theta},\dot{\boldsymbol{\theta}}) = \begin{bmatrix} -m_2 l_1 l_2 s_2 \dot{\theta}_2^2 - 2m_2 l_1 l_2 s_2 \dot{\theta}_1 \dot{\theta}_2 \\ m_2 l_1 l_2 s_2 \dot{\theta}_2^2 \end{bmatrix},$$

$$\mathbf{g}(\boldsymbol{\theta}) = \begin{bmatrix} m_2 l_2 g\, s_{12} + (m_1 + m_2) l_1 g\, s_1 \\ m_2 l_2 g\, s_{12} \end{bmatrix},$$

$$s_1 \equiv \operatorname{sen}(\theta_1), \quad s_2 \equiv \operatorname{sen}(\theta_2), \quad s_{12} \equiv \operatorname{sen}(\theta_1 + \theta_2), \quad c_2 \equiv \cos(\theta_2)$$

nas quais g é a aceleração da gravidade e m_1 , m_2 são as massas dos elos (consideradas no fim de cada elo).

O sistema de controle comanda o robô manipulador para se mover através do espaço de trabalho baseado em um plano ou uma trajetória desejada. O plano é tipicamente desenvolvido usando um algoritmo de cinemática inversa que calcula o próximo ponto desejado dada a configuração atual do robô manipulador. Existem vários métodos para se controlar o robô definido pela Eq. (44). Um dos mais utilizados é o método do torque calculado. Para o cálculo do torque nesse método, o modelo matemático completo do robô

manipulador precisa ser conhecido. Erros de modelagem podem comprometer a eficácia deste método. No entanto, existem métodos em que o conhecimento completo do modelo matemático não é necessário, como por exemplo, os métodos de controle recursivo [OSTOJIC, 1996].

3.2. FALHAS EM ROBÔS MANIPULADORES

Falhas em robôs manipuladores podem causar movimentos descontrolados dos elos, podendo causar sérios danos ao robô e ao ambiente de trabalho. Por exemplo, uma falha no sensor de velocidade da segunda junta de um robô industrial com 6 graus de liberdade pode fazer com que o sistema de controle aplique nessa junta um alto valor de torque. Esta ação pode levar o robô a, por exemplo, bater no chão ou em outros objetos do ambiente de trabalho. Entre as falhas mais frequentes em robôs manipuladores podem-se citar as falhas nos sensores, nos atuadores, na fonte de alimentação e no sistema de controle [VISINSKY *et al*., 1994].

Os sensores dos robôs manipuladores estão sujeitos a diversas falhas. Um sensor pode ter um fio rompido, permanecer enviando o mesmo sinal ou enviar o sinal errado. Se um sensor está com falhas e o sistema não conseguir detectá-las, o controlador irá receber informações incorretas sobre a junta e irá tomar as decisões de controle erradas.

Atuadores em robôs manipuladores são obviamente críticos. Podem, por exemplo, travar a junta em uma única posição ou não aplicar força alguma. Em sistemas tolerantes a falhas reconfiguráveis, os atuadores devem ter a habilidade de travar a junta na posição em que se encontra. Se o atuador falha e trava no lugar, ele não irá afetar as outras juntas através do acoplamento. No entanto, o atuador irá parar de contribuir para a realização da tarefa requerida.

Pelo que foi visto, sistemas de DDF em robôs manipuladores são muito importantes para manter a sua confiabilidade e segurança. Quando a falha ocorre, ela deve ser detectada de modo rápido e devem ser geradas atitudes para preservar a segurança do equipamento e do ambiente de trabalho. Através da DDF, pode-se, também, obter sistemas tolerantes a falhas reconfiguráveis. A idéia da tolerância a falhas via DDF diz respeito à habilidade do sistema em detectar e isolar a falha e, então, tomar atitudes para que continue funcionando.

O interesse em robôs manipuladores com tolerância a falhas tem se expandido nos últimos anos. Um dos motivos é a crescente aplicação de robôs em áreas como exploração espacial, medicina, plantas nucleares e outros ambientes hostis. A periculosidade e/ou a

distância destes ambientes torna o envio de seres humanos inviável para reparar um robô com falhas. No caso de ambientes espaciais, falhas em robôs sem tolerância podem fazer com que a missão aborte. Em medicina, um robô com falhas pode matar um paciente ou gerar sequelas permanentes. Em plantas nucleares, o mau funcionamento de robôs pode causar acidentes irreversíveis com alta periculosidade. Tais fatos justificam a formulação de métodos de DDF em robôs manipuladores.

Considerando agora o efeito da falha na dinâmica do robô manipulador, a equação de estados após a falha pode ser dada por

$$\dot{\mathbf{x}}_{\phi} = \begin{bmatrix} \dot{\boldsymbol{\theta}} \\ \ddot{\boldsymbol{\theta}} \end{bmatrix} = \begin{bmatrix} \dot{\boldsymbol{\theta}} \\ \mathbf{M}(\boldsymbol{\theta})^{-1}\left[\boldsymbol{\tau} - \mathbf{v}(\boldsymbol{\theta},\dot{\boldsymbol{\theta}}) - \mathbf{g}(\boldsymbol{\theta}) - \mathbf{z}(\boldsymbol{\theta},\dot{\boldsymbol{\theta}},t) - \mathbf{d}(t)\right] + \boldsymbol{\phi}(\boldsymbol{\theta},\dot{\boldsymbol{\theta}},\boldsymbol{\tau},t) \end{bmatrix} \qquad (46)$$

na qual $\boldsymbol{\phi}$ é o vetor de falhas (n_g - dimensional).

3.3. MÉTODOS PARA DETECÇÃO E DIAGNÓSTICO DE FALHAS EM ROBÔS MANIPULADORES

Em geral, a maioria dos métodos utilizados para DDF em robôs manipuladores emprega redundância analítica (veja Capítulo 1). Portanto, utilizam o modelo matemático do robô manipulador. A diferença básica entre os métodos se encontra na análise dos resíduos. Analisar simplesmente se o resíduo é diferente de zero é inviável para a detecção da falha. Este procedimento leva a um grande número de alarmes falsos, já que os erros de modelagem inerentes ao sistema sem falhas aumentam devido a linearização das equações do sistema e às incertezas dos parâmetros do modelo.

O que acontece tipicamente nos sistemas de DDF para robôs manipuladores é que bandas de detecção (*thresholds*) fixas são estipuladas para mascarar tais incertezas. Geralmente, a banda de detecção é determinada empiricamente, tomando-se cuidado para que seu valor não seja grande o suficiente para esconder as falhas. No entanto, os efeitos de erros de modelagem flutuam dinamicamente conforme o manipulador se move e quando ocorrem falhas. Desta forma, os alarmes falsos podem aparecer conforme o robô se move, pois a banda de detecção fixa não é projetada para todas as situações dinâmicas possíveis. Como exemplo prático deste problema, pode-se citar o sistema de DDF utilizado no robô manipulador do Ônibus Espacial Americano na missão STS-49 (maio de 1992). Neste vôo, uma sequência de 13 alarmes falsos foram disparados porque condições operacionais

especiais, tais como entradas do controlador manual e carga no manipulador não usuais, não foram levadas em consideração na modelagem do sistema [VISINSKY *et al*., 1994]. Assim, as bandas de detecção devem ser variáveis para que os efeitos dos erros de modelagem possam ser superados.

Vários métodos foram criados para superar os problemas dos sistemas dinâmicos incertos. Dentre esses, pode-se citar o RMI (*reachable measurement intervals*) desenvolvido por HORAK (1988) para sistemas de aeronaves. RMI provê condições para se achar os extremos possíveis das saídas medidas para cada entrada e estado do sistema sob condições livres de falhas, dadas faixas de variações paramétricas nos coeficientes das matrizes das equações dinâmicas. RMI, contudo, é concebido para sistemas com dinâmica linear e requer procedimentos especiais custosos para tratar sistemas dinâmicos não-lineares.

Para robôs manipuladores, um outro método baseado nos conceitos de RMI foi desenvolvido. É o chamado ThMB (*model-based threshold algorithm*) que utiliza a idéia de se encontrar a máxima variância possível devido às faixas dos erros paramétricos em cada iteração [VISINSKY *et al*., 1995]. Utilizando tais dados, bandas de detecção dependentes dos estados e das entradas são construídas para se alcançar robustez na análise dos resíduos. Este método explora as similaridades entre RMI e redundância analítica.

Um outro enfoque é utilizar técnicas de Inteligência Artificial para criar bandas de detecção adaptativas. SCHNEIDER & FRANK (1996) utilizaram lógica difusa (seção 1.3.2) e NAUGHTON *et al*. (1996) utilizaram um MLP com treinamento por retropropagação para produzir bandas de detecção variáveis. O trabalho de SCHNEIDER & FRANK preocupa-se basicamente com os erros de modelagem causados pela fricção. Como a fricção é difícil de ser modelada em robôs manipuladores, construiu-se um conjunto de regras difusas para variar as bandas de detecção em função dos dados de velocidade e aceleração nas juntas. Produzem, assim, bandas de detecção dinâmicas. Neste caso, o conhecimento de um especialista é necessário para a confecção das regras. Já NAUGHTON *et al*. utilizaram um MLP com treinamento por retropropagação para a classificação dos resíduos. Treinando o MLP com dados do vetor de resíduos como entrada e das falhas como saída desejada, a RNA pôde construir bandas de detecção variáveis. No entanto, o sistema de DDF neste caso consegue apenas detectar as falhas em um conjunto de trajetórias limitado (aquele usado no treinamento) apesar de os autores acreditarem que novas trajetórias podem ser utilizadas se o conjunto de treinamento empregado for maior.

Todos os métodos citados acima utilizam o modelo matemático do manipulador. Empregam técnicas robustas para a geração de resíduos (através, por exemplo, de

observadores robustos) e/ou para a análise dos resíduos (através de bandas de detecção adaptativas). Todos os métodos descritos acima têm as seguintes características: 1) o modelo nominal do sistema é considerado linear e, 2) as falhas são modeladas como entradas aditivas externas (ou seja, dependentes apenas do tempo e não dos estados e das entradas do sistema). Apesar de ser conveniente do ponto de vista analítico o estudo de problemas de DDF em estruturas lineares, a dinâmica do manipulador é inerentemente não-linear e a maioria das falhas reais são funções não-lineares dos estados e das entradas [VEMURI & POLYCARPOU, 1997].

Um enfoque para DDF em robôs manipuladores diferente dos utilizados anteriormente é o desenvolvido por VEMURI & POLYCARPOU (1997), no qual uma RNA é empregada para mapear a função de falha (ver Eq. 46). Seu método é interessante pois a falha não é considerada como uma entrada aditiva externa e sim como uma função das variáveis do sistema. A tarefa da RNA é mapear a função da falha, tendo como entradas as velocidades e as posições das juntas. Assim, o sistema pode não só detectar, mas também reproduzir a função da falha. O método utiliza um observador para, baseado no modelo matemático e na função de falha estimada pela RNA, gerar uma estimativa das variáveis medidas. O erro entre o valor estimado e o valor medido é utilizado para o ajuste dos pesos da rede. O trabalho não propõe nenhum esquema para isolação da falha, propondo que algum método de classificação de padrões deva ser utilizado. O maior problema deste método é que erros de modelagem podem comprometer, durante o treinamento da RNA, o mapeamento do vetor de falhas.

3.4. TOLERÂNCIA A FALHAS EM ROBÔS MANIPULADORES

Dependendo da falha que afeta o manipulador, a ação de controle pode ser reconfigurada para permitir que o robô execute sua tarefa tal qual originalmente foi proposta. A maioria dos trabalhos em tolerância de falhas mecânicas em robôs tem se concentrado naqueles algoritmos que, após a falha, ativam a parte duplicada [VISINSKY *et al*., 1994]. Por exemplo, se um motor falha, o sistema detecta e isola tal falha, desliga o componente faltoso e ativa o motor redundante. Os problemas citados para tais métodos são os mesmos do sistema de DDF via redundância física, ou seja, incremento de custo, tamanho e potência de alimentação. Uma outra alternativa é trabalhar com robôs cinemáticamente redundantes. Assim, se por exemplo, um mínimo de 6 graus de liberdade são necessários para uma determinada tarefa, trabalha-se com um robô com 8 graus de liberdade, o que

permite que, dependendo da junta onde ocorre a falha, o robô consiga executar a tarefa requerida mesmo com uma falha nos componentes de uma junta (neste caso, a falha deve ser detectada e isolada para que a junta seja travada). Em outra alternativa, como a usada no Sistema Manipulador Remoto do Ônibus Espacial, o sistema de DDF deve desligar o sistema de controle e o operador deve manualmente executar a tarefa [VISINSKY *et al.*, 1994].

Por outro lado surgem sistemas de tolerância a falhas que não necessitam de mecanismos adicionais. Dois exemplos serão aqui citados. No primeiro, considerando que uma falha que afete o atuador de uma junta seja detectada e isolada, reconfigura-se o sistema de controle para trabalhar com o manipulador subatuado. Em BERGERMAN (1996), os problemas de modelagem, controlabilidade, controle e planejamento de trajetórias em manipuladores subatuados foram estudados. Se uma falha que impossibilite o uso do atuador de uma junta qualquer do manipulador foi detectada e isolada, o sistema de controle pode ser reconfigurado para, através das juntas que ainda têm atuação, controlar a junta não atuada. Salienta-se que as juntas devem ser dotadas de freios. O segundo exemplo trabalha com falhas que não prejudiquem seriamente o manipulador, tal qual mudanças paramétricas (como aumento de massa nos elos). Utilizando o conceito de geração de resíduos, pode-se reconfigurar o sistema de controle para que tenha a habilidade de tolerância a falhas [VEMURI & POLYCARPOU, 1997]. Para isso, o vetor de resíduos é utilizado para estimar a função de falha, que indica a diferença do modelo esperado (sem falhas) e do sistema faltoso. Esta diferença deve ser usada para reconfigurar o sistema de controle que usa o modelo matemático do sistema. Tal qual para os sistemas de DDF, a precisão do mapeamento dinâmico do sistema (via RNA) é essencial para o bom funcionamento do mecanismo.

Capítulo 4

Sistema de detecção e diagnóstico de falhas via redes neurais artificiais para robôs manipuladores

A proposta do presente trabalho é utilizar um sistema de DDF via RNA baseado no conceito de geração de resíduos [KöPPEN-SELIGER & FRANK, 1996] para robôs manipuladores. Um MLP é utilizado para reproduzir o comportamento dinâmico do robô manipulador e uma rede RBF para a classificação do vetor de resíduos. Mas, por que utilizar o MLP para o mapeamento dinâmico se o modelo matemático do robô é tão conhecido? A principal motivação é que os robôs manipuladores reais estão sujeitos a fenômenos cuja modelagem é uma tarefa intrincada. Um bom exemplo é a fricção nas juntas. Já quando se utiliza o MLP, o conhecimento detalhado destes fenômenos não é necessário pois seus efeitos são abstraídos pelo mapeamento feito pela RNA. O uso da rede RBF para a classificação dos resíduos advém do fato de que, ao invés de conceber bandas de detecção fixas, fato que iria produzir inúmeros alarmes falsos já que os resíduos são dependentes da dinâmica do sistema, constrói-se bandas de detecção variáveis. Utilizando-se uma RNA para a análise dos resíduos, a banda de detecção é determinada de acordo com os padrões do conjunto de treinamento, não sendo necessário a utilização dos conhecimentos de um especialista. A seguir, as duas partes principais do sistema de DDF (geração dos resíduos e classificação dos resíduos) serão descritas. Na seção 4.3, o procedimento para treinamento e operação do sistema de DDF via RNA será descrito.

4.1. GERAÇÃO DOS RESÍDUOS

A geração dos resíduos (Capítulo 1) é feita através da comparação dos estados (posições e velocidades das juntas - Eq. 45) medidos no instante ($t+\Delta t$) com as respectivas estimações produzidas pelas saídas do MLP com treinamento por retropropagação (Figura 17). O MLP deve reproduzir o comportamento dinâmico do sistema sem falhas. Os estados **x** (posições e velocidades) e as entradas **u** (torques) do sistema medidos no instante t alimentam o MLP que deve gerar a estimação dos estados, $\hat{\mathbf{x}}$, no instante ($t+\Delta t$). O resíduo $\hat{\boldsymbol{\phi}}$ é gerado através da subtração $\mathbf{x}-\hat{\mathbf{x}}$. O MLP deve ser treinado para que o resíduo seja zero quando o robô manipulador não apresenta nenhuma falha. Quando ocorrer uma falha, este resíduo terá um valor diferente de zero, possibilitando a sua detecção. Espera-se que cada falha tenha uma assinatura particular, o que permite que a rede RBF possa realizar a classificação.

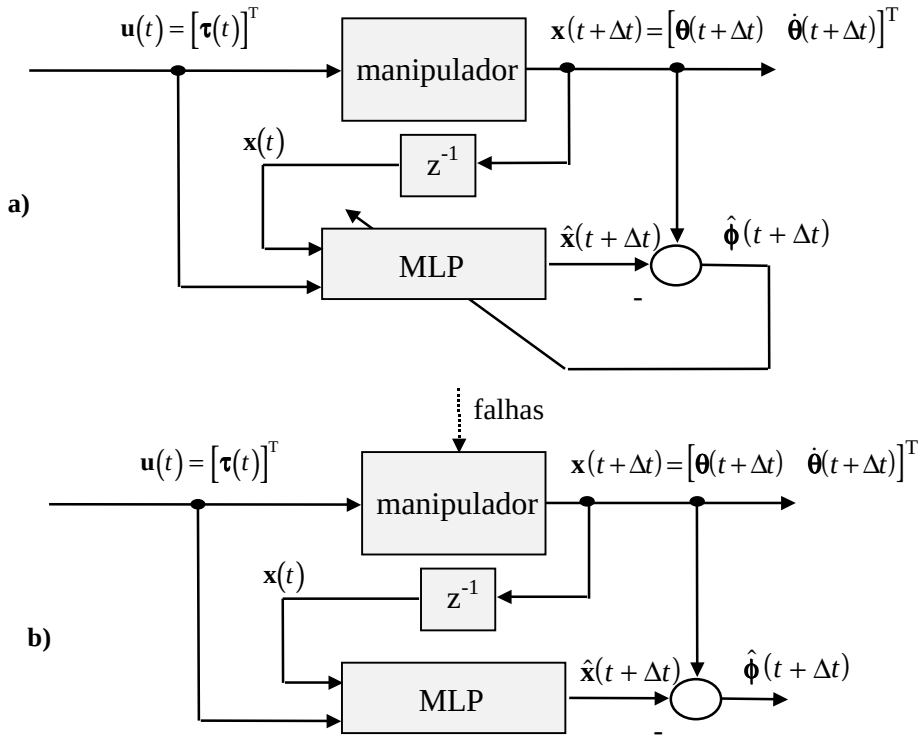

Figura 17. Geração dos resíduos: a) Treinamento do *perceptron* multicamadas; b) Operação.

Utilizando a arquitetura descrita acima, verificou-se que os resíduos produzidos pelas posições das juntas não tinham um alto poder discriminatório para as falhas consideradas. Isto ocorre porque, sendo baixo o período de amostragem, a diferença entre as posições no

instante t e no instante $t+\Delta t$ é pequena e, consequentemente, o resíduo produzido é baixo mesmo para o manipulador com falhas. Se existem ruídos de medida no sistema e erros no mapeamento realizado pelo MLP, a classificação feita pela rede RBF pode ser prejudicada pois estes resíduos poderão ser encobertos. Optou-se, portanto, por empregar somente os resíduos produzidos pelas velocidades das juntas na entrada da rede RBF. Esse procedimento também facilita o treinamento da rede RBF pois, diminuindo a dimensão do espaço de entradas, o número de padrões de treinamento necessário para uma boa generalização decresce.

A seguir será descrita a forma da função não-linear que o MLP deve reproduzir. Como os resíduos de posição não serão empregados, será considerada na saída do MLP somente as velocidades das juntas no instante $t+\Delta t$.

Para a utilização da equação dinâmica do manipulador (Eq. 45) como função a ser reproduzida, é preciso que os dados de aceleração nas juntas sejam mensurados. Entretanto, os dados de aceleração geralmente não são disponíveis nos robôs manipuladores. Para resolver tal problema, a aceleração será relacionada com a velocidade. Substituindo

$$\ddot{\boldsymbol{\theta}}(t) = \frac{\dot{\boldsymbol{\theta}}(t+\Delta t) - \dot{\boldsymbol{\theta}}(t)}{\Delta t}. \qquad (47)$$

na equação (45), tem-se (considerando apenas a segunda equação) que a dinâmica do manipulador sem falhas é dada por

$$\dot{\boldsymbol{\theta}}(t+\Delta t) = \mathbf{M}(\boldsymbol{\theta})^{-1}\left[\boldsymbol{\tau} - \mathbf{v}(\boldsymbol{\theta},\dot{\boldsymbol{\theta}}) - \mathbf{g}(\boldsymbol{\theta}) - \mathbf{z}(\boldsymbol{\theta},\dot{\boldsymbol{\theta}},t) - \mathbf{d}(t)\right]\Delta t + \dot{\boldsymbol{\theta}}(t). \qquad (48)$$

E, procedendo da mesma forma na equação (46), tem-se que a dinâmica do manipulador com falhas é dada por

$$\dot{\boldsymbol{\theta}}(t+\Delta t) = \mathbf{M}(\boldsymbol{\theta})^{-1}\left[\boldsymbol{\tau} - \mathbf{v}(\boldsymbol{\theta},\dot{\boldsymbol{\theta}}) - \mathbf{g}(\boldsymbol{\theta}) - \mathbf{z}(\boldsymbol{\theta},\dot{\boldsymbol{\theta}},t) - \mathbf{d}(t)\right]\Delta t + \boldsymbol{\phi}(\boldsymbol{\theta},\dot{\boldsymbol{\theta}},\boldsymbol{\tau},t)\Delta t + \dot{\boldsymbol{\theta}}(t). \qquad (49)$$

O MLP deve reproduzir a equação dinâmica do manipulador sem falhas (Eq. 48). Assim, os dados de entrada do MLP serão os estados (posições e velocidades das juntas) e as entradas (torques nas juntas) medidas no instante t. As saídas do MLP serão as estimações das velocidades medidas no instante ($t+\Delta t$). Deste modo

$$\boldsymbol{\xi}(n) = \begin{bmatrix} \boldsymbol{\theta}(t) \\ \dot{\boldsymbol{\theta}}(t) \\ \boldsymbol{\tau}(t) \end{bmatrix} \text{ e,}$$

$$\hat{\boldsymbol{\psi}}(n) = \left[\hat{\dot{\boldsymbol{\theta}}}(t + \Delta t)\right].$$

É importante observar que o período amostral está implícito no mapeamento realizado pelo MLP. O erro de mapeamento do MLP (manipulador sem falhas) no padrão *n* é dado por

$$\boldsymbol{\psi}(n) - \hat{\boldsymbol{\psi}}(n) = \dot{\boldsymbol{\theta}}(t + \Delta t) - \hat{\dot{\boldsymbol{\theta}}}(t + \Delta t) = \mathbf{e}(t + \Delta t). \qquad (50)$$

Note que os ruídos de medida podem estar implícitos nesta equação. Assim, o vetor de resíduo do manipulador livre de falhas deve ser dado por

$$\hat{\boldsymbol{\phi}}(t + \Delta t) = \mathbf{e}(t + \Delta t). \qquad (51)$$

Portanto, para o manipulador sem falhas, o vetor de resíduos varia exclusivamente devido aos erros de mapeamento do MLP e aos ruídos de medida. Considerando, agora, os efeitos das falhas no manipulador, o vetor de resíduos deve ser dado pela diferença das Eq. (49) e (48) acrescida do erro de mapeamento

$$\hat{\boldsymbol{\phi}}(t + \Delta t) = \boldsymbol{\phi}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \boldsymbol{\tau}, t)\Delta t + \mathbf{e}(t + \Delta t). \qquad (52)$$

Desta forma, se o erro de mapeamento do MLP e os ruídos de medida forem significativos, o vetor de falhas poderá ser encoberto, ocasionando erros na DDF. Como os erros de mapeamento e os ruídos de medida são inevitáveis em sistemas reais, utiliza-se a rede RBF para construir bandas de detecção variáveis.

4.2. ANÁLISE DOS RESÍDUOS

A análise é feita através da rede RBF que tem por objetivo classificar os padrões de resíduo. Os padrões de entrada desta rede serão o vetor de resíduos e as velocidades das juntas. As velocidades são utilizadas pois nos manipuladores, os erros de mapeamento do

MLP e os ruídos de medida variam dinamicamente. Este é o caso, por exemplo, de um mapeamento incorreto dos efeitos dos termos de fricção. Utilizando, portanto, os dados de velocidade das juntas, a rede RBF constrói bandas de detecção que variam dinamicamente. Por exemplo, se os erros de mapeamento são grandes em velocidades pequenas, a rede RBF conseguirá abstrair essa característica e, assim, alarmes falsos não serão disparados. O efeito negativo de se acrescentar os dados de velocidade das juntas é que durante o treinamento, a rede RBF necessitará de mais padrões para que consiga uma boa generalização, pois a dimensão do espaço de entradas aumenta.

A arquitetura do sistema de análises dos resíduos pode ser vista na Figura 18. A saída da rede RBF apresenta um vetor *q*-dimensional, na qual *q* é o número de falhas considerado. Cada unidade deste vetor é responsável pela detecção de uma falha diferente. A rede RBF deve separar o espaço 2 x n_g dimensional de entradas (n_g resíduos e n_g velocidades) em regiões que devem ser associadas à operação normal do robô manipulador e às diferentes falhas.

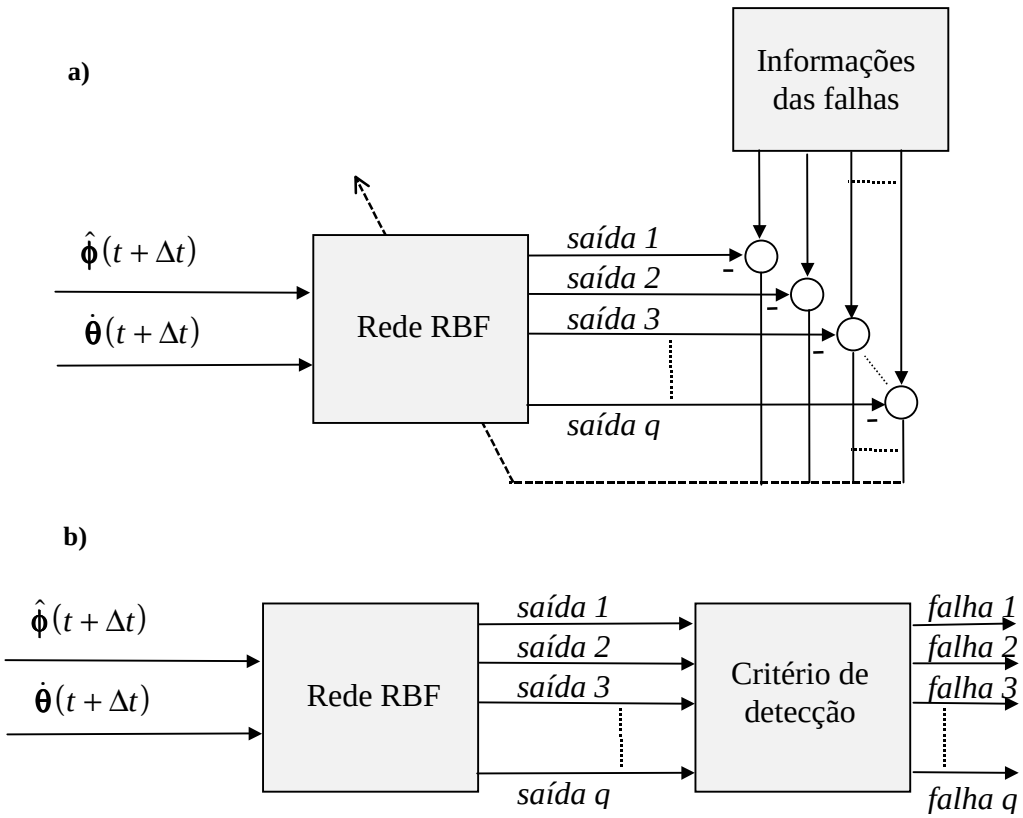

Figura 18. Análise dos resíduos: a) Treinamento da rede RBF; b) Operação.

A rede RBF é treinada para que suas saídas apresentem sinal 1 no caso de uma falha do seu tipo relacionado e 0 em caso contrário (por exemplo, se ocorre uma falha do tipo 2, o

neurônio de saída 2 deve ter ativação igual a 1 e os outros neurônios devem ter ativação igual a 0). O critério de detecção considera que para uma falha ser detectada, o neurônio de saída correspondente deve ter uma sequência (com um número pré-definido de amostras) de ativações maiores que 0,5. Este critério é adotado para evitar que padrões isolados mal-classificados causem alarmes falsos. Assim, se é considerado que o número de amostras para a detecção deve ser 3, então a falha é detectada quando um neurônio de saída apresentar 3 ativações consecutivas maiores que 0,5. Este critério não diminui significativamente a sensibilidade do sistema de DDF pois, em geral, o período amostral é baixo.

4.3. PROCEDIMENTOS DO SISTEMA DE DETECÇÃO E DIAGNÓSTICO DE FALHAS

A seguir, os procedimentos para treinamento e operação do sistema de DDF serão descritos.

4.3.1. Procedimento para treinamento

1. Gera-se um conjunto significativo de trajetórias livres de falhas e, em cada amostra, mede-se as posições, as velocidades angulares e os torques nas juntas;
2. as amostras deste conjunto de treinamento são normalizadas e, posteriormente, empregadas para treinar o MLP. Na entrada do MLP, apresenta-se as posições, as velocidades e os torques das juntas em cada instante amostral. O vetor de saídas do MLP é comparado com as velocidades das juntas do instante amostral seguinte. A diferença entre estes dois vetores é utilizada para ajustar os pesos do MLP. Este procedimento deve ser repetido até que um critério de paradas (como o número máximo de épocas de treinamento) seja alcançado;
3. avalia-se o mapeamento do MLP apresentando-se os dados normalizados de trajetórias não-treinadas. Se a análise é negativa, repete-se os passos anteriores mudando-se o conjunto de treinamento (provavelmente o conjunto não é significativo o bastante) e/ou os parâmetros de treinamento. Se a análise é positiva, passe para o passo seguinte;
4. gera-se um conjunto significativo de trajetórias. Os dados de cada trajetória são normalizados e devem ser apresentados q+1 vezes (q é o número de falhas): em cada apresentação, um tipo de falha diferente deve ocorrer e, na última, nenhuma falha ocorre.

Em cada amostra, as posições, as velocidades e os torques das juntas são apresentadas ao MLP que gera, assim, o vetor de resíduos;

5. as amostras dos vetores de resíduos gerados e das velocidades normalizadas são apresentadas à rede RBF. Um dos quatro algoritmos descritos na seção 2.2 é utilizado para o treinamento da rede RBF. O vetor de saídas da rede RBF, após ser analisado pelo critério de falhas, é comparado com as informações reais das falhas e,
6. avalia-se a classificação realizada pela rede RBF apresentando-se dados de trajetórias não-treinadas em que ocorrem cada tipo de falha e para operação normal. Analisa-se a ocorrência de alarmes falsos e se as falhas são detectadas e isoladas corretamente. Se a análise é negativa, repete-se os passos 4, 5 e 6 mudando-se o conjunto de treinamento (provavelmente o conjunto não é significativo o bastante) e/ou os parâmetros de treinamento. Se a análise é positiva, finaliza-se o treinamento.

4.3.2. Procedimento para operação

1. Medem-se as variáveis de posições e velocidades angulares das juntas no instante $t+\Delta t$ através de sensores;
2. os sinais medidos e os torques (fornecidos pelo sistema de controle) no instante $t+\Delta t$ sofrem pré-prossessamento (conversões, normalizações, etc.);
3. os valores das velocidades, posições e torques das juntas no instante t (já pré-processados) são apresentados à entrada do MLP, que tem como saída a estimativa das velocidades das juntas no instante $t+\Delta t$;
4. a saída do MLP é comparada com as velocidades das juntas pré-processadas medidas no instante $t+\Delta t$, gerando o vetor de resíduos;
5. o vetor de resíduos e as velocidades das juntas pré-processadas medidas no instante $t+\Delta t$ são apresentadas à entrada da rede RBF;
6. a rede RBF classifica os padrões de entrada e gera um vetor de saídas que, após ser analisado pelo critério de falhas, indica o estado de operação do sistema (sem falhas ou ocorrência de determinada falha) e,
7. incrementa-se o tempo t e retorna-se para o passo 1.

Capítulo 5

Resultados

Dois manipuladores foram simulados em MATLAB para testes do sistema de DDF: um manipulador planar com 2 graus de liberdade e um manipulador Puma 560. Os algoritmos FS, GRR e LRR também foram executados em MATLAB, pois fez-se uso de rotinas do *toolbox* RBF desenvolvido por ORR (1996 b). A descrição dos programas construídos encontra-se no Apêndice A8.

5.1. MANIPULADOR PLANAR COM 2 GRAUS DE LIBERDADE

Para o estudo do sistema de DDF foi simulado através do programa MATLAB, um robô manipulador planar com dois elos rígidos e juntas rotativas adaptado de CRAIG (1988). O robô, que pode ser visto na Figura 16, utiliza o método do torque calculado para o controle de posicionamento das juntas. No entanto, outros métodos de controle podem ser utilizados sem que se precise fazer mudanças no sistema de DDF pois o MLP mapeia as equações do manipulador e não as do controlador. Os parâmetros do manipulador e do controlador podem ser vistos na Tabela 1.

Tabela 1. Parâmetros do manipulador com 2 graus de liberdade e do sistema de controle.

Parâmetro	Valor	Descrição
l_1	0,3 m	tamanho do elo 1
l_2	0,3 m	tamanho do elo 2
m_1	5 kg	massa no elo 1
m_2	5 kg	massa no elo 2
g	9,81 m/s^2	aceleração da gravidade
τ_{1max}	100 N m	torque máximo aplicado no elo 1
τ_{1min}	-100 N m	torque mínimo aplicado no elo 1
τ_{2max}	50 N m	torque máximo aplicado no elo 2
τ_{2min}	-50 N m	torque mínimo aplicado no elo 2
h	0,015 s	período amostral
$\mathbf{K}_p$	400 rad $^{-1}$	ganho proporcional do controlador
$\mathbf{K}_v$	40 s rad $^{-1}$	ganho derivativo do controlador

É considerado que cada junta do robô possui dois sensores: um para velocidade angular e outro para posição. Durante a simulação, ruídos de medida são considerados nos sinais de posição das juntas (média=0 e variância=0.005) e de velocidade angular das juntas (média=0 e variância=0.05). O robô executa suas tarefas através de trajetórias pré-definidas. Em cada trajetória, os elos saem de uma posição inicial e realizam o movimento até a posição final. Por exemplo, a trajetória vista na Figura 16 vai de $\boldsymbol{\theta}=[\pi/4\ 0]^T$ até $\boldsymbol{\theta}=[\pi/2\ \pi/4]^T$.

5.1.1. Geração dos Resíduos

Para o treinamento do MLP, utilizam-se 10 trajetórias com 50 amostras cada em que não se considera nenhuma falha no robô. Portanto, um total de 500 padrões são usados no treinamento do MLP. As trajetórias utilizadas no treinamento são escolhidas para que a maior parte do espaço de posições das juntas seja coberto. Isto deve acontecer para que os padrões utilizados nos treinamento sejam bastante significativos, permitindo a generalização do mapeamento para padrões não-vistos. Durante o treinamento, o robô manipulador é simulado para todas as trajetórias e os dados de posição, velocidade angular e torque das juntas são normalizados e em seguida apresentados ao MLP. A normalização é feita entre 0

e 1 devido a função de ativação utilizada (sigmoidal). A normalização é feita do seguinte modo: para cada variável do vetor de entradas, relacionam-se o maior e o menor valores do conjunto de treinamento respectivamente a 0 e a 1. Os parâmetros do treinamento por retropropagação do erro para o MLP podem ser vistos na Tabela 2. Os vetor de entradas é formado pelas posições, velocidades angulares e torques das 2 juntas em cada instante amostral t e o vetor de saídas é formado pelas velocidades angulares das 2 juntas no instante amostral $t+\Delta t$. As trajetórias utilizadas no treinamento do MLP podem ser vistas na Apêndice A9.

Tabela 2. Parâmetros do treinamento por retropropagação do erro no MLP para o manipulador com 2 graus de liberdade.

Número de unidades na primeira camada	6
Número de unidades na camada escondida	13
Número de unidades na última camada	2
Função de ativação utilizada	sigmoidal
Taxa de aprendizagem	0,4
Momentum	0,1

Como se deseja um resíduo baixo quando não existem falhas no manipulador, um grande número de épocas de treinamento é necessário. Neste trabalho, um total de 18.000 épocas de treinamento foram utilizadas. O erro médio quadrático do MLP depois deste período foi de $5{,}208 \times 10^{-5}$ para o conjunto de treinamento. Como teste de validação, apresentam-se trajetórias não-treinadas ao MLP. Neste trabalho, o MLP reproduz a dinâmica do sistema em operação normal com um resíduo baixo mesmo para trajetórias não-treinadas, comprovando a generalização do procedimento. Em determinados instantes de certas trajetórias, o resíduo se torna maior (geralmente quando a velocidade angular é muito grande). Tal fato se explica por um erro no mapeamento realizado pelo MLP. Este erro pode ser evitado aumentando-se o número de padrões de treinamento e/ou o número de épocas. No entanto, tal comportamento não prejudica a resposta do sistema de DDF pois a rede RBF consegue abstraí-lo (ou seja, graças aos dados de velocidade das juntas, a rede RBF consegue construir bandas de detecção dinâmicas). A Figura 19 mostra os sinais de posição, velocidade e torque das juntas que deverão ser normalizados para serem apresentados ao MLP. Estes sinais foram obtidos através da simulação de uma trajetória não-treinada livre de falhas. A Figura 20 traz as velocidades reais das juntas e as respectivas saídas não-

normalizadas do MLP para os mesmos padrões da figura anterior. Os resíduos (das velocidades das juntas) resultantes podem ser vistos na Figura 21. Repare que quando a velocidade das juntas é alta (em módulo), o resíduo é alto. Observa-se também uma oscilação de baixa amplitude presente no sinal de resíduo que é provocada pelos ruídos de medida.

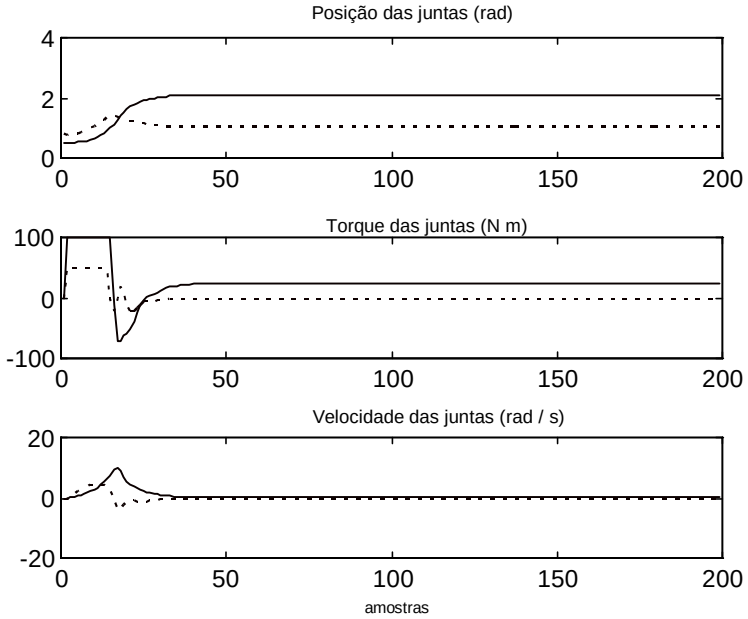

Figura 19. Entradas do MLP (antes da normalização) para a trajetória não-treinada livre de falhas de $\theta=[\pi/6\ \pi/4]^T$ até $\theta=[2\pi/3\ \pi/3]^T$. Os dados da junta 1 são representados pelas linhas contínuas e os dados da junta 2, pelas linhas tracejadas.

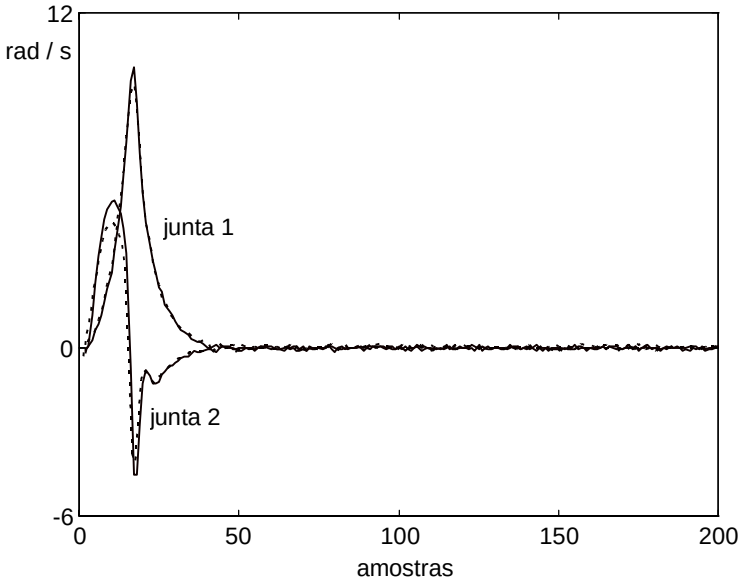

Figura 20. Velocidades reais (linhas contínuas) e saídas do MLP sem normalização (linhas tracejadas) para a trajetória não-treinada livre de falhas de $\theta=[\pi/6\ \pi/4]^T$ até $\theta=[2\pi/3\ \pi/3]^T$.

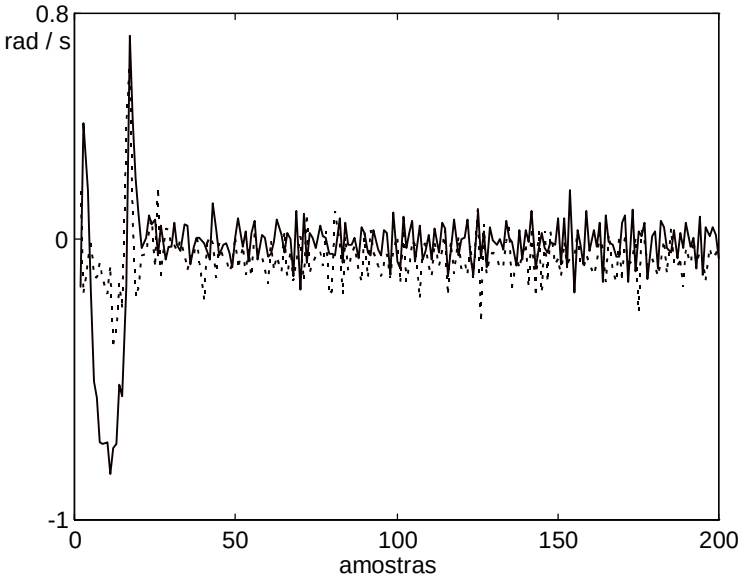

Figura 21. Resíduos das velocidades (não-normalizados) para a trajetória não-treinada livre de falhas de $\theta=[\pi/6\ \pi/4]^T$ até $\theta=[2\pi/3\ \pi/3]^T$. Os dados da junta 1 são representados pelas linhas contínuas e os dados da junta 2, pelas linhas tracejadas.

Dois tipos de falhas são considerados na simulação do sistema de DDF para o manipulador com 2 graus de liberdade. Na primeira (falha 1) a junta do elo 1 é bloqueada em um determinado instante. Na segunda (falha 2) o mesmo ocorre para a junta do elo 2. Estas falhas podem ter como causa, por exemplo, um travamento do motor de acionamento ou um travamento das engrenagens que ligam a junta ao motor. As Figuras 22, 23 e 24 mostram as entradas e as saídas do MLP e os sinais de resíduo resultantes para a mesma

trajetória empregada nas figuras anteriores, mas com a ocorrência da falha do tipo 1 entre as amostras 10 e 160.

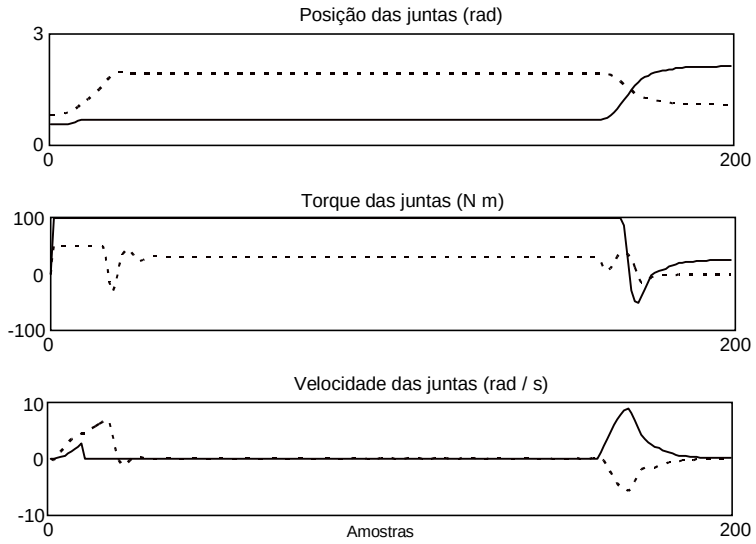

Figura 22. Entradas do MLP (antes da normalização) para a trajetória não-treinada de $\theta=[\pi/6\ \pi/4]^T$ até $\theta=[2\pi/3\ \pi/3]^T$ em que ocorre a falha 1 entre as amostras 10 e 160. Os dados da junta 1 são representados pelas linhas contínuas e os dados da junta 2, pelas linhas tracejadas.

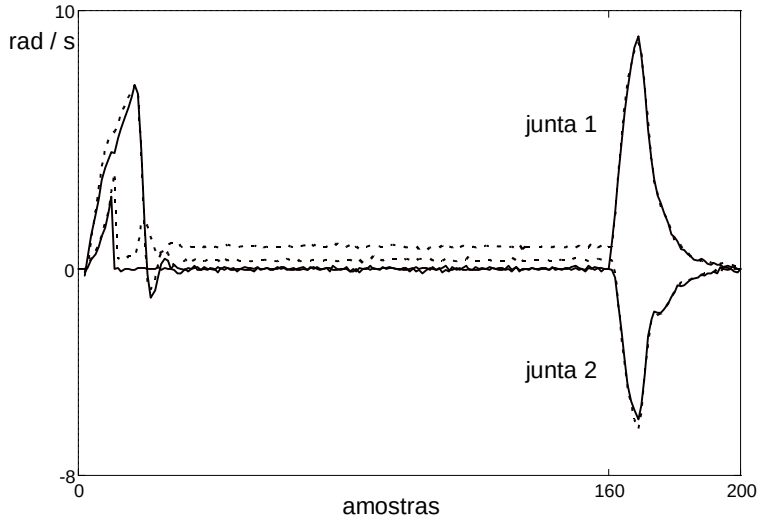

Figura 23. Velocidades reais (linhas contínuas) e saídas do MLP não-normalizadas (linhas tracejadas) para a trajetória não-treinada de $\theta=[\pi/6\ \pi/4]^T$ até $\theta=[2\pi/3\ \pi/3]^T$ em que ocorre a falha 1 entre as amostras 10 e 160.

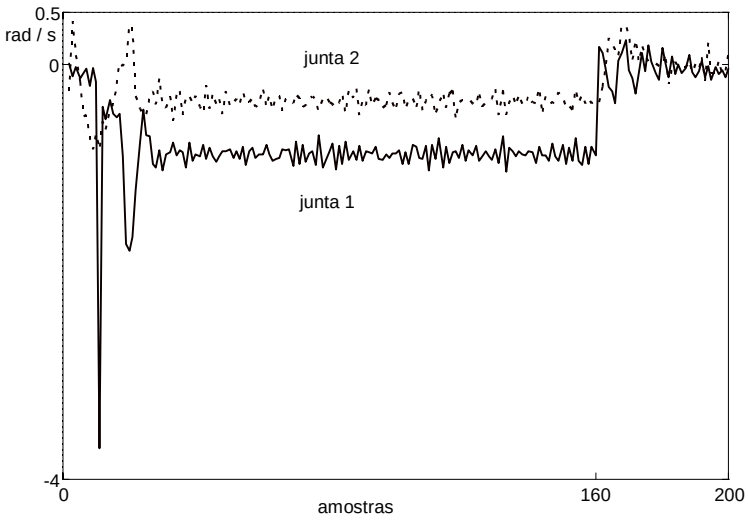

Figura 24. Resíduos das velocidades (não-normalizados) para a trajetória não-treinada de $\theta=[\pi/6\ \pi/4]^T$ até $\theta=[2\pi/3\ \pi/3]^T$ em que ocorre a falha 1 entre as amostras 10 e 160. Os dados da junta 1 são representados pelas linhas contínuas e os dados da junta 2, pelas linhas tracejadas.

5.1.2. Análise dos Resíduos

A rede RBF para o manipulador com dois graus de liberdade tem 4 entradas (resíduos e velocidades) e duas saídas (2 falhas). Para o treinamento da rede RBF, 9 trajetórias com 40 amostras cada são utilizadas. Estas trajetórias são apresentadas à rede RBF três vezes: uma em operação normal, uma em que ocorre a falha 1 e uma em que ocorre a falha 2, perfazendo um total de 1080 padrões. A definição dos padrões mais significativos de treinamento e dos parâmetros que definem o tamanho do campo receptivo (diagonal da matriz **R**) são essenciais para a rede RBF. Os padrões de treinamento devem ocupar regiões do espaço de entradas significativas para que os novos padrões possam ser associados através dos campos receptivos das unidades radiais. As trajetórias utilizadas para o treinamento da rede RBF podem ser vistas no Apêndice A9. Neste trabalho, o tamanho do campo receptivo, definido pela diagonal de **R**, é dado por $[0.05\ 0.035\ 0.1\ 0.1]^T$, no qual os dois primeiros valores atuam sobre os resíduos e os dois últimos atuam sobre as velocidades.

Para comparação de resultados, os quatro métodos descritos na seção 2.2 foram utilizados. O método FS utilizou o OLS para otimizar o algoritmo e o GCV como critério de parada. Durante o treinamento, foram selecionadas 415 unidades radiais. A Figura 25 mostra o treinamento da rede RBF pelo método FS. Note que o erro GCV diminui conforme aumenta o número de centros escolhidos. O algoritmo termina a seleção quando o erro GCV

cresce durante duas iterações seguidas. Nota-se que o algoritmo pára a seleção quando o primeiro mínimo (local ou global) é alcançado.

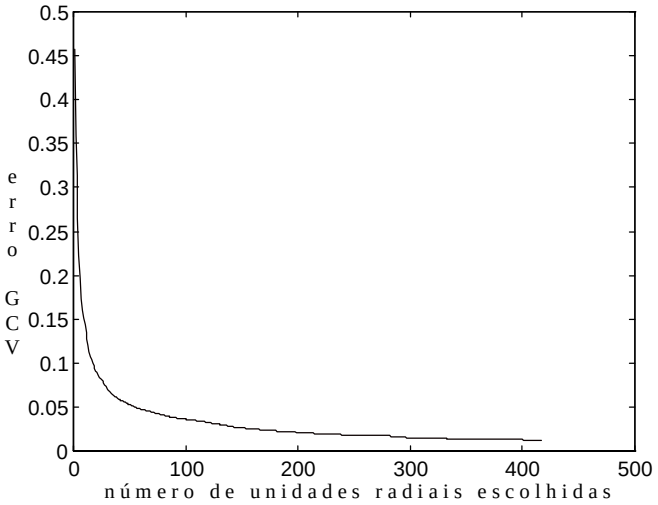

Figura 25. Convergência do método FS durante o treinamento da rede RBF.

Para o método GRR, utiliza-se como critério de convergência uma mudança relativa de 1/1000, ou seja, o método deve parar a seleção do parâmetro de regularização global ($\hat{\lambda}$) quando a mudança no valor do erro GCV entre duas iterações for menor que 1/1000. O valor selecionado foi $\hat{\lambda}$ = 7,642 x 10^{-6}. A Figura 26 mostra a convergência para este valor. O algoritmo começa com um valor pré-definido ($\hat{\lambda}$ = 1 x 10^{-4}) e reestima o parâmetro de regularização global até que o critério de convergência seja alcançado.

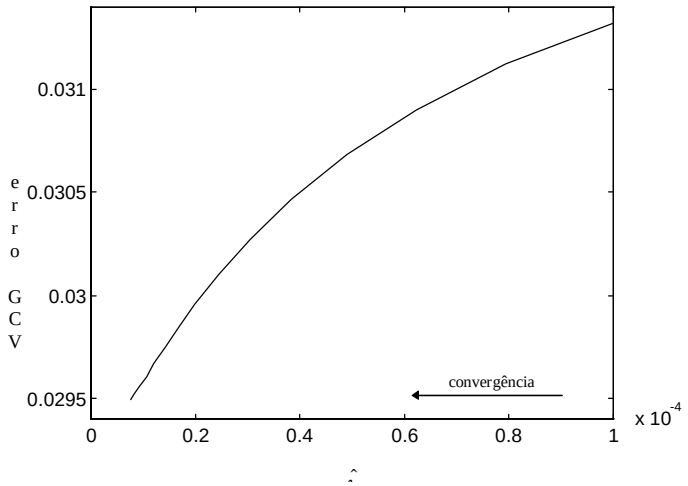

Figura 26. Convergência do método GRR durante o treinamento da rede RBF.

Para o LRR, o critério de convergência utilizado foi o mesmo do GRR. Foram "podadas" 761 unidades radiais da rede (estas unidades tiveram $\hat{\lambda}_j = \infty$). Assim, a rede utiliza somente 319 unidades radiais. Cada uma destas unidades radiais tem um parâmetro de regularização individual.

No método MAOK, os parâmetros utilizados foram: $\alpha=t^{-1}$, σ=0,015 t^{-1} e t_{max}=1000 (número de vezes que cada padrão é apresentado). Usando tais parâmetros, 215 unidades radiais foram selecionadas. A Tabela 3 mostra o erro médio quadrático das duas saídas da rede RBF (falha 1 e falha 2) para o conjunto de treinamento.

Tabela 3. Erro médio quadrático da rede RBF para o conjunto de treinamento.

	saída 1 - rede RBF (Falha 1)	saída 2 - rede RBF (Falha 2)
Forward Selection	0.0008	0.0042
Global Ridge Regression	0.0015	0.0066
Local Ridge Regression	0.0013	0.0058
Mapa Auto-organizável de Kohonen	0.0100	0.0212

Para teste de validação dos sistemas de DDF resultantes, 17 trajetórias não-treinadas com 100 amostras cada são apresentadas três vezes aos sistemas (uma para operação normal e uma para cada uma das duas falhas). Assim, o total de trajetórias usadas durante o teste é de 51. As trajetórias podem ser vistas no Apêndice A9. A Tabela 4 mostra o erro médio quadrático das duas saídas da rede RBF para o conjunto de teste.

Tabela 4. Erro médio quadrático da rede RBF para o conjunto de teste.

	saída 1 - rede RBF (Falha 1)	saída 2 - rede RBF (Falha 2)
Forward Selection	0.0699	0.2727
Global Ridge Regression	0.0354	0.0829
Local Ridge Regression	0.0414	0.1233
Mapa auto-organizável de Kohonen	0.0273	0.0788

As Figuras 27, 28, 29 e 30 mostram as saídas das rede RBF treinadas pelos quatro métodos para uma trajetória não-treinada em que ocorre a falha 1 entre as amostras 10 e 160. A saída 1 é responsável pela detecção da falha 1 e a saída 2 é responsável pela detecção da falha 2.

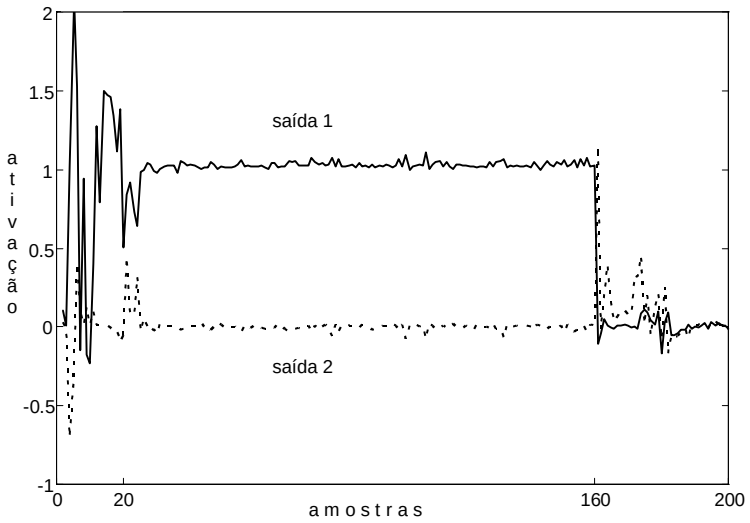

Figura 27. Saídas da rede RBF treinada por FS para a trajetória de $\theta=[\pi/6\ \pi/4]^T$ até $\theta=[2\pi/3\ \pi/3]^T$ em que ocorre a falha 1 entre as amostras 10 e 160.

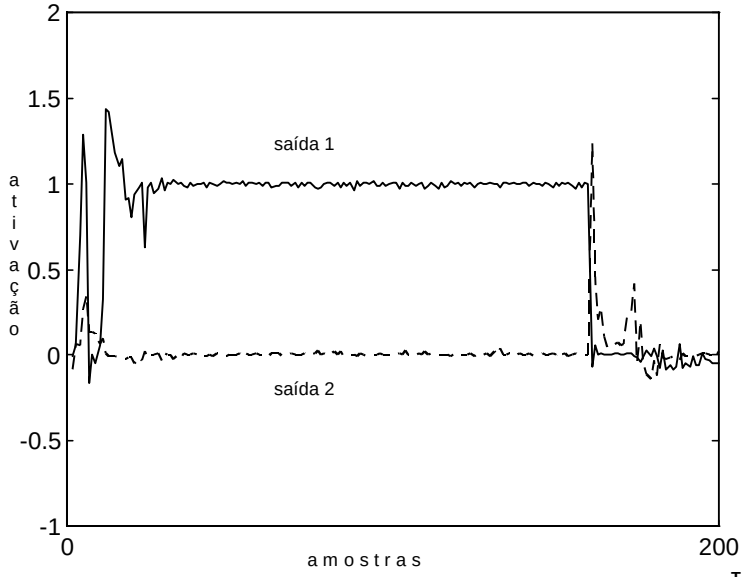

Figura 28. Saídas da rede RBF treinada por GRR para a trajetória de $\theta=[\pi/6\ \pi/4]^T$ até $\theta=[2\pi/3\ \pi/3]^T$ em que ocorre a falha 1 entre as amostras 10 e 160.

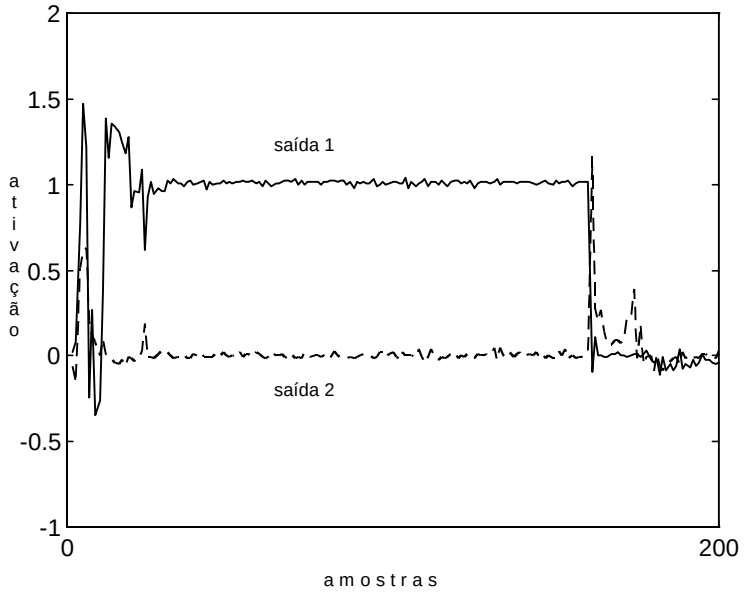

Figura 29. Saídas da rede RBF treinada por LRR para a trajetória de $\theta=[\pi/6\ \pi/4]^T$ até $\theta=[2\pi/3\ \pi/3]^T$ em que ocorre a falha 1 entre as amostras 10 e 160.

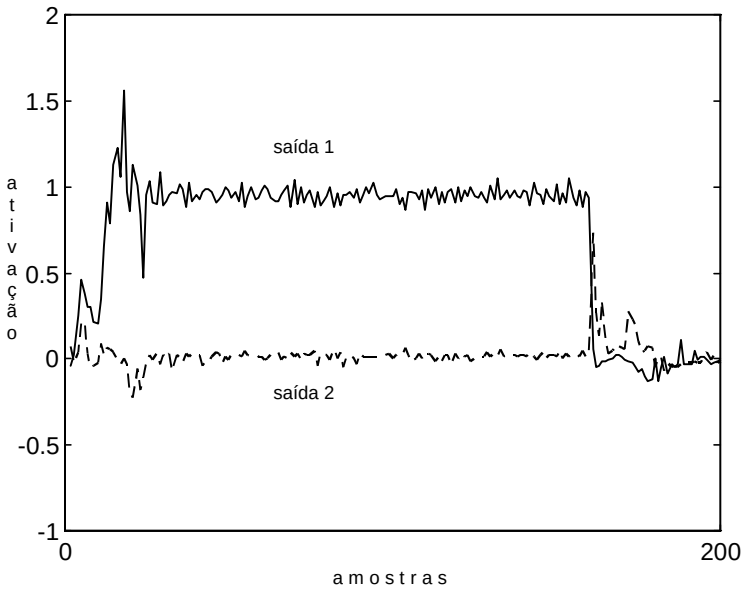

Figura 30. Saídas da rede RBF treinada por MAOK para a trajetória de $\theta=[\pi/6\ \pi/4]^T$ até $\theta=[2\pi/3\ \pi/3]^T$ em que ocorre a falha 1 entre as amostras 10 e 160.

O critério de falhas adotado é o seguinte: para ser caracterizada uma falha, a saída da rede RBF tem que apresentar cinco sinais seguidos maiores que 0,5. Isto não causa uma perda de sensibilidade significativa na detecção das falhas porque o período amostral é

baixo. A Figura 31 mostra a detecção e a isolação da falha 1 para as saídas da rede RBF vistas na Figura 28.

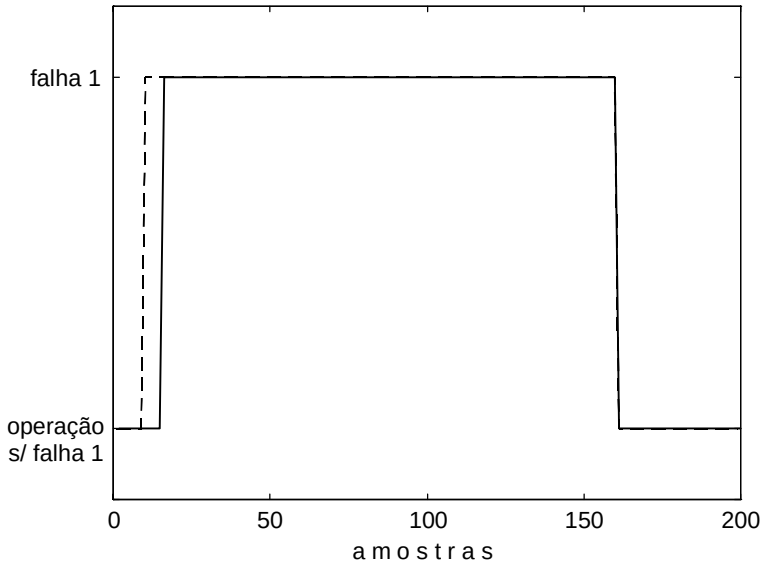

Figura 31. Detecção da falha 1 para as saídas da rede RBF treinada por GRR vistas na Figura 28. A falha 1 real esta representada pela linha tracejada.

A Figura 32 mostra os resíduos gerados em outra trajetória, a de $\theta=[2\pi/3\ \pi/2]^T$ até $\theta=[5\pi/12\ 0]^T$ em que ocorre a falha 2 entre as amostras 5 e 85. Note que agora o resíduo não é mostrado na figura não-normalizado. As saídas da rede RBF, treinada pelo método FS, e a detecção (após o critério de falhas ser aplicado) utilizando tais saídas podem ser vistas respectivamente nas Figuras 33 e 34. As Figuras 35 e 36 mostram tais sinais para a rede treinada pelo método GRR, as Figuras 37 e 38 para a rede treinada pelo método LRR e, as Figuras 39 e 40 para a rede treinada pelo método MAOK.

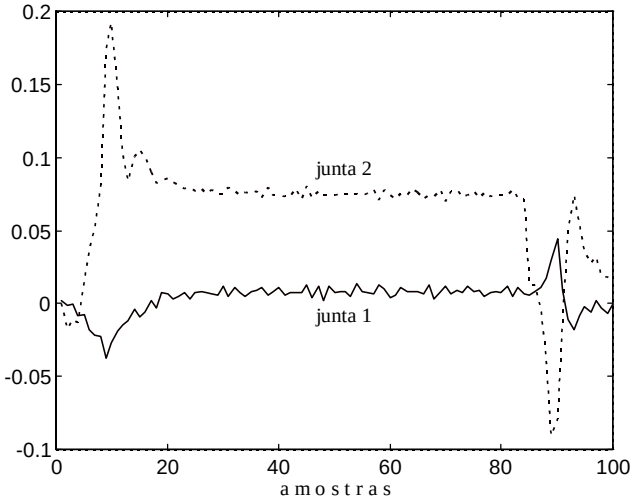

Figura 32. Resíduos para a trajetória $\theta=[2\pi/3\ \pi/2]^T$ até $\theta=[5\pi/12\ 0]^T$, com a falha 2 ocorrendo entre os amostras 5 e 85. Note que os resíduos não são apresentados não-normalizados.

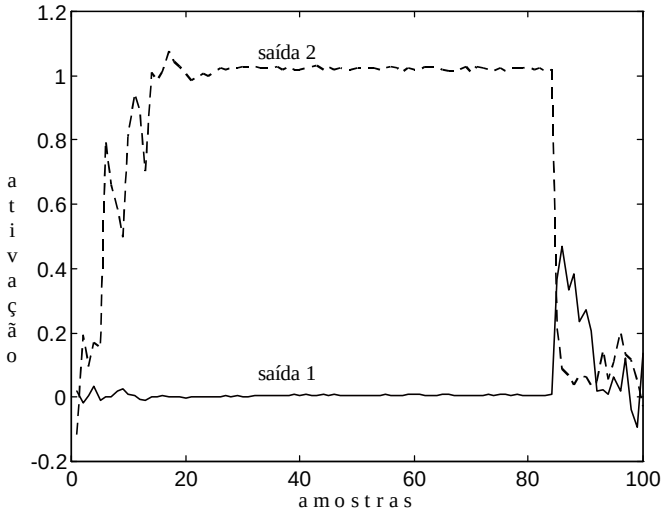

Figura 33. Saída da rede RBF treinada pelo método FS para a trajetória $\theta=[2\pi/3\ \pi/2]^T$ até $\theta=[5\pi/12\ 0]^T$, com a falha 2 ocorrendo entre os amostras 5 e 85.

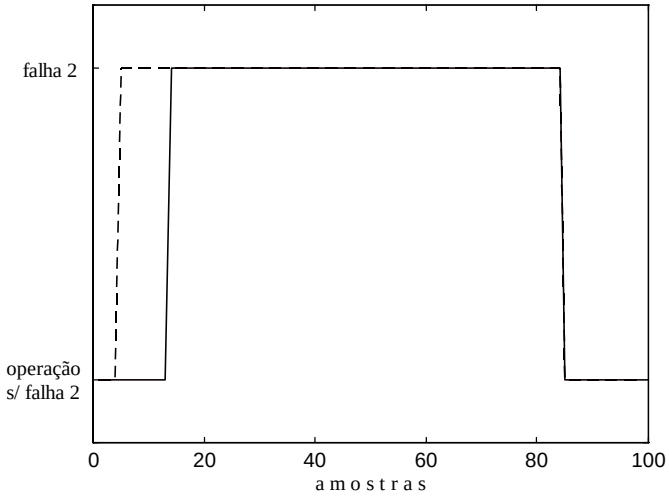

Figura 34. Detecção da falha 2 utilizando a rede RBF treinada pelo método FS (linha contínua) para a trajetória $\theta=[2\pi/3\ \pi/2]^T$ até $\theta=[5\pi/12\ 0]^T$. A falha 2 real está representada pela linha tracejada.

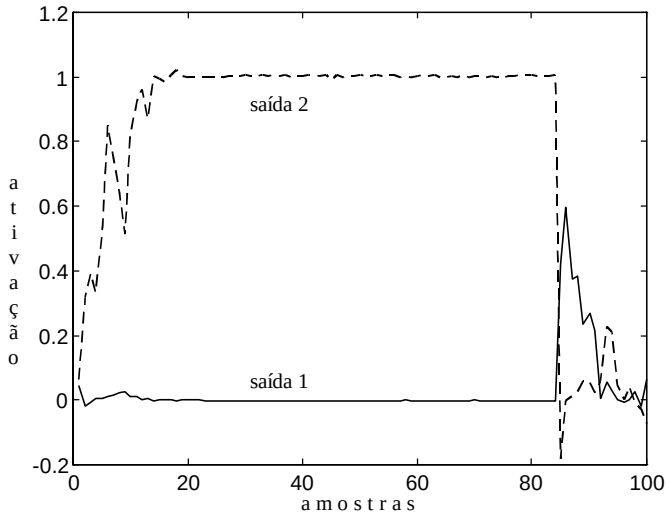

Figura 35. Saída da rede RBF treinada pelo método GRR para a trajetória $\theta=[2\pi/3\ \pi/2]^T$ até $\theta=[5\pi/12\ 0]^T$, com a falha 2 ocorrendo entre os amostras 5 e 85.

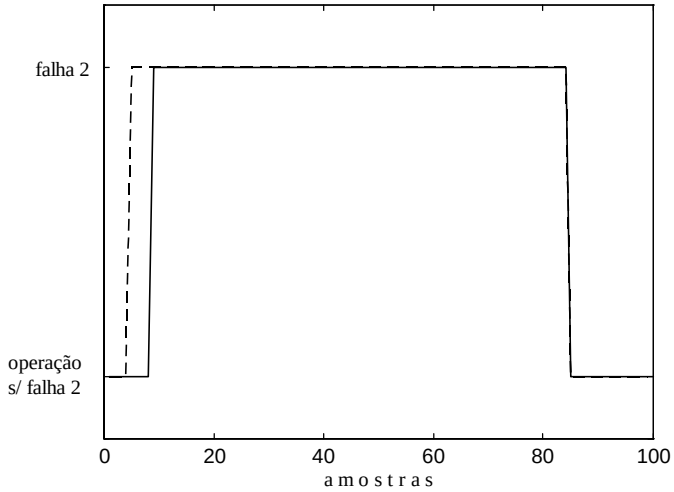

Figura 36. Detecção da falha 2 utilizando a rede RBF treinada pelo método GRR (linha contínua) para a trajetória $\theta=[2\pi/3\ \pi/2]^T$ até $\theta=[5\pi/12\ 0]^T$. A Falha 2 real está representada pela linha tracejada.

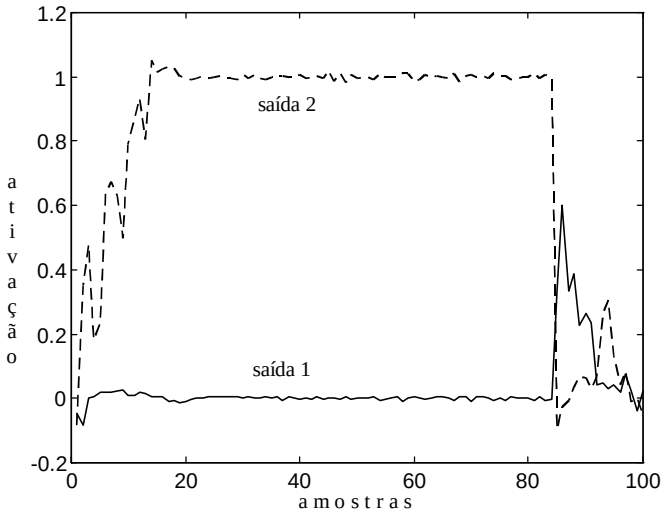

Figura 37. Saída da rede RBF treinada pelo método LRR para a trajetória $\theta=[2\pi/3\ \pi/2]^T$ até $\theta=[5\pi/12\ 0]^T$, com a falha 2 ocorrendo entre os amostras 5 e 85.

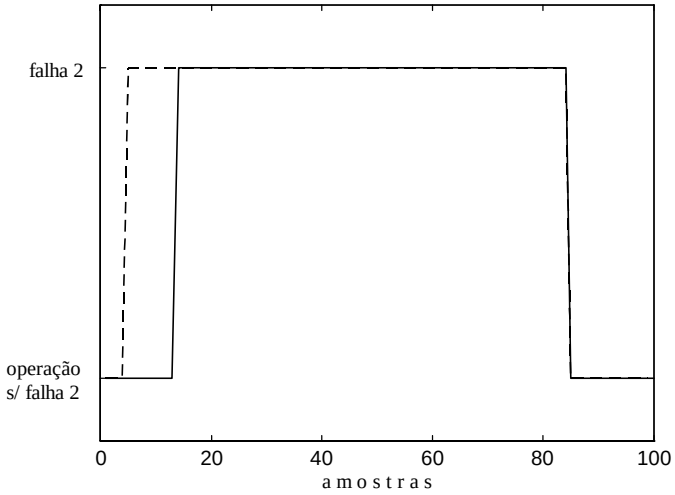

Figura 38. Detecção da falha 2 utilizando a rede RBF treinada pelo método LRR (linha contínua) para a trajetória $\theta=[2\pi/3\ \pi/2]^T$ até $\theta=[5\pi/12\ 0]^T$. A Falha 2 real está representada pela linha tracejada.

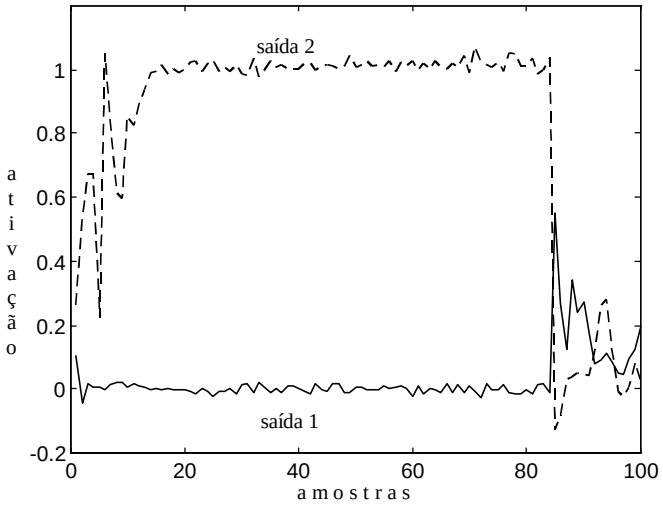

Figura 39. Saída da rede RBF treinada pelo método MAOK para a trajetória $\theta=[2\pi/3\ \pi/2]^T$ até $\theta=[5\pi/12\ 0]^T$, com a falha 2 ocorrendo entre os amostras 5 e 85.

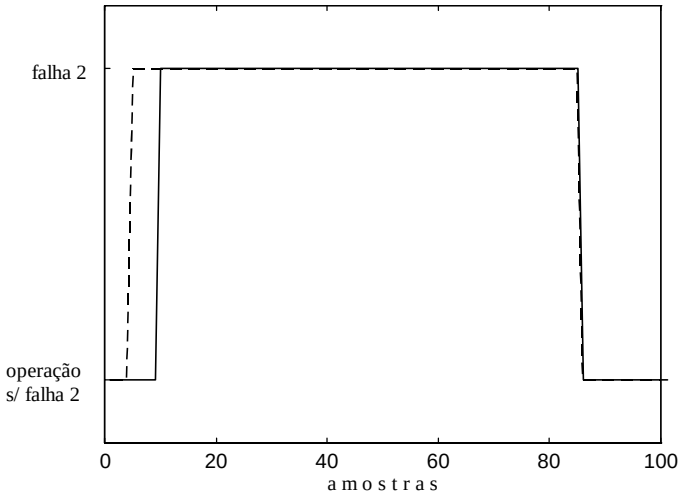

Figura 40. Detecção da falha 2 utilizando a rede RBF treinada pelo método MAOK (linha contínua) para a trajetória $\theta=[2\pi/3\ \pi/2]^T$ até $\theta=[5\pi/12\ 0]^T$. A Falha 2 real está representada pela linha tracejada.

Nos testes realizados com as 51 trajetórias formadas pelas 17 trajetórias do Apêndice A9, os sistemas de DDF com as redes RBF treinadas pelos métodos GRR, LRR e MAOK conseguiram detectar e isolar todas as falhas. Nenhum alarme falso foi detectado. Para o sistema de DDF com a rede RBF treinada pelos método FS todas as falhas foram detectadas e isoladas. Um único pico de alarme falso ocorreu quando a trajetória 3 (apêndice A9) foi apresentada em operação normal. Este resultado pode ser visto nas Figuras 41, 42, 43, 44, 45 e 46. Note que os sistemas que utilizam os métodos GRR, LRR e MAOK não apresentam o alarme falso.

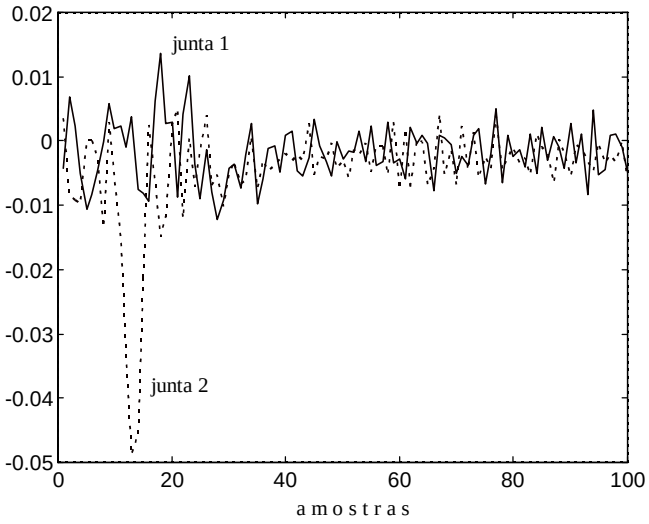

Figura 41. Resíduos para a trajetória livre de falhas de $\theta=[5\pi/12\ \pi/2]^T$ até $\theta=[\pi/6\ \pi/8]^T$.

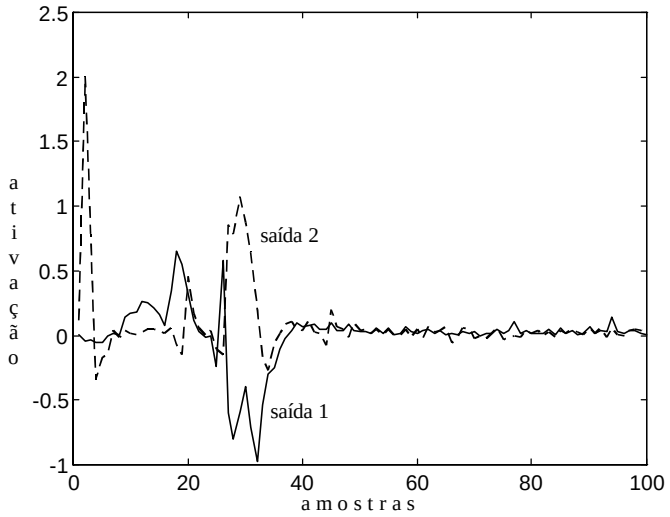

Figura 42. Saída da rede RBF treinada pelo método FS para a trajetória livre de falhas de $\theta=[5\pi/12\ \pi/2]^T$ até $\theta=[\pi/6\ \pi/8]^T$.

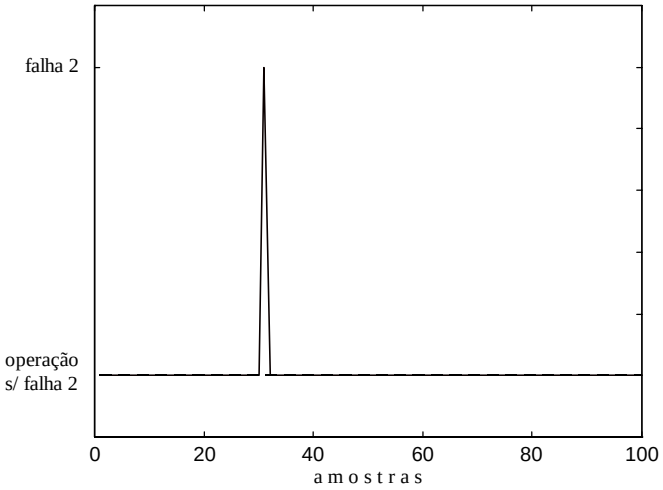

Figura 43. Detecção da falha 2 utilizando a rede RBF treinada pelo método FS (linha contínua) para a trajetória livre de falhas de $\theta=[5\pi/12\ \pi/2]^T$ até $\theta=[\pi/6\ \pi/8]^T$. A Falha 2 real está representada pela linha tracejada.

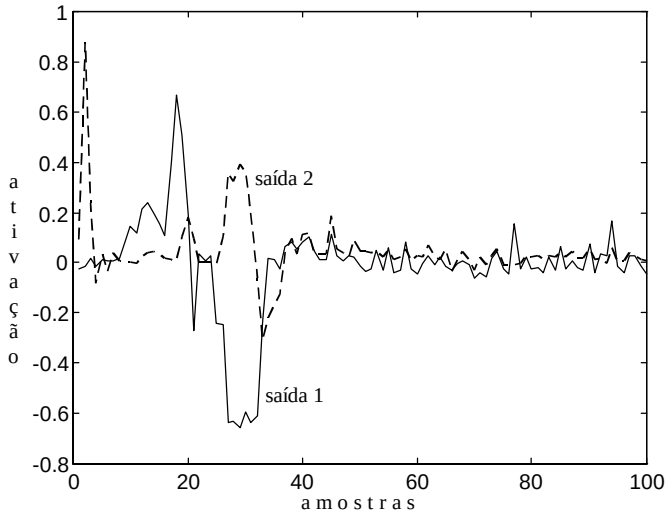

Figura 44. Saída da rede RBF treinada pelo método GRR para a trajetória $\theta=[5\pi/12\ \pi/2]^T$ até $\theta=[\pi/6\ \pi/8]^T$.

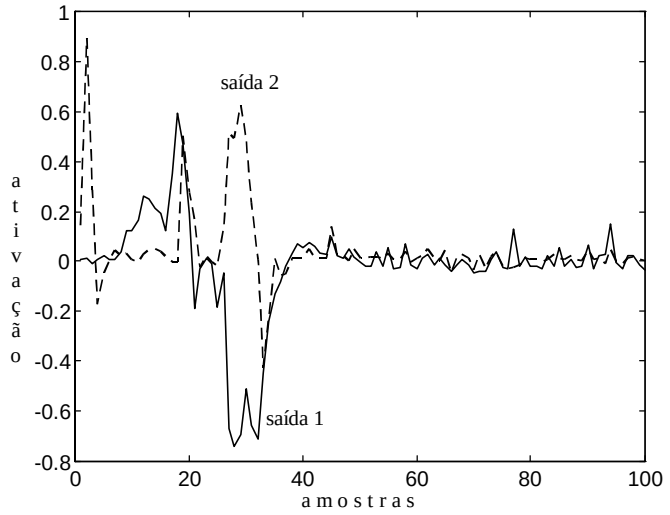

Figura 45. Saída da rede RBF treinada pelo método LRR para a trajetória $\theta=[5\pi/12\ \pi/2]^T$ até $\theta=[\pi/6\ \pi/8]^T$.

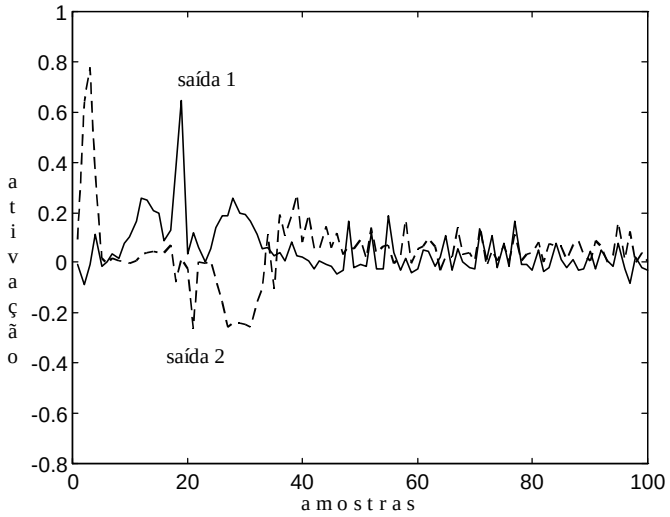

Figura 46. Saída da rede RBF treinada pelo método MAOK para a trajetória $\theta=[5\pi/12\ \pi/2]^T$ até $\theta=[\pi/6\ \pi/8]^T$.

5.2. MANIPULADOR PUMA 560

O segundo manipulador a ser estudado neste trabalho é o Puma 560, um robô comum tanto em ambientes industriais, como em ambientes acadêmicos [CORKE, 1994]. Este manipulador foi implementado em MATLAB através do *toolbox Robotics*, desenvolvido por CORKE (1996). Informações sobre os parâmetros de tal sistema podem ser encontrados em [CORKE & ARMSTRONG-HÉLOUVRY, 1994]. O período amostral utilizado é de 0,056 s. O robô, neste trabalho, utiliza o método do torque calculado para o controle de posicionamento das juntas e tem seis graus de liberdade: três para posicionamento e três para orientação.

5.2.1. Geração dos Resíduos

Para o treinamento do MLP, utiliza-se 150 trajetórias com 54 amostras cada, em que se considera o robô livre de falhas. Portanto, um total de 8100 padrões são usados no treinamento do MLP. As 20 primeiras trajetórias utilizadas no treinamento são escolhidas para que grande parte do espaço de posições das juntas seja coberto, e as trajetórias restantes são escolhidas aleatoriamente. A normalização é feita do mesmo modo que para o manipulador com 2 graus de liberdade. Aqui, os efeitos dos termos de fricção não foram considerados. Os parâmetros do treinamento por retropropagação do erro para o MLP podem ser vistos na Tabela 5. O vetor de entradas é formado pelas posições, velocidades angulares e torques das 3 primeiras juntas (posicionamento) em cada instante amostral *t* e o

vetor de saídas é formado pelas velocidades angulares das 3 primeiras juntas no instante amostral $t+\Delta t$.

Tabela 5. Parâmetros do treinamento por retropropagação do erro no MLP para o manipulador Puma 560.

Número de unidades na primeira camada	9
Número de unidades na camada escondida	29
Número de unidades na última camada	3
Função de ativação utilizada	sigmoidal
Taxa de aprendizagem	0,05
Momentum	0,90

Para o treinamento do MLP, um total de 8.000 épocas foram utilizadas. O erro médio quadrático do MLP depois deste período foi de 1,5795 x 10^{-5} (para o conjunto de treinamento). Neste trabalho, o MLP reproduz a dinâmica do sistema em operação normal com um resíduo baixo mesmo para trajetórias não-treinadas, comprovando a generalização do procedimento. A Figura 47 mostra os sinais resíduos (das velocidades das juntas) para uma trajetória não-treinada livre de falhas.

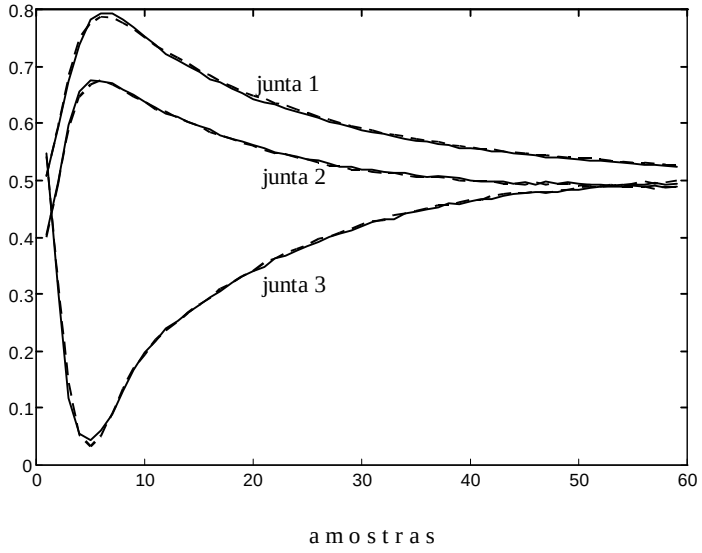

Figura 47. Velocidades reais normalizadas (linha tracejada) e saídas do MLP (linha contínua) para a trajetória livre de falhas de $\theta=[0\ 0\ 0\ 0\ 0\ 0]^T$ até $\theta=[\pi\ \pi/3\ -5\pi/6\ \pi/6\ \pi/2\ 5\pi/12]^T$.

Três tipos de falhas são considerados na simulação do sistema de DDF para o manipulador Puma 560. Na primeira (falha 1), o sinal de controle não é transmitido à junta 1. Na segunda (falha 2), o mesmo ocorre para a junta 2 e na terceira (falha 3), o mesmo ocorre para a junta 3. Estas falhas podem ter como causa, por exemplo, rompimentos dos fios que ligam o sistema de controle aos atuadores, falhas nas saídas do sistema de controle, falhas nos atuadores das juntas (por exemplo, o atuador deixa de funcionar) e falhas no sistema de engrenagens que liga o atuador à junta. Note que como o torque não é medido (supõe-se que o torque aplicado pelo sinal de controle seja transmitido às juntas), o sistema de controle continua aplicando o sinal de controle. Como existem inércia e forças gravitacionais no sistema, se não forem tomadas atitudes rápidas (por exemplo, freiar os elos), sérios danos podem ser causados ao manipulador, à carga do efetuador e ao ambiente de trabalho. A Figura 48 mostra os sinais de resíduo resultantes de 4 simulações de uma mesma trajetória com 12 amostras. Na primeira simulação (amostras 1 a 12), ocorre a partir da amostra 3 a falha 1, na segunda simulação (amostras 13 a 24) ocorre a partir da amostra 16 a falha 2, na terceira simulação (amostras 25 a 36) ocorre a partir da amostra 28 a falha 3 e, na última simulação, nenhuma falha ocorre. Foi usado um número pequeno de amostras (12) em cada simulação, pois quando ocorrem falhas deste tipo, se uma atitude não for tomada rapidamente, o sistema pode atingir suas restrições (por exemplo, bater no chão).

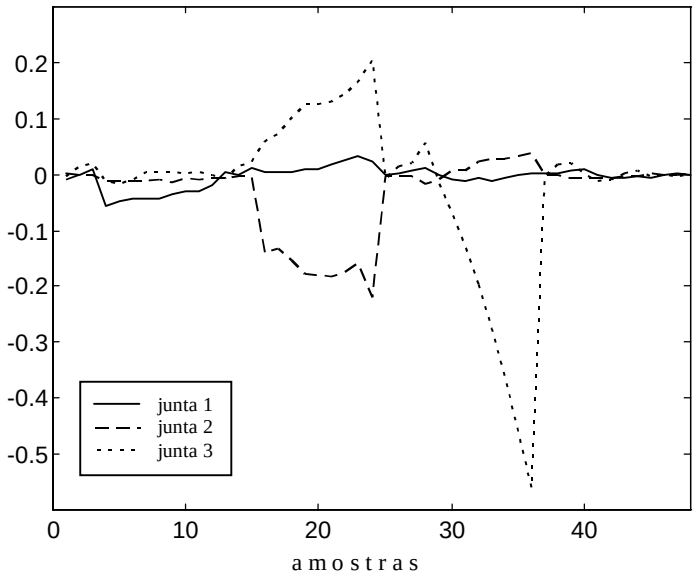

Figura 48. Sinais de resíduo para quatro simulações de uma mesma trajetória com 12 amostras de $\theta=[0\ 0\ 0\ 0\ 0\ 0]^T$ até $\theta=[\pi\ \pi/3\ -5\pi/6\ \pi/6\ \pi/2\ 5\pi/12]^T$. Em cada uma das 3 primeiras simulações, uma falha diferente ocorre e, na última, nenhuma falha ocorre.

5.2.2. Análise dos Resíduos

A rede RBF para o manipulador Puma 560 tem 6 entradas (resíduos e velocidades) e duas saídas (3 falhas). Para o treinamento da rede RBF, 15 trajetórias com 12 amostras cada são utilizadas. Estas trajetórias são apresentadas à rede RBF quatro vezes: uma em operação normal, uma em que ocorre a falha 1, uma em que ocorre a falha 2 e uma em que ocorre a falha 3, perfazendo um total de 720 padrões. Neste trabalho, o tamanho do campo receptivo, definido pela diagonal de **R**, é dado por $[0{,}022\ 0{,}027\ 0{,}027\ 0{,}8\ 0{,}8\ 0{,}8]^T$, no qual os três primeiros valores atuam sobre os resíduos e os três últimos atuam sobre as velocidades.

Para comparação de resultados, os quatro métodos descritos na seção 2.2 foram utilizados. O método FS utilizou o OLS para otimizar o algoritmo e o GCV como critério de parada. Durante o treinamento, foram selecionadas 251 unidades radiais.

Para o método GRR, utiliza-se como critério de convergência uma mudança relativa de 1/1000, ou seja, o método deve parar a seleção do parâmetro de regularização global ($\hat{\lambda}$) quando a mudança no valor do erro GCV entre duas iterações for menor que 1/1000. O valor selecionado foi $\hat{\lambda}$ = 0,5123.

Para o LRR, o critério de convergência utilizado foi o mesmo do GRR descrito acima. Foram "podadas" 425 unidades radiais da rede (estas unidades tiveram $\hat{\lambda}_j = \infty$). Assim, a rede utiliza somente 295 unidades radiais. Notou-se que para este treinamento, o LRR apresentou problemas de mau condicionamento.

No método MAOK, os parâmetros utilizados foram: $\alpha=t^{-1}$, σ=0,020 t^{-1} e t_{max}=1000 (número de vezes que cada padrão é apresentado). Usando tais parâmetros, 255 unidades radiais foram selecionadas. A Tabela 6 mostra o erro médio quadrático das três saídas da rede RBF (falha 1, falha 2 e falha 3) para o conjunto de treinamento.

Tabela 6. Erro médio quadrático da rede RBF para o conjunto de treinamento.

	saída 1 - rede RBF (Falha 1)	saída 2 - rede RBF (Falha 2)	saída 3 - rede RBF (Falha 3)
FS	0.0040	0.0017	0.0017
GRR	0.0155	0.0075	0.0119
LRR	0.0061	0.0022	0.0030
MAOK	0.0096	0.0042	0.0092

Para teste de validação dos sistemas de DDF resultantes, 30 trajetórias não-treinadas, com 15 amostras cada são apresentadas quatro vezes ao sistema (uma para operação normal

e uma para cada uma das três falhas), perfazendo um total de 120 trajetórias. A Tabela 7 mostra o erro médio quadrático das três saídas da rede RBF para o conjunto de teste. Os valores altos do erro para o LRR são devidos a problemas de mau condicionamento.

Tabela 7. Erro médio quadrático da rede RBF para o conjunto de teste.

	saída 1 - rede RBF (Falha 1)	saída 2 - rede RBF (Falha 2)	saída 3 - rede RBF (Falha 3)
FS	0.0712	0.0431	0.0347
GRR	0.0419	0.0371	0.0271
LRR	0.7930	241.7819	2.6576
MAOK	0.0462	0.0400	0.0293

As Figuras 49, 50, 51 e 52 mostram as saídas das rede RBF treinadas pelos quatro métodos para as quatro simulações da trajetória não-treinada cujos sinais de resíduos são mostradas na Figura 48. A saída 1 da rede RBF é responsável pela detecção da falha 1, a saída 2 é responsável pela detecção da falha 2 e a saída 3 é responsável pela detecção da falha 3.

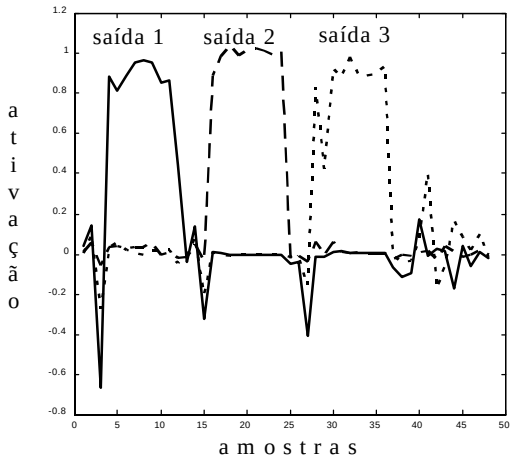

Figura 49. Saídas da rede RBF treinada por FS para os sinais de resíduo mostrados na Figura 48.

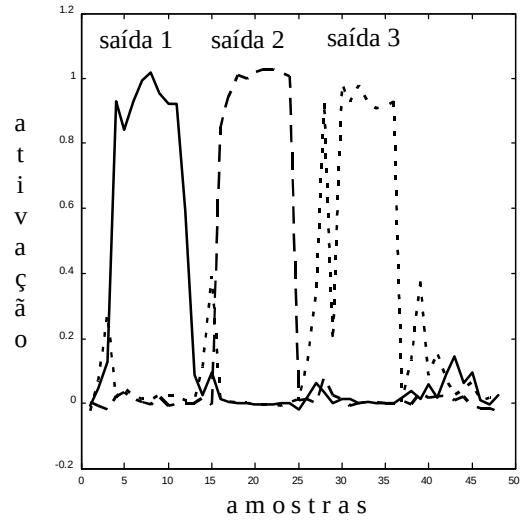

Figura 50. Saídas da rede RBF treinada por GRR para os sinais de resíduo mostrados na Figura 48.

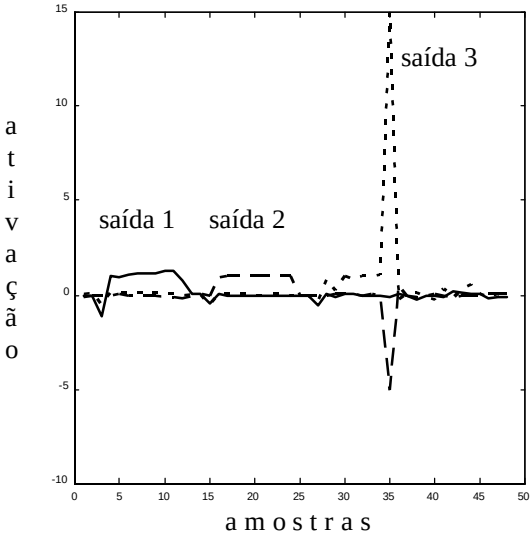

Figura 51. Saídas da rede RBF treinada por LRR para os sinais de resíduo mostrados na Figura 48.

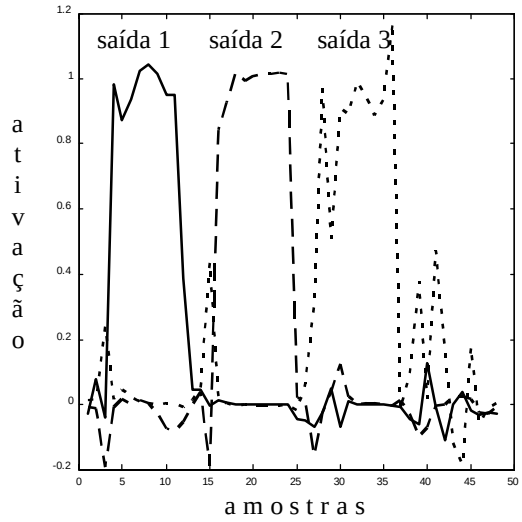

Figura 52. Saídas da rede RBF treinada por MAOK para os sinais de resíduo mostrados na Figura 48.

O critério de falhas adotado é o seguinte: para ser caracterizada uma falha, a saída da rede RBF tem que apresentar três sinais seguidos maiores que 0,5. Isto não causa uma perda de sensibilidade significativa na detecção das falhas porque o período amostral é baixo. A Tabela 8 mostra os resultados da DDF após a aplicação do critério de detecção de falhas nos padrões do conjunto de teste. Salienta-se que estes resultados poderiam ser melhores se o número de padrões utilizados no conjunto de treinamento fosse maior.

Tabela 8. Resultados dos testes para o sistema de DDF aplicado ao Puma 560.

	número de alarmes falsos	número de falhas não-detectadas
FS	8	2
GRR	1	0
LRR	7	1
MAOK	4	0

Capítulo 6

Conclusões

O sistema de DDF via RNA apresenta bons resultados quando aplicado a um robô manipulador planar com dois elos rígidos e a um manipulador Puma560. Salienta-se que o grande atrativo é que, de acordo com os testes, o sistema de DDF consegue detectar e isolar as falhas que ocorrem em trajetórias não-vistas e em instantes diferentes daqueles dos padrões treinados. Deve-se observar também que os resultados apresentados para ambos os casos (principalmente para o Puma 560) podiam ser melhores se o número de padrões utilizados nos treinamentos das redes RBF fosse maior.

O MLP reproduz o comportamento dinâmico do sistema sem falhas produzindo geralmente um resíduo baixo. A qualidade do treinamento do MLP é muito importante para os futuros resultados do sistema de DDF, quando padrões não-treinados deverão ser generalizados. Um erro do mapeamento grande resulta em resíduos altos e, consequentemente, alarmes falsos e erros de classificação das falhas.

As redes RBF treinadas pelos quatro métodos apresentaram comportamentos diferentes quando aplicadas ao problema de DDF. Neste trabalho, as redes RBF treinadas pelo método FS apresentaram os menores erros médios quadráticos para os conjuntos de treinamento. Isso ocorre porque as redes RBF treinadas por FS apresentam os menores *bias* nos erros quadráticos médios. Contudo, as redes RBF treinadas pelos outros métodos apresentaram erros menores nos conjuntos de testes, indicando que as variâncias dos erros foram menores. Tal fato indica que as redes RBF treinadas por FS foram mais sensíveis às peculiaridades (tal qual ruídos e escolha dos padrões) dos conjuntos de treinamento. Isso ocorre porque os centros das unidades radiais das redes treinadas via FS são escolhidas a partir das posições dos padrões de treinamento.

Para as redes treinadas por GRR e LRR, apesar de os centros das unidades radiais serem escolhidos a partir dos padrões de treinamento, os pesos grandes foram penalizados,

produzindo um efeito de suavidade na função de saída das redes (pesos grandes geralmente são requeridos para produzir funções de saída que variam bruscamente). Isto ocorre porque o *bias* introduzido nos métodos GRR e LRR favorece a solução envolvendo pesos pequenos. O LRR deveria adaptar a suavidade às condições locais (através de parâmetros de regularização individuais) do espaço de entradas da rede RBF. O método "poda" algumas conexões da rede através de um grande aumento nos valores dos parâmetros de regularização das unidades radiais associadas a estas conexões, o que explica os melhores resultados de generalização do GRR. Isto deixa o método de certa forma parecido com o FS. Todavia, o uso de parâmetros de regularização suaviza a função de saída da rede. Note que o método FS selecionou mais unidades radiais que o método LRR, o que deveria tornar a função de saída da rede mais suave. Isto não acontece, no entanto, porque o método GRR faz uso dos parâmetros de regularização, fazendo com que os pesos utilizados sejam menores.

Apesar de a rede RBF treinada pelo método GRR apresentar melhores resultados para o Puma 560, as redes RBF treinadas por MAOK apresentaram os melhores resultados para os conjuntos de teste, indicando uma melhor generalização. Isto ocorre porque, diferentemente dos outros métodos, MAOK fixa os centros em posições diferentes daquelas dos padrões de treinamento. Tais posições, próximas dos centros dos aglomerados de padrões (*clusters*), são mais interessantes pois indicam um comportamento coletivo (dos padrões do aglomerado) ao invés de comportamentos individuais (dos padrões de treinamento isolados). No caso do Puma 560, a rede treinada por GRR apresentou os melhores resultados de generalização porque houve uma grande penalização nos pesos grandes, gerando um efeito de suavidade na função de saída da rede RBF. No entanto, o resultado da rede RBF treinada por MAOK foi bastante próxima, apesar desta utilizar um número menor de unidades radiais (255 do MAOK contra 720 do GRR). O número de unidades radiais selecionadas indica a complexidade da operação da rede RBF, sendo um número menor de unidades radiais mais interessante (se o número de unidades radiais é grande, o esforço computacional para os diversos cálculos envolvendo a rede RBF e a memória requerida são maiores). Assim, o número de unidades radiais é um bom indicador da velocidade de operação da rede RBF. Um aspecto negativo do método MAOK é a escolha dos parâmetros de treinamento, que influenciam drasticamente o desempenho da rede RBF resultante.

Salienta-se, também, que durante o treinamento, o esforço computacional do MAOK é baixo pois este método não emprega cálculos complexos envolvendo matrizes grandes. O

algoritmo que apresenta o esforço computacional maior é o LRR (porque precisa calcular todos os parâmetros de regularização individuais), seguido do GRR. O algoritmo FS apresenta um esforço computacional relativamente baixo porque faz uso de modelos reduzidos (com um número menor de unidades radiais).

A definição do tamanho do campo receptivo das redes RBF tem um papel importante na generalização dos resultados. Como exemplo, na rede RBF treinada pelo método FS, se um campo receptivo muito grande é definido, o método escolhe poucas unidades radiais conseguindo classificar apenas os padrões nos grandes aglomerados e, portanto, identificando erroneamente os padrões isolados. Se o campo receptivo é muito pequeno, um grande número de unidades radiais é escolhido, resultando em uma sobre-sensibilidade aos detalhes do conjunto particular de treinamento e, portanto, em uma baixa generalização. Neste trabalho, a escolha dos parâmetros que definem o tamanho do campo receptivo foram feitos heuristicamente.

Os resultados podem ser generalizados para um maior número de falhas. Entretanto, falhas cujo sistema apresenta comportamentos semelhantes, independentemente dos graus de liberdade do robô, são difíceis de serem isoladas (apesar de serem detectadas) porque os padrões de resíduo devem ocupar as mesmas regiões do espaço de entradas.

Referências bibliográficas

Allen, D. M. (1974). “The relationship between variable selection and data augmentation and a method for prediction ”, *Technometrics*, vol. 16, 1, p.125-127.

Bergerman, M. (1996). “*Dynamics and control of underactuated manipulators*”, PhD Dissertation, Carnegie Mellon University, 129p., Pittsburgh, USA.

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

Brown, M. and Harris, C. (1994). “*Neurofuzzy adaptive modelling and control ”*, Prentice Hall International.

Calado, J. M. F. and Roberts, P. D. (1997). “Fault detection and diagnosis based on fuzzy qualitative”, *Proceedings of the IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes*, Kingston Upon Hull, U. K., vol.1, p.534-539.

Caminhas, W. M. (1997). “*Estratégias de detecção e diagnóstico de falhas em sistemas dinâmicos*”, Tese (Doutorado), Universidade Estadual de Campinas, 161p., Campinas.

Caminhas, W. M., Tavares, H. M. F. e Gomide, F. (1996). "Rede lógica neurofuzzy: aplicação em diagnóstico de falhas em sistemas dinâmicos", *Anais do XI Congresso Brasileiro de Automática*, São Paulo, Brasil, p.459-464.

Charniak, E. and McDermott, D. V. (1985). “*Introduction to artificial intelligence*”, Addison-Wesley Publishing Company Inc.

Chen, S., Cowan, C. F. N. and Grant, P. M. (1991). “Orthogonal least squares learning algorithm for radial basis function networks”, *IEEE Transactions on Neural Networks*, vol. 2, 2, p.302-309.

Chow, E. Y. and Willsky, A. S. (1984). “Analytical redundancy and design of robust failure detection systems”, *IEEE Transactions on Automatic Control*, 29, p.603-614.

Corke, P. I. (1994). "*The Unimation Puma servo system*", Technical Report MTN-226, CSIRO Division of Manufacturing Technology, Australia.

Corke, P. I. (1996). "A robotics toolbox for MATLAB", *IEEE Transactions on Robotics & Automation Magazine*, vol. 3, n. 1, p.24-32.

Corke, P. I., and Armstrong-Hélouvry, B. S. (1994). "A search for consensus among model parameters reported for Puma 560 robot", *In the Proceedings of the IEEE Conference on Robotics and Automation*, San Diego, USA.

Craig, J. J. (1988). "*Introduction to robotics: Mechanics and control*", Addison-Wesley.

Cybenko, G. (1989). "Approximation by superpositions of a sigmoidal function", *Mathematics of Control, Signals, and Systems*, 2, p.303-314.

Efrati, H. F. T. (1997). "Tracking of mechanical systems using artificial neural networks", *Revista Brasileira de Ciências Mecânicas*, vol. 19, n. 2, p.217-227.

Evsukoff, A., Montmain, J. and Gentil, S. (1997). "Dynamic model and causal knowledge-based fault detection and isolation", *Proceedings of the IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes*, Kingston Upon Hull, U. K., vol. 2, p.699-704.

Frank, P. M. (1987). "Fault diagnosis in dynamic systems via state estimation - A survey", In Tsafestas, S. M. Singh and G. Schmidt (Eds.), *System Fault Diagnostics, Reliability and Related Knowledge-Based Approaches*, vol. 1, p.35-98.

Frank, P. M. (1990). "Fault Diagnosis in Dynamic Systems Using Analytical and Knowledge-based Redundancy - A Survey and Some New Results". *Automatica*, vol. 26, 3, p.459-474.

Füssel, D., Ballé, P. and Isermann, R. (1997). "Closed-loop fault diagnosis based on a nolinear process model and automatic fuzzy rule generation", *Proceedings of the IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes*, Kingston Upon Hull, U. K., vol.1, p.359-364.

Geman, S., Bienenstock, E. and Doursat, R. (1992). "Neural networks and the bias/variance dilemma", *Neural Computation*, vol. 4, 1, p.1-58.

Gertler, J. (1988). "A survey of model-based failure detection and isolation in complex plants", *IEEE Control Systems Magazine*, vol. 8, n. 6, p.3-11.

Gertler, J. (1991). “Analytical redundancy methods in fault detection and isolation - Survey and synthesis”, *Proceedings of the IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes*, Baden Baden, Germany, vol. 1, p.9-23.

Gertler, J. (1997). “A cautious look at robustness in residual generation”, *Proceedings of the IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes*, Kingston Upon Hull, U. K., vol.1, p.133-139.

Golub, G. H., Heath, M. and Wahba, G. (1979). “Generalised cross-validation as a method for choosing a good ridge parameter”, *Technometrics*, vol. 21, 2, p.215-223.

Haykin, S. S. (1994). “*Neural networks: a comprehensive foundation*”, New York: Macmillan.

Hertz, J., Krogh, A. and Palmer, R. G. (1991). “*Introduction to the theory of neural computation*”, Addison-Wesley Publishing Company.

Hoerl, A. E. and R. W. Kennard (1970). “Ridge regression: biased estimation for nonorthogonal problems”, *Technometrics*, vol. 12, 3, p.55-67.

Horn, R. A. and Johnson, C. R. (1985). “*Matrix analysis*”, Cambridge University Press.

Horak, D. T. (1988). “Failure detection in dynamic systems with modeling errors”, *AIAA Journal of Guidance, Control and Dynamics*, vo. 11, 6, p.508-516.

Isermann, R. (1984). “Process fault detection based on modeling and estimation methods - A survey”, *Automatica*, vol. 20, 4, p.387-404.

Kohonen, T. (1995). “*Self-organizing maps*”, Springer-Verlag, Berlin.

Köppen-Seliger, B. and Frank, P. M. (1996). "Neural Networks in Model-Based Fault Diagnosis", *Proceedings of the 13th IFAC World Congress*, San Francisco, USA, p.67-72.

Korbicz, J. (1997). “Neural networks and their application in fault detection and diagnosis”, *Proceedings of the IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes*, Kingston Upon Hull, U. K., vol.1, p.377-382.

Leonard , J. A. and Kramer, M. A. (1991). “Radial basis function networks for classifying process faults”, *IEEE Control Systems*, vol. 11, 3, p.31-38.

Miller, A. J. (1990). “*Subset Selection in Regression*”, Chapman and Hall.

Looney, C. G. (1997). "*Pattern recognition using neural networks*", Oxford University Press.

Mangoubi, R., Appleby, B. D. and Farrell, J. (1992). "Robust estimation in fault detection", *Proceedings of the 31st IEEE Conference on Decision and Control*, p.3943-3948.

Moody, J. and Darken, C. (1989). "Fast learning in networks of locally-tuned processing units", *Neural Computation*, 1, p.289-303.

Naughton, J. M., Chen, Y. C. and Jiang, J. (1996). "A neural network aplication to fault diagnosis for robotic manipulator", *Proceedings of IEEE International Conference on Control Applications*, vol.1, p.988-1003.

Nelles, O. and Isermann, R. (1995). "A comparison between RBF networks and classical methods for identification of nonlinear dynamic systems", *Proceedings of the IFAC Congress on Adaptive System in Control and Sygnal Processing*, Budapest, Hungary, p.233-238.

Ojala, T. and Vuorimaa, P. (1995). "Modified Kohonen's learning laws for RBF networks", *Proceedings of the International Conference on Artificial Neural Nets and Genetic Algorithms*, Ales, France, p.356-359.

Orr, M. J. L. (1995 a). "Regularization in the selection of radial basis function centers", *Neural Computation*, 7, p.606-623.

Orr, M. J. L. (1995 b). "Local smoothing of radial basis function networks", *Proceedings of the 1995 International Symposium on Neural Networks*, Hsinchu, Taiwan.

Orr, M. J. L. (1996 a). "*Introduction to radial basis function networks*", Technical Report, Center for Cognitive Science, Edinburgh University, Scotland, U. K.

Orr, M. J. L. (1996 b). "*MATLAB routines for subset selection and ridge regression in linear neural networks*", Technical Report, Center for Cognitive Science, Edinburgh University, Scotland, U. K.

Ostojic, M. (1996). "Recursive control of robotic motion", *International Journal of Control*, vol. 64, n. 5, p.775-787.

Patton, R. J. (1994). "Robust model-based fault diagnosis: the state of the art", *Proceedings of the IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes*, Helsinki, Finland, vol. 1, p.1-24.

Patton, R. J.(1997). "Fault-tolerant control: The 1997 situation", *Proceedings of the IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes*, Kingston Upon Hull, U. K., vol.2, p.1033-1055.

Patton, R. J., Frank, P. M. e Clark, R. N. (1989). "*Fault diagnosis in dynamic systems - theory and application*", Prentice Hall International.

Press, W. H., Teukolsky, S. A., Vetterling, W. T. and Flannery, B. P. (1992). "*Numerical recipes in C*", Cambridge University Press, Cambridge, U. K., 2nd ed.

Rawlings, J. O. (1988). "*Applied regression analysis - a research tool*", Wadsworth & Brooks / Cole Advanced Books & Software.

Sadrnia, M.A, Chen, J. and Patton, R. J. (1997 a). "Robust fault detection observer design using H_∞/μ techniques for uncertain flight control systems", *Proceedings of the 2nd IFAC Symposium on Robust Control Design*, Budapest, Hungary, p.531-536.

Sadrnia, M.A, Chen, J. and Patton, R. J. (1997 b). "Robust H_∞/μ observer-based residual generation for fault diagnosis", *Proceedings of the IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes*, Kingston Upon Hull, U. K., vol.1, p.147-153.

Schneider, H. and Frank, P. M. (1996). "Observer-based supervision and fault-detection in robots using nonlinear and fuzzy logic residual evaluation", *IEEE Transactions on Control Systems Technology*, vol. 4, 3, p.274-282.

Sciavicco, L. and Siciliano, B. (1996). "*Modeling and control of robot manipulators*", McGraw-Hill.

Stengel, R. F. (1991). "Intelligent failure-tolerant control", *IEEE Control Systems*, vol.11, 4, p.14-23.

Tikhonov, A. N. and Arsenin, V. Y. (1977). "*Solutions of ill-posed problems*", Winston, Washington.

Vemuri, A. T. and Polycarpou, M. M. (1997). "Neural-network-based robust fault diagnosis in robotic systems", *IEEE Transaction on Neural Networks*, vol. 8, 6, p.1410-1420.

Visinsky, M. L., Cavallaro, J. R. and Walker, I. D. (1994). "Robotic fault detection and fault tolerance: A survey", *Reliability Engineering and System Safety*, 46, p.139-158.

Visinsky, M. L., Cavallaro, J. R. and Walker, I. D. (1995). “A dynamic fault tolerance framework for remote robots”. *IEEE Transactions on Robotics and Automation*, vol. 11, 4, p.477-490.

Warwick, K. and Craddock, R. (1996). “An introduction to radial basis function for system identification: A comparison with other neural networks”, *Conference on Decision and Control*, Kobe, Japan, p.464-469.

Watanabe, K. and Himmelblau, D. M. (1982). “Instrument fault detection in systems with uncertainties”, *International Journal of Systems Science*, 13, p.137-158.

Willsky, A. S. (1976). “A survey of design methods for failure detection in dynamic systems”, *Automatica*, vol. 12, 6, p.601-611.

Apêndice A1

Dados utilizados no exemplo 1

Tabela A1.1. Treinamento do MLP por retropropagação do erro para o exemplo 1.

Número de unidades na primeira camada	2
Número de unidades na camada escondida	6
Número de unidades na última camada	3
Função de ativação utilizada	sigmoidal
Taxa de aprendizagem	0,08
Momentum	0,1

Tabela A1.2. Treinamento da rede RBF para o exemplo 1.

Número de unidades na primeira camada	2
Número de unidades na última camada	3
Função de ativação utilizada nas unidades radiais	Gaussiana
Método utilizado para treinamento	*global ridge regression*
R (define o tamanho do campo receptivo)	$\begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix}$
Parâmetro de regularização (λ) selecionado	$1{,}21 \times 10^{-5}$

Apêndice A2

A matriz de pesos ótima para a rede RBF

Para o cálculo da matriz de pesos ótima, a função de custo será minimizada. Como as análises para os diferentes métodos são similares, será considerada a função de custo mais complexa: a do método LRR. No entanto, o resultado pode ser generalizado para o GRR fazendo $\lambda_j = \lambda$ para $j = 1,...,m$. Para o FS e o MAOK basta fazer $\lambda_j = 0$ para $j = 1,...,m$. Sabe-se que a função de custo para o método LRR é dada por (equação 37)

$$C_k = \sum_{n=1}^{n_p} \left(\psi(n) - \hat{\psi}(n)\right)^2 + \sum_{j=1}^{m} \lambda_j \, \omega_{kj}^2$$

na qual $k = 1,...,q$. O que se deseja é achar o extremo da função. Assim, primeiro a função de custo será diferenciada e em seguida será igualada a zero. Por simplicidade, será considerada uma única saída para a rede RBF (q=1). Diferenciando a função de custo, tem-se para a unidade radial j

$$\frac{\partial C}{\partial \omega_j} = 2\sum_{n=1}^{n_p} \left(\hat{\psi}(n) - \psi(n)\right) \frac{\partial \hat{\psi}(n)}{\partial \omega_j} + 2\lambda_j \omega_j .$$

Como o modelo é linear, da Eq. (25)

$$\frac{\partial \hat{\psi}(n)}{\partial \omega_j} = h_j(n).$$

Substituindo na equação anterior e igualando a zero

$$\frac{\partial C}{\partial \omega_j} = 2\sum_{n=1}^{n_p} \left(\hat{\psi}(n) - \psi(n)\right) h_j(n) + 2\lambda_j \hat{\omega}_j = 0$$

$$2\sum_{n=1}^{n_p} \hat{\psi}(n) h_j(n) - 2\sum_{n=1}^{n_p} \psi(n) h_j(n) + 2\lambda_j \hat{\omega}_j = 0$$

$$\sum_{n=1}^{n_p} \hat{\psi}(n) h_j(n) + \lambda_j \hat{\omega}_j = \sum_{n=1}^{n_p} \psi(n) h_j(n).$$

Existem *m* destas equações (*j* = 1,...,*m*), tal que cada uma representa uma limitação na solução. Como existem o número exato de restrições e de equações desconhecidas, deve existir uma única solução. A solução única pode ser encontrada através de álgebra linear. Usando a notação vetorial na equação anterior, tem-se para a unidade radial *j*

$$\mathbf{h}_j^{\mathrm{T}} \hat{\boldsymbol{\psi}} + \lambda_j \hat{\omega}_j = \mathbf{h}_j^{\mathrm{T}} \boldsymbol{\psi}$$

na qual $\mathbf{h}_j = \left[h_j(1) \quad h_j(2) \quad \ldots \quad h_j(n_p)\right]^{\mathrm{T}}$, $\hat{\boldsymbol{\psi}} = \left[\hat{\psi}(1) \quad \hat{\psi}(2) \quad \ldots \quad \hat{\psi}(n_p)\right]^{\mathrm{T}}$ e, $\boldsymbol{\psi} = \left[\psi(1) \quad \psi(2) \quad \ldots \quad \psi(n_p)\right]^{\mathrm{T}}$. Considerando a equação acima para todas as unidades radiais, ou seja, *j* = 1,...,m, tem-se

$$\begin{bmatrix} \mathbf{h}_1^{\mathrm{T}} \hat{\boldsymbol{\psi}} \\ \mathbf{h}_2^{\mathrm{T}} \hat{\boldsymbol{\psi}} \\ \vdots \\ \mathbf{h}_m^{\mathrm{T}} \hat{\boldsymbol{\psi}} \end{bmatrix} + \begin{bmatrix} \lambda_1 \hat{\omega}_1 \\ \lambda_2 \hat{\omega}_2 \\ \vdots \\ \lambda_m \hat{\omega}_m \end{bmatrix} = \begin{bmatrix} \mathbf{h}_1^{\mathrm{T}} \boldsymbol{\psi} \\ \mathbf{h}_2^{\mathrm{T}} \boldsymbol{\psi} \\ \vdots \\ \mathbf{h}_m^{\mathrm{T}} \boldsymbol{\psi} \end{bmatrix}. \qquad \text{(A2.1)}$$

Fazendo

$$\mathbf{H} = \left[\mathbf{h}_1 \quad \mathbf{h}_2 \quad \ldots \quad \mathbf{h}_m\right],$$

$$\hat{\boldsymbol{\omega}} = \left[\hat{\omega}_1 \quad \hat{\omega}_2 \quad \ldots \quad \hat{\omega}_m\right]^{\mathrm{T}} \text{ e,}$$

$$\boldsymbol{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & \ldots & 0 \\ 0 & \lambda_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \lambda_m \end{bmatrix}$$

a Eq. A2.1 fica

$$\mathbf{H}^{\mathrm{T}} \hat{\boldsymbol{\psi}} + \boldsymbol{\Lambda}\, \boldsymbol{\omega} = \mathbf{H}^{\mathrm{T}} \boldsymbol{\psi}. \qquad \text{(A2.2)}$$

Sabe-se que a saída da rede RBF (Eq. 25) é

$$\hat{\psi}\,(n) = \sum_{j=1}^{m} \omega_{\,j} h_j(n).$$

Passando para a notação vetorial

$$\hat{\psi}(n) = \bar{\mathbf{h}}_n^{\mathrm{T}}\,\hat{\boldsymbol{\omega}}$$

na qual: $\bar{\mathbf{h}}_n = \begin{bmatrix} h_1(n) & h_2(n) & \ldots & h_m(n) \end{bmatrix}^{\mathrm{T}}$ e

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}_1 & \mathbf{h}_2 & \ldots & \mathbf{h}_m \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{h}}_1 & \bar{\mathbf{h}}_2 & \ldots & \bar{\mathbf{h}}_{n_p} \end{bmatrix}^{\mathrm{T}}.$$

Para todo o conjunto de treinamento, ou seja $n = 1, ..., n_p$, tem-se

$$\hat{\boldsymbol{\psi}} = \begin{bmatrix} \hat{\psi}(1) \\ \hat{\psi}(2) \\ \vdots \\ \hat{\psi}(n_p) \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{h}}_1^{\mathrm{T}}\,\hat{\boldsymbol{\omega}} \\ \bar{\mathbf{h}}_2^{\mathrm{T}}\,\hat{\boldsymbol{\omega}} \\ \vdots \\ \bar{\mathbf{h}}_{n_p}^{\mathrm{T}}\,\hat{\boldsymbol{\omega}} \end{bmatrix} = \mathbf{H}\,\hat{\boldsymbol{\omega}}.$$

Substituindo na equação A2.2, tem-se

$$\mathbf{H}^{\mathrm{T}}\,\mathbf{H}\,\hat{\boldsymbol{\omega}} + \boldsymbol{\Lambda}\,\hat{\boldsymbol{\omega}} = \mathbf{H}^{\mathrm{T}}\boldsymbol{\psi}$$

$$\left(\mathbf{H}^{\mathrm{T}}\,\mathbf{H} + \boldsymbol{\Lambda}\right)\hat{\boldsymbol{\omega}} = \mathbf{H}^{\mathrm{T}}\boldsymbol{\psi}$$

$$\hat{\boldsymbol{\omega}} = \left(\mathbf{H}^{\mathrm{T}}\,\mathbf{H} + \boldsymbol{\Lambda}\right)^{-1}\mathbf{H}^{\mathrm{T}}\boldsymbol{\psi}$$

que é o vetor de pesos ótimo para o método LRR. Considerando-se agora q>1 (várias neurônios de saída na rede RBF) tem-se a matriz de pesos ótima

$$\hat{\boldsymbol{\Omega}} = \left(\mathbf{H}^{\mathrm{T}}\,\mathbf{H} + \boldsymbol{\Lambda}\right)^{-1}\mathbf{H}^{\mathrm{T}}\boldsymbol{\Psi}$$

na qual: $\hat{\boldsymbol{\Omega}} = \left[\hat{\boldsymbol{\omega}}_1 \quad \hat{\boldsymbol{\omega}}_2 \quad \ldots \quad \hat{\boldsymbol{\omega}}_q\right]$,

$\hat{\boldsymbol{\omega}}_k = \left[\hat{\omega}_{k1} \quad \hat{\omega}_{k2} \quad \ldots \quad \hat{\omega}_{km}\right]^{\mathrm{T}}$,

$\boldsymbol{\Psi} = \left[\boldsymbol{\psi}_1 \quad \boldsymbol{\psi}_2 \quad \ldots \quad \boldsymbol{\psi}_q\right]$ e,

$\boldsymbol{\psi}_k = \left[\psi_k(1) \quad \psi_k(2) \quad \ldots \quad \psi_k\left(n_p\right)\right]^{\mathrm{T}}$.

Para o método GRR, basta fazer $\lambda_j = \lambda$ para j = 1,...,m. Portanto, $\boldsymbol{\Lambda}=\lambda\,\mathbf{I}_m$, na qual $\mathbf{I}_m$ é uma matriz identidade de tamanho m. Assim, a matriz de pesos ótima para o método GRR é

$$\hat{\boldsymbol{\Omega}} = \left(\mathbf{H}^{\mathrm{T}}\,\mathbf{H} + \lambda\,\mathbf{I}_m\right)^{-1}\mathbf{H}^{\mathrm{T}}\boldsymbol{\Psi}\,.$$

Quando a penalização dos pesos não é considerada (como no método FS e no método MAOK), basta fazer $\lambda_j = 0$ para j = 1,...,m. Assim, a matriz de pesos ótima é dada por

$$\hat{\boldsymbol{\Omega}} = \left(\mathbf{H}^{\mathrm{T}}\,\mathbf{H}\right)^{-1}\mathbf{H}^{\mathrm{T}}\boldsymbol{\Psi}\,.$$

Apêndice A3

A matriz de projeção

Esta matriz (**P**) projeta vetores de um espaço n_p-dimensional (vetor de saídas desejadas de cada neurônio de saída) perpendicular para um subspaço m-dimensional gerado pelo modelo dos mínimos quadráticos sem a penalização dos pesos. Considere que existe apenas um neurônio de saída (q=1) e que não existe penalização (como nos métodos FS e MAOK). O conjunto de vetores de saídas desejadas $\boldsymbol{\psi}$ está em um espaço n_p-dimensional (n_p é o número de padrões de treinamento). Contudo, o modelo sendo linear e possuindo somente m graus de liberdade (os m pesos), pode somente atingir pontos em um hiperplano m-dimensional, que é um subspaço do espaço n_p-dimensional (se $m \leq n_p$). Por exemplo, se n_p=3 e m=2 o modelo pode somente atingir pontos posicionados em algum plano fixado e nunca pode igualar $\boldsymbol{\psi}$ se este está fora de um plano (veja Figura A3.1). O principio dos mínimos quadráticos implica que o modelo ótimo é aquele com a mínima distância do vetor $\boldsymbol{\psi}$ até o plano, ou seja, o vetor ($\mathbf{P}\,\boldsymbol{\psi}$).

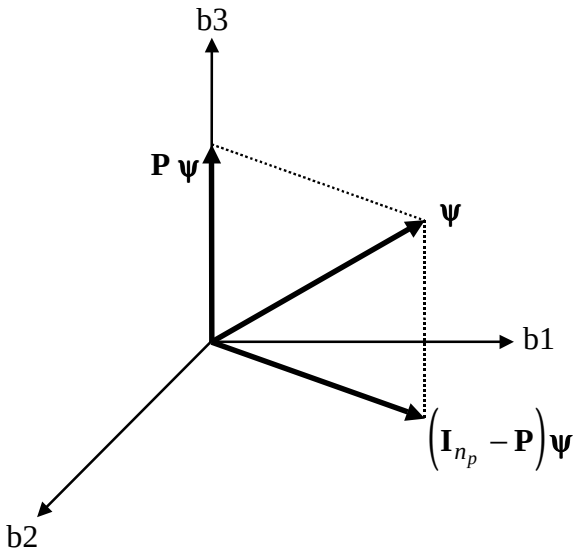

Figura A3.1. O modelo gera um plano (com as bases b1 e b2) em um espaço tridimensional (que utiliza a base b3) e não pode igualar ψ. O modelo dos mínimos quadráticos é a projeção de ψ no plano gerado e o vetor de erro é dado pelo vetor P ψ.

Sabe-se que a saída da rede RBF (equação 25) é dada por

$$\hat{\psi}(n) = \sum_{j=1}^{m} \omega_j h_j(n)$$

Passando para a notação vetorial

$$\hat{\psi}(n) = \bar{\mathbf{h}}_n^{\mathrm{T}}\, \hat{\boldsymbol{\omega}}$$

na qual: $\bar{\mathbf{h}}_n = \begin{bmatrix} h_1(n) & h_2(n) & \dots & h_m(n) \end{bmatrix}^{\mathrm{T}}$ e

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}_1 & \mathbf{h}_2 & \dots & \mathbf{h}_m \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{h}}_1 & \bar{\mathbf{h}}_2 & \dots & \bar{\mathbf{h}}_{n_p} \end{bmatrix}^{\mathrm{T}}.$$

Para $n = 1, ..., n_p$, tem-se

$$\hat{\boldsymbol{\psi}} = \begin{bmatrix} \hat{\psi}(1) \\ \hat{\psi}(2) \\ \vdots \\ \hat{\psi}(n_p) \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{h}}_1^{\mathrm{T}}\, \hat{\boldsymbol{\omega}} \\ \bar{\mathbf{h}}_2^{\mathrm{T}}\, \hat{\boldsymbol{\omega}} \\ \vdots \\ \bar{\mathbf{h}}_{n_p}^{\mathrm{T}}\, \hat{\boldsymbol{\omega}} \end{bmatrix} = \mathbf{H}\, \hat{\boldsymbol{\omega}}\,.$$

Sabe-se que o vetor de pesos ótimo é dado por (Apêndice A2)

$$\hat{\boldsymbol{\omega}} = \left(\mathbf{H}^{\mathrm{T}}\, \mathbf{H} + \boldsymbol{\Lambda}\right)^{-1} \mathbf{H}^{\mathrm{T}} \boldsymbol{\psi}\,.$$

Portanto

$$\hat{\boldsymbol{\psi}} = \mathbf{H}\left(\mathbf{H}^{\mathrm{T}}\mathbf{H} + \boldsymbol{\Lambda}\right)^{-1} \mathbf{H}^{\mathrm{T}}\, \boldsymbol{\psi}\,.$$

A matriz de variância é definida como

$$\mathbf{A}^{-1} = \left(\mathbf{H}^{\mathrm{T}}\mathbf{H} + \boldsymbol{\Lambda}\right)^{-1}.$$

Assim:

$$\hat{\boldsymbol{\psi}} = \mathbf{H}\mathbf{A}^{-1}\mathbf{H}^{\mathrm{T}}\, \boldsymbol{\psi}\,.$$

O erro entre a predição realizada pela rede RBF a partir do conjunto de treinamento e a saída observada é dado por

$$\boldsymbol{\psi} - \hat{\boldsymbol{\psi}} = \boldsymbol{\psi} - \mathbf{H}\mathbf{A}^{-1}\mathbf{H}^{\mathrm{T}}\boldsymbol{\psi}$$

$$\boldsymbol{\psi} - \hat{\boldsymbol{\psi}} = \left(\mathbf{I}_{n_p} - \mathbf{H}\mathbf{A}^{-1}\mathbf{H}^{\mathrm{T}}\right)\boldsymbol{\psi}$$

$$\boldsymbol{\psi} - \hat{\boldsymbol{\psi}} = \mathbf{P}\,\boldsymbol{\psi} \qquad \text{(A3.1)}$$

na qual **P** é a matriz de projeção dada por

$$\mathbf{P} = \left(\mathbf{I}_{n_p} - \mathbf{H}\mathbf{A}^{-1}\mathbf{H}^{\mathrm{T}}\right).$$

Apêndice A4

O critério *generalized cross-validation* (GCV)

O critério de seleção de modelo GCV é uma variação do critério *leave-one-out* (LOO) *cross-validation*. Portanto, primeiro o critério LOO será descrito aqui para posterior descrição do método GCV.

Se os dados disponíveis não são escassos, então o conjunto de medidas pode ser dividido em duas partes: uma parte para treinamento e outra parte para teste. Deste modo vários modelos diferentes, todos treinados sobre o mesmo conjunto de treinamento, podem ser comparados sobre um mesmo conjunto de teste. Esta é a forma básica do critério *cross-validation*.

O melhor método é particionar o conjunto de treinamento original de vários modos diferentes e calcular o escore sobre as diferentes partições. Uma variação extrema deste método é dividir os n_p padrões em um conjunto de treinamento de tamanho n_p-1 e um conjunto de teste de tamanho 1 e medir o erro quadrático do padrão de teste sobre os n_p possíveis modos de obter tal partição [ALLEN, 1974]. Este método é chamado de *leave-one-out cross-validation*. A vantagem é que todos os dados podem ser usados para o treinamento (nenhum dado precisa se separado para ser usado só no conjunto de teste). O atrativo de LOO é que para modelos lineares, tais como a rede RBF utilizada aqui, a predição da variância do erro pode ser calculada analiticamente.

Considere que existe apenas um neurônio de saída e que $\hat{\psi}(n)$ é a predição do modelo para o n-ésimo padrão do conjunto de treinamento depois de ter sido treinado por outros n_p-1 padrões. A predição da variância do erro para o critério LOO é dada por

$$\hat{\sigma}^2_{\mathrm{LOO}} = \frac{1}{n_p}\sum_{n=1}^{n_p}\left(\psi(n)-\hat{\psi}(n)\right)^2 . \qquad \text{(A4.1)}$$

O vetor de pesos ótimo para o conjunto de treinamento reduzido (isto é, sem o n-ésimo padrão) é dado por

$$\hat{\boldsymbol{\omega}}_n = \left(\mathbf{H}_n^{\mathrm{T}}\mathbf{H}_n + \boldsymbol{\Lambda}\right)^{-1}\mathbf{H}_n^{\mathrm{T}}\boldsymbol{\psi}_n = \mathbf{A}_n^{-1}\mathbf{H}_n^{\mathrm{T}}\boldsymbol{\psi}_n$$

na qual as matrizes $\mathbf{A}_n$ e $\mathbf{H}_n$ e o vetor $\boldsymbol{\psi}_n$ referem-se ao conjunto de treinamento reduzido. Aplicando-se o vetor de pesos ótimo calculado sobre o conjunto de treinamento reduzido no cálculo da predição da variância do erro para o *n*-ésimo padrão (conjunto de teste), tem-se

$$\begin{aligned} \psi(n) - \hat{\psi}(n) &= \psi(n) - \hat{\boldsymbol{\omega}}_n^{\mathrm{T}} \bar{\mathbf{h}}_n \\ \psi(n) - \hat{\psi}(n) &= \psi(n) - \boldsymbol{\psi}_n^{\mathrm{T}} \mathbf{H}_n \mathbf{A}_n^{-1} \bar{\mathbf{h}}_n \end{aligned} \qquad \text{(A4.2)}$$

na qual: $\bar{\mathbf{h}}_n = \begin{bmatrix} h_1(n) & h_2(n) & \dots & h_m(n) \end{bmatrix}^{\mathrm{T}}$ e,

$$\mathbf{H} = \begin{bmatrix} \bar{\mathbf{h}}_1 & \bar{\mathbf{h}}_2 & \dots & \bar{\mathbf{h}}_{n_p} \end{bmatrix}^{\mathrm{T}}.$$

A matriz $\mathbf{H}_n$ e o vetor $\boldsymbol{\psi}_n$ são obtidos removendo-se as *n*-ésimas linhas respectivamente de **H** e **y**. Consequentemente:

$$\mathbf{H}_n^{\mathrm{T}} \boldsymbol{\psi}_n = \mathbf{H}^{\mathrm{T}} \boldsymbol{\psi} - \psi(n)\ \bar{\mathbf{h}}_n \qquad \text{(A4.3)}$$

na qual $\bar{\mathbf{h}}_n^{\mathrm{T}}$ é a linha removida da matriz **H** e $\psi(n)$ é a componente tirada do vetor $\boldsymbol{\psi}$.

A seguir, a matriz $\mathbf{A}_n^{-1}$ será deduzida. Como não importa a ordem em que os padrões estão na matriz de projeto **H**, a linha removida será representada como estando na posição n_p. Assim

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_n \\ \bar{\mathbf{h}}_n^{\mathrm{T}} \end{bmatrix}.$$

Da equação da matriz de variância (Apêndice A3), tem-se

$$\begin{aligned} \mathbf{A} &= \mathbf{H}^{\mathrm{T}} \mathbf{H} \\ \mathbf{A} &= \begin{bmatrix} \mathbf{H}_n^{\mathrm{T}} & \bar{\mathbf{h}}_n \end{bmatrix} \begin{bmatrix} \mathbf{H}_n \\ \bar{\mathbf{h}}_n^{\mathrm{T}} \end{bmatrix} \\ \mathbf{A} &= \mathbf{A}_n + \bar{\mathbf{h}}_n\, \bar{\mathbf{h}}_n^{\mathrm{T}} \\ \mathbf{A}_n &= \mathbf{A} - \bar{\mathbf{h}}_n\, \bar{\mathbf{h}}_n^{\mathrm{T}} \end{aligned} \qquad \text{(A4.4)}$$

que é valida sem ou com penalização.

A seguir, será utilizado um lema para inversão de matrizes [HORN & JONHSON, 1985]:

Se a inversa de uma dada matriz é conhecida, e deseja-se saber como a inversa varia quando da adição de uma matriz de posto "pequeno", existe uma fórmula simples que, se a forma de ajuste da matriz é suficientemente simples, pode-se calcular a nova inversa de modo simples. Suponha uma matriz não-singular $\mathbf{A} \in \Re^{m \times m}$, cuja inversa $\mathbf{A}^{-1}$ é conhecida

$$\mathbf{B} = \mathbf{A} + \mathbf{XRY}$$

na qual **X** é uma matriz $n \times r$, **Y** é uma matriz $r \times n$ e **R** é uma matriz não-singular $r \times r$. Se **B** é não-singular, então

$$\mathbf{B}^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{X}\left(\mathbf{R}^{-1} + \mathbf{YA}^{-1}\mathbf{X}\right)^{-1}\mathbf{YA}^{-1}.$$

Se r é muito menor que n, então a matriz **R** e o termo entre parêntesis podem ser muito mais fáceis de se inverter do que a matriz **B**.

Aplicando o lema acima na Eq. (A4.4) com $\mathbf{X} = -\overline{\mathbf{h}}_n$, $\mathbf{Y} = \overline{\mathbf{h}}_n^{\mathrm{T}}$ e $\mathbf{R}$=1, tem-se

$$\mathbf{A}_n^{-1} = \mathbf{A}^{-1} + \frac{\mathbf{A}^{-1}\overline{\mathbf{h}}_n\,\overline{\mathbf{h}}_n^{\mathrm{T}}\mathbf{A}^{-1}}{1 - \overline{\mathbf{h}}_n^{\mathrm{T}}\mathbf{A}^{-1}\overline{\mathbf{h}}_n}. \qquad \text{(A4.5)}$$

Desta equação deduz-se que

$$\mathbf{A}_n^{-1}\overline{\mathbf{h}}_n = \frac{\mathbf{A}^{-1}\overline{\mathbf{h}}_n}{1 - \overline{\mathbf{h}}_n^{\mathrm{T}}\mathbf{A}^{-1}\overline{\mathbf{h}}_n}.$$

Substituindo esta expressão e a transposta da Eq. (A4.3) na Eq. (A4.2), tem-se

$$\psi(n) - \hat{\psi}(n) = \frac{\psi(n) - \boldsymbol{\psi}^{\mathrm{T}}\mathbf{H}\mathbf{A}^{-1}\overline{\mathbf{h}}_n}{1 - \overline{\mathbf{h}}_n^{\mathrm{T}}\mathbf{A}^{-1}\overline{\mathbf{h}}_n}.$$

O numerador desta relação é a *n*-ésima componente do vetor $\mathbf{P}\,\boldsymbol{\psi}$ e o denominador é a *n*-ésima componente da diagonal principal de **P**, na qual **P** é a matriz de projeção (Apêndice A3). Portanto, o vetor das n_p predições das variâncias dos erros é dado por

$$\begin{bmatrix} \psi(1)-\hat{\psi}(1) \\ \psi(2)-\hat{\psi}(2) \\ \vdots \\ \psi(n_p)-\hat{\psi}(n_p) \end{bmatrix} = \left(\text{diag}(\mathbf{P})\right)^{-1}\mathbf{P}\,\boldsymbol{\psi}\,.$$

A matriz diag (**P**) tem o mesmo tamanho e a mesma diagonal principal de **P**, mas os componentes fora da diagonal principal são iguais a zero. Finalmente, da equação acima e da Eq. (A4.1), tem-se a predição da variância do erro (ou erro LOO)

$$\hat{\sigma}^2_{\text{LOO}} = \frac{1}{n_p}\boldsymbol{\psi}^{\text{T}}\,\mathbf{P}\left(\text{diag}(\mathbf{P})\right)^{-2}\mathbf{P}\,\boldsymbol{\psi}\,.$$

No entanto, a matriz diag (**P**) torna a equação um tanto complicada de se manusear matematicamente. Da equação do LOO surge o critério GCV [GOLUB *et al*., 1979], que é mais conveniente matematicamente. A equação do critério GCV é conseguida substituindo-se a matriz diag (**P**) por $\left(\text{tr}(\mathbf{P})/n_p\right)\mathbf{I}_{n_p}$, na qual tr (**P**) denota o traço da matriz de projeção (o traço de uma matriz quadrada é a soma dos elementos da diagonal principal) e a matriz de identidade tem posto n_p. Assim, a predição da variância do erro GCV(ou erro GCV) fica:

$$\hat{\sigma}^2_{\text{GCV}} = \frac{n_p\,\boldsymbol{\psi}^{\text{T}}\mathbf{P}^2\boldsymbol{\psi}}{\left(\text{tr}(\mathbf{P})\right)^2}\,.$$

Apêndice A5

O valor ótimo para o parâmetro de regularização global

Como foi visto no Apêndice A4, o erro GCV é dado por

$$\hat{\sigma}^2_{\mathrm{GCV}} = \frac{n_p \, \boldsymbol{\psi}^{\mathrm{T}} \mathbf{P}^2 \boldsymbol{\psi}}{\left(\mathrm{tr}\left(\mathbf{P}\right)\right)^2}.$$

Para se achar o mínimo deste erro como uma função do parâmetro de regularização λ será preciso diferenciar a equação em relação a λ e, então, igualar o resultado a zero. Para isto, a matriz de projeção (Apêndice A3) precisa ser diferenciada. Para tal, a matriz de variância (Eq. 34) será diferenciada. Assim, assuma que a matriz **H** tenha a seguinte decomposição singular [HORN & JONHSON, 1985]

$$\mathbf{H} = \mathbf{U} \boldsymbol{\Sigma} \mathbf{V}^{\mathrm{T}}$$

na qual **U** e **V** são ortogonais e

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sqrt{\mu_1} & 0 & \cdots & 0 \\ 0 & \sqrt{\mu_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sqrt{\mu_m} \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

na qual $\left\{\sqrt{\mu_j}\right\}_{j=1}^{m}$ são os valores singulares de **H** e $\left\{\mu_j\right\}_{j=1}^{m}$ são os autovalores de $\mathbf{H}^{\mathrm{T}}\,\mathbf{H}$. Assumindo que **H** tenha mais linhas do que colunas ($n_p > m$), apesar que o mesmo resultado pode ser obtido para o caso oposto, segue que

$$\mathbf{H}^{\mathrm{T}}\mathbf{H} = \mathbf{V} \boldsymbol{\Sigma}^{\mathrm{T}} \boldsymbol{\Sigma} \mathbf{V}^{\mathrm{T}}.$$

Substituindo na equação da matriz de variância, tem-se

$$\mathbf{A}^{-1} = \left(\mathbf{H}^{\mathrm{T}}\mathbf{H} + \mathbf{\Lambda}\right)^{-1}$$

$$\mathbf{A}^{-1} = \left(\mathbf{V}\,\mathbf{\Sigma}^{\mathrm{T}}\mathbf{\Sigma}\,\mathbf{V}^{\mathrm{T}} + \lambda\,\mathbf{I}_m\right)^{-1}$$

$$\mathbf{A}^{-1} = \left(\mathbf{V}\,\mathbf{\Sigma}^{\mathrm{T}}\mathbf{\Sigma}\,\mathbf{V}^{\mathrm{T}} + \lambda\,\mathbf{V}\mathbf{V}^{\mathrm{T}}\right)^{-1}$$

$$\mathbf{A}^{-1} = \left(\mathbf{V}\left(\mathbf{\Sigma}^{\mathrm{T}}\mathbf{\Sigma} + \lambda\ \mathbf{I}_m\right)\mathbf{V}^{\mathrm{T}}\right)^{-1}.$$

Como $\mathbf{V}$ é ortogonal, então $\mathbf{V}^{-1} = \mathbf{V}^{\mathrm{T}}$ e $(\mathbf{V}^{\mathrm{T}})^{-1}$. Assim

$$\mathbf{A}^{-1} = \mathbf{V}\left(\mathbf{\Sigma}^{\mathrm{T}}\mathbf{\Sigma} + \lambda\ \mathbf{I}_m\right)^{-1}\mathbf{V}^{\mathrm{T}}. \qquad \text{(A5.1)}$$

Como a matriz entre parêntesis é diagonal, a inversa é fácil de se achar

$$\left(\mathbf{\Sigma}^{\mathrm{T}}\mathbf{\Sigma} + \lambda\ \mathbf{I}_m\right)^{-1} = \begin{bmatrix} \dfrac{1}{\lambda+\mu_1} & 0 & \cdots & 0 \\ 0 & \dfrac{1}{\lambda+\mu_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \dfrac{1}{\lambda+\mu_m} \end{bmatrix}.$$

Esta inversa é a única parte da equação (A5.1) que depende de λ. Derivando a inversa acima, tem-se

$$\frac{\partial}{\partial\lambda}\left(\mathbf{\Sigma}^{\mathrm{T}}\mathbf{\Sigma} + \lambda\ \mathbf{I}_m\right)^{-1} = \begin{bmatrix} \dfrac{-1}{(\lambda+\mu_1)^2} & 0 & \cdots & 0 \\ 0 & \dfrac{-1}{(\lambda+\mu_2)^2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \dfrac{-1}{(\lambda+\mu_m)^2} \end{bmatrix}$$

$$\frac{\partial}{\partial\lambda}\left(\boldsymbol{\Sigma}^{\mathrm{T}}\boldsymbol{\Sigma}+\lambda\ \mathbf{I}_m\right)^{-1}=-\left(\left(\boldsymbol{\Sigma}^{\mathrm{T}}\boldsymbol{\Sigma}+\lambda\ \mathbf{I}_m\right)^{-1}\right)^2$$

$$\frac{\partial}{\partial\lambda}\left(\boldsymbol{\Sigma}^{\mathrm{T}}\boldsymbol{\Sigma}+\lambda\ \mathbf{I}_m\right)^{-1}=-\left(\boldsymbol{\Sigma}^{\mathrm{T}}\boldsymbol{\Sigma}+\lambda\ \mathbf{I}_m\right)^{-2}.$$

Assim, a derivada em relação a λ da matriz de variância fica

$$\frac{\partial\,\mathbf{A}^{-1}}{\partial\lambda}=-\mathbf{V}\left(\boldsymbol{\Sigma}^{\mathrm{T}}\boldsymbol{\Sigma}+\lambda\ \mathbf{I}_m\right)^{-2}\mathbf{V}^{\mathrm{T}}$$

$$\frac{\partial\,\mathbf{A}^{-1}}{\partial\lambda}=-\mathbf{V}\left(\boldsymbol{\Sigma}^{\mathrm{T}}\boldsymbol{\Sigma}+\lambda\ \mathbf{I}_m\right)^{-1}\left(\boldsymbol{\Sigma}^{\mathrm{T}}\boldsymbol{\Sigma}+\lambda\ \mathbf{I}_m\right)^{-1}\mathbf{V}^{\mathrm{T}}$$

$$\frac{\partial\,\mathbf{A}^{-1}}{\partial\lambda}=-\mathbf{V}\left(\boldsymbol{\Sigma}^{\mathrm{T}}\boldsymbol{\Sigma}+\lambda\ \mathbf{I}_m\right)^{-1}\mathbf{V}^{\mathrm{T}}\mathbf{V}\left(\boldsymbol{\Sigma}^{\mathrm{T}}\boldsymbol{\Sigma}+\lambda\ \mathbf{I}_m\right)^{-1}\mathbf{V}^{\mathrm{T}}$$

$$\frac{\partial\,\mathbf{A}^{-1}}{\partial\lambda}=-\mathbf{A}^{-1}\mathbf{A}^{-1}$$

$$\frac{\partial\,\mathbf{A}^{-1}}{\partial\lambda}=-\mathbf{A}^{-2}.$$

A derivada da matriz de projeção em relação a λ é dada por

$$\frac{\partial\mathbf{P}}{\partial\lambda}=\frac{\partial}{\partial\lambda}\left(\mathbf{I}_{n_p}-\mathbf{H}\mathbf{A}^{-1}\mathbf{H}^{\mathrm{T}}\right)$$

$$\frac{\partial\mathbf{P}}{\partial\lambda}=-\mathbf{H}\frac{\partial\,\mathbf{A}^{-1}}{\partial\lambda}\mathbf{H}^{\mathrm{T}}$$

$$\frac{\partial\mathbf{P}}{\partial\lambda}=-\mathbf{H}\mathbf{A}^{-2}\mathbf{H}^{\mathrm{T}}.$$

Similarmente, pode-se diferenciar as outras componentes da equação do erro GCV

$$\frac{\partial}{\partial\lambda}\mathrm{tr}(\mathbf{P})=-\mathrm{tr}\left(\mathbf{H}\frac{\partial\,\mathbf{A}^{-1}}{\partial\lambda}\mathbf{H}^{\mathrm{T}}\right)$$

$$\frac{\partial}{\partial\lambda}\mathrm{tr}(\mathbf{P})=-\mathrm{tr}\left(\mathbf{H}\mathbf{A}^{-2}\mathbf{H}^{\mathrm{T}}\right)$$

$$\frac{\partial}{\partial\lambda}\mathrm{tr}(\mathbf{P}) = -\mathrm{tr}\left(\mathbf{A}^{-2}\mathbf{H}^{\mathrm{T}}\mathbf{H}\right)$$

$$\frac{\partial}{\partial\lambda}\mathrm{tr}(\mathbf{P}) = -\mathrm{tr}\left(\mathbf{A}^{-2}\left(\mathbf{A} - \lambda\,\mathbf{I}_m\right)\right)$$

$$\frac{\partial}{\partial\lambda}\mathrm{tr}(\mathbf{P}) = -\mathrm{tr}\left(\mathbf{A}^{-1} - \lambda\,\mathbf{A}^{-2}\right)$$

e,

$$\frac{\partial}{\partial\lambda}\left(\boldsymbol{\psi}^{\mathrm{T}}\,\mathbf{P}^2\,\boldsymbol{\psi}\right) = 2\,\boldsymbol{\psi}^{\mathrm{T}}\mathbf{P}\frac{\partial\mathbf{P}}{\partial\lambda}\boldsymbol{\psi}$$

$$\frac{\partial}{\partial\lambda}\left(\boldsymbol{\psi}^{\mathrm{T}}\,\mathbf{P}^2\,\boldsymbol{\psi}\right) = 2\,\boldsymbol{\psi}^{\mathrm{T}}\mathbf{P}\,\mathbf{H}\mathbf{A}^{-2}\mathbf{H}^{\mathrm{T}}\boldsymbol{\psi}$$

$$\frac{\partial}{\partial\lambda}\left(\boldsymbol{\psi}^{\mathrm{T}}\,\mathbf{P}^2\,\boldsymbol{\psi}\right) = 2\,\boldsymbol{\psi}^{\mathrm{T}}\left(\mathbf{I}_{n_p} - \mathbf{H}\mathbf{A}^{-1}\mathbf{H}^{\mathrm{T}}\right)\mathbf{H}\mathbf{A}^{-2}\mathbf{H}^{\mathrm{T}}\boldsymbol{\psi}$$

$$\frac{\partial}{\partial\lambda}\left(\boldsymbol{\psi}^{\mathrm{T}}\,\mathbf{P}^2\,\boldsymbol{\psi}\right) = 2\,\boldsymbol{\psi}^{\mathrm{T}}\left(\mathbf{H}\mathbf{A}^{-2}\mathbf{H}^{\mathrm{T}}\boldsymbol{\psi} - \mathbf{H}\mathbf{A}^{-1}\mathbf{H}^{\mathrm{T}}\mathbf{H}\mathbf{A}^{-2}\mathbf{H}^{\mathrm{T}}\boldsymbol{\psi}\right)$$

$$\frac{\partial}{\partial\lambda}\left(\boldsymbol{\psi}^{\mathrm{T}}\,\mathbf{P}^2\,\boldsymbol{\psi}\right) = 2\,\boldsymbol{\psi}^{\mathrm{T}}\left(\mathbf{H}\mathbf{A}^{-2}\mathbf{H}^{\mathrm{T}}\boldsymbol{\psi} - \mathbf{H}\mathbf{A}^{-1}\left(\mathbf{A} - \lambda\,\mathbf{I}_m\right)\mathbf{A}^{-2}\mathbf{H}^{\mathrm{T}}\boldsymbol{\psi}\right)$$

$$\frac{\partial}{\partial\lambda}\left(\boldsymbol{\psi}^{\mathrm{T}}\,\mathbf{P}^2\,\boldsymbol{\psi}\right) = 2\,\boldsymbol{\psi}^{\mathrm{T}}\left(\mathbf{H}\mathbf{A}^{-2}\mathbf{H}^{\mathrm{T}}\boldsymbol{\psi} - \mathbf{H}\mathbf{A}^{-1}\mathbf{A}\mathbf{A}^{-2}\mathbf{H}^{\mathrm{T}}\boldsymbol{\psi} + \mathbf{H}\mathbf{A}^{-1}\lambda\;\mathbf{A}^{-2}\mathbf{H}^{\mathrm{T}}\boldsymbol{\psi}\right)$$

$$\frac{\partial}{\partial\lambda}\left(\boldsymbol{\psi}^{\mathrm{T}}\,\mathbf{P}^2\,\boldsymbol{\psi}\right) = 2\lambda\;\boldsymbol{\psi}^{\mathrm{T}}\mathbf{H}\mathbf{A}^{-3}\,\mathbf{H}^{\mathrm{T}}\boldsymbol{\psi}\,.$$

Como o vetor de pesos ótimo é dado por

$$\hat{\boldsymbol{\omega}} = \mathbf{A}^{-1}\mathbf{H}^{\mathrm{T}}\,\boldsymbol{\psi}$$

então

$$\frac{\partial}{\partial\lambda}\left(\boldsymbol{\psi}^{\mathrm{T}}\,\mathbf{P}^2\,\boldsymbol{\psi}\right) = 2\lambda\;\hat{\boldsymbol{\omega}}^{\mathrm{T}}\mathbf{A}^{-1}\,\hat{\boldsymbol{\omega}}\,.$$

Utilizando as diferenciações acima para diferenciar a equação do erro GCV, tem-se

$$\frac{\partial \hat{\sigma}^2_{\mathrm{GCV}}}{\partial \lambda} = \frac{n_p}{\left(\mathrm{tr}\left(\mathbf{P}\right)\right)^2} \frac{\partial}{\partial \lambda}\left(\boldsymbol{\psi}^{\mathrm{T}}\mathbf{P}^2\boldsymbol{\psi}\right) - \frac{2\,n_p \boldsymbol{\psi}^{\mathrm{T}}\mathbf{P}^2\boldsymbol{\psi}}{\left(\mathrm{tr}\left(\mathbf{P}\right)\right)^3} \frac{\partial}{\partial \lambda}\mathrm{tr}\left(\mathbf{P}\right)$$

$$\frac{\partial \hat{\sigma}^2_{\mathrm{GCV}}}{\partial \lambda} = \frac{2n_p}{\left(\mathrm{tr}\left(\mathbf{P}\right)\right)^3}\left\{\lambda\, \hat{\boldsymbol{\omega}}^{\mathrm{T}}\mathbf{A}^{-1}\hat{\boldsymbol{\omega}}\,\mathrm{tr}\left(\mathbf{P}\right) - \boldsymbol{\psi}^{\mathrm{T}}\mathbf{P}^2\boldsymbol{\psi}\ \mathrm{tr}\left(\mathbf{A}^{-1} - \lambda\,\mathbf{A}^{-2}\right)\right\}.$$

Igualando a equação acima a zero pode se encontrar o parâmetro de regularização ótimo

$$\hat{\lambda}\ \hat{\boldsymbol{\omega}}^{\mathrm{T}}\mathbf{A}^{-1}\hat{\boldsymbol{\omega}}\,\mathrm{tr}\left(\mathbf{P}\right) = \boldsymbol{\psi}^{\mathrm{T}}\mathbf{P}^2\boldsymbol{\psi}\ \mathrm{tr}\left(\mathbf{A}^{-1} - \hat{\lambda}\,\mathbf{A}^{-2}\right).$$

Isolando o parâmetro de regularização ótimo, obtém-se a fórmula de reestimação para o erro GCV

$$\hat{\lambda} = \frac{\boldsymbol{\psi}^{\mathrm{T}}\mathbf{P}^2\boldsymbol{\psi}\,\mathrm{tr}\left(\mathbf{A}^{-1} - \hat{\lambda}\,\mathbf{A}^{-2}\right)}{\hat{\boldsymbol{\omega}}^{\mathrm{T}}\mathbf{A}^{-1}\hat{\boldsymbol{\omega}}\,\mathrm{tr}\left(\mathbf{P}\right)}.$$

Apêndice A6

Os valores ótimos para os parâmetros de regularização locais

Como está-se interessado na relação de dependência entre a matriz de projeção e os parâmetros locais, uma fórmula para remoção de uma unidade radial será usada, já que esta relação aparece de forma clara em sua equação. Assim, primeiro será deduzida a fórmula que relaciona a matriz de projeção antes e depois de uma unidade radial ser removida.

Considere que da matriz de projeto **H** é removida a coluna *j* (ou seja, a unidade radial *j* é removida), resultando em uma nova matriz $\mathbf{H}_{m-1}$. Assim

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_{m-1} & \mathbf{h}_j \end{bmatrix}$$

na qual

$$\mathbf{h}_j = \begin{bmatrix} h_j(1) \\ h_j(2) \\ \vdots \\ h_j\left(n_p\right) \end{bmatrix}.$$

Calculando a inversa da matriz de variância ($\mathbf{A}^{-1}$)

$$\mathbf{A} = \mathbf{H}^{\mathrm{T}}\mathbf{H} + \mathbf{\Lambda}$$

$$\mathbf{A} = \begin{bmatrix} \mathbf{H}_{m-1}^{\mathrm{T}} \\ \mathbf{h}_j^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} \mathbf{H}_{m-1} & \mathbf{h}_j \end{bmatrix} + \begin{bmatrix} \mathbf{\Lambda}_{m-1} & \mathbf{0} \\ \mathbf{0}^{\mathrm{T}} & \lambda_{\mathrm{j}} \end{bmatrix}$$

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{m-1} & \mathbf{H}_{m-1}^{\mathrm{T}}\mathbf{h}_j \\ \mathbf{h}_j^{\mathrm{T}}\mathbf{H}_{m-1} & \lambda_{\mathrm{j}} + \mathbf{h}_j^{\mathrm{T}}\mathbf{h}_j \end{bmatrix}. \qquad \text{(A6.1)}$$

A seguir será utilizado o seguinte lema para inversão de matrizes [HORN & JONHSON, 1985]:

A inversa de uma matriz particionada

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 & \mathbf{A}_2 \\ \mathbf{A}_3 & \mathbf{A}_4 \end{bmatrix}$$

pode ser expressa por

$$\mathbf{A}^{-1} = \begin{bmatrix} \left(\mathbf{A}_1 - \mathbf{A}_2\mathbf{A}_3^{-1}\mathbf{A}_4\right)^{-1} & \mathbf{A}_1^{-1}\mathbf{A}_2\left(\mathbf{A}_3\mathbf{A}_1^{-1}\mathbf{A}_2 - \mathbf{A}_4\right)^{-1} \\ \left(\mathbf{A}_3\mathbf{A}_1^{-1}\mathbf{A}_2 - \mathbf{A}_4\right)^{-1}\mathbf{A}_3\mathbf{A}_1^{-1} & \left(\mathbf{A}_4 - \mathbf{A}_3\mathbf{A}_1^{-1}\mathbf{A}_2\right)^{-1} \end{bmatrix}$$

assumindo que todas as inversas indicadas existam. Alternativamente, usando a fórmula de ajuste de posto pequeno (ver Apêndice A4) e definindo $\boldsymbol{\Delta} = \mathbf{A}_4 - \mathbf{A}_3\mathbf{A}_1^{-1}\mathbf{A}_2$, tem-se

$$\mathbf{A}^{-1} = \begin{bmatrix} \mathbf{A}_1^{-1} + \mathbf{A}_1^{-1}\mathbf{A}_2\boldsymbol{\Delta}^{-1}\mathbf{A}_3\mathbf{A}_1^{-1} & -\mathbf{A}_1^{-1}\mathbf{A}_2\boldsymbol{\Delta}^{-1} \\ -\boldsymbol{\Delta}^{-1}\mathbf{A}_3\mathbf{A}_1^{-1} & \boldsymbol{\Delta}^{-1} \end{bmatrix}.$$

Aplicando o lema acima na equação (A6.1), tem-se

$$\mathbf{A}^{-1} = \begin{bmatrix} \mathbf{A}_{m-1} & \mathbf{0} \\ \mathbf{0}^{\mathrm{T}} & 0 \end{bmatrix} + \frac{1}{\lambda_j + \mathbf{h}_j^{\mathrm{T}}\mathbf{P}_{m-1}\mathbf{h}_j} \begin{bmatrix} \mathbf{A}_{m-1}\mathbf{H}_{m-1}^{\mathrm{T}}\mathbf{h}_j \\ -1 \end{bmatrix} \begin{bmatrix} \mathbf{A}_{m-1}\mathbf{H}_{m-1}^{\mathrm{T}}\mathbf{h}_j \\ -1 \end{bmatrix}^{\mathrm{T}}. \qquad \text{(A6.2)}$$

A matriz de projeção (Apêndice A3) é dada por

$$\mathbf{P} = \mathbf{I}_{n_p} - \mathbf{H}\ \mathbf{A}^{-1}\mathbf{H}^{\mathrm{T}}.$$

Aplicando a equação (A6.2), tem-se

$$\mathbf{P} = \mathbf{P}_{m-1} - \frac{\mathbf{P}_{m-1}\mathbf{h}_j\mathbf{h}_j^{\mathrm{T}}\mathbf{P}_{m-1}}{\lambda_j + \mathbf{h}_j^{\mathrm{T}}\mathbf{P}_{m-1}\mathbf{h}_j}. \qquad \text{(A6.3)}$$

Pode se ver que na equação acima existe uma relação entre a matriz de projeção e os parâmetros de regularização. Fazendo $\Delta_j = \lambda_j + \mathbf{h}_j^{\mathrm{T}}\mathbf{P}_{m-1}\mathbf{h}_j$, tem-se

$$\mathbf{P} = \mathbf{P}_{m-1} - \frac{\mathbf{P}_{m-1}\mathbf{h}_j\mathbf{h}_j^{\mathrm{T}}\mathbf{P}_{m-1}}{\Delta_j}. \tag{A6.4}$$

O erro GCV é dado por (Apêndice A4)

$$\hat{\sigma}^2_{\mathrm{GCV}} = \frac{n_p\,\boldsymbol{\psi}^{\mathrm{T}}\mathbf{P}^2\boldsymbol{\psi}}{\left(\mathrm{tr}\left(\mathbf{P}\right)\right)^2}.$$

A função do erro GCV é um polinômio racional de ordem 2 em λ_j fácil de se analisar (isto é, de achar o mínimo para $\lambda_j \geq 0$). Substituindo a Eq. (A6.4) na equação acima

$$\hat{\sigma}^2_{\mathrm{GCV}}\left(\lambda_j\right) = \frac{n_p\left(a\Delta_j^2 - 2b\Delta_j + c\right)}{\left(\alpha\Delta_j - \beta\right)^2} \tag{A6.5}$$

na qual

$$a = \mathbf{y}^{\mathrm{T}}\mathbf{P}_{m-1}^2\boldsymbol{\psi}\ ,$$

$$b = \boldsymbol{\psi}^{\mathrm{T}}\mathbf{P}_{m-1}^2\mathbf{h}_j\boldsymbol{\psi}^{\mathrm{T}}\mathbf{P}_{m-1}\mathbf{h}_j\ ,$$

$$c = \mathbf{h}_j^{\mathrm{T}}\mathbf{P}_{m-1}^2\mathbf{h}_j\left(\boldsymbol{\psi}^{\mathrm{T}}\mathbf{P}_{m-1}\mathbf{h}_j\right)^2\ ,$$

$$\alpha = \mathrm{tr}\left(\mathbf{P}_{m-1}\right)\ \text{e,}$$

$$\beta = \mathbf{h}_j^{\mathrm{T}}\mathbf{P}_{m-1}^2\mathbf{h}_j\ .$$

Esta equação mostra como o erro GCV depende de λ_j quando todos os outros parâmetros de regularização são mantidos constantes. Esta dependência aparece através de um polinômio racional em Δ_j (ou λ_j). Como pode se ver, o polinômio tem um pólo em $\Delta_j = \beta / \alpha$, o qual nunca ocorre para valores positivos de λ_j visto que é sempre verdade que

$$\mathbf{h}_j^{\mathrm{T}}\mathbf{P}_{m-1}\mathbf{h}_j \geq \frac{\mathbf{h}_j^{\mathrm{T}}\mathbf{P}_{m-1}^2\mathbf{h}_j}{\mathrm{tr}\left(\mathbf{P}_{m-1}\right)}.$$

O valor mínimo do erro GCV pode ser alcançado diferenciando a Eq. (A6.5) com respeito a λ_j e igualando a zero a equação diferenciada. Fazendo a diferenciação, tem-se

$$\frac{\partial \hat{\sigma}^2_{\mathrm{GCV}}}{\partial \lambda_j} = \frac{\partial \hat{\sigma}^2_{\mathrm{GCV}}}{\partial \Delta_j}\frac{\partial \Delta_j}{\partial \lambda_j} = \frac{\partial \hat{\sigma}^2_{\mathrm{GCV}}}{\partial \Delta_j}$$

$$\frac{\partial \hat{\sigma}^2_{\mathrm{GCV}}}{\partial \Delta_j} = \frac{2n_p\left(a\Delta_j - b\right)}{\left(\alpha\Delta_j - \beta\right)^2} - \frac{2n_p\alpha\left(a\Delta_j^2 - 2b\Delta_j + c\right)}{\left(\alpha\Delta_j - \beta\right)^3}$$

$$\frac{\partial \hat{\sigma}^2_{\mathrm{GCV}}}{\partial \Delta_j} = \frac{2n_p\left(\left(b\alpha - a\beta\right)\Delta_j - \left(c\alpha - b\beta\right)\right)}{\left(\alpha\Delta_j - \beta\right)^3}.$$

Se $b\alpha = a\beta$ esta equação somente alcançará o valor zero se $\Delta_j = \pm\infty$ ($\lambda_j = \pm\infty$). Como o interesse é em valores de λ_j iguais ou maiores do que zero, então neste caso $\lambda_j = +\infty$. Se $b\alpha \neq a\beta$, igualando a equação a zero, tem-se

$$\hat{\Delta}_j = \frac{c\alpha - b\beta}{b\alpha - a\beta}$$

ou, ainda

$$\hat{\lambda}_j = \frac{c\alpha - b\beta}{b\alpha - a\beta} - \mathbf{h}_j^{\mathrm{T}}\mathbf{P}_{m-1}\mathbf{h}_j .$$

Este valor ótimo deve ser sempre maior que zero. Se o valor calculado for negativo, existem duas alternativas. Primeiro, se

$$\frac{\beta}{\alpha} > \frac{b}{a}$$

então o pólo (em $\Delta_j = \beta / \alpha$) ocorre à direita do mínimo global. Quando λ_j vai de -∞ até ∞, a derivada do erro GCV primeiro decresce até que seja alcançado o mínimo global (em $\lambda_j<0$). Daí a derivada cresce até que λ_j atinja o pólo e a partir deste ponto decresce até que λ_j chegue a ∞. Portanto, o erro mínimo para $\lambda_j \geq 0$ deve ser em $\hat{\lambda}_j = \infty$.

Entretanto, se

$$\frac{\beta}{\alpha} < \frac{b}{a}$$

então o pólo (em $\Delta_j = \beta / \alpha$) ocorre à esquerda do mínimo. Quando λ_j vai de -∞ até ∞, a derivada do erro GCV primeiro cresce até que seja alcançado o pólo. Daí a derivada decresce até que λ_j atinja o mínimo global (em λ_j<0) e a partir deste ponto cresce até que λ_j chegue a ∞. Portanto, o erro mínimo para $\lambda_j \geq 0$ deve ser em $\hat{\lambda}_j = 0$.

Assim, pode-se resumir para os valores ótimos do parâmetro de regularização da unidade radial *j* na equação (A6.4)

$$\hat{\lambda}_j = \begin{cases} \infty & \text{se} \quad a\beta = b\alpha \\ \infty & \text{se} \quad \mathbf{h}_j^{\mathrm{T}} \mathbf{P}_{m-1} \mathbf{h}_j > \dfrac{c\alpha - b\beta}{b\alpha - a\beta} \quad \text{e} \quad a\beta > b\alpha \\ 0 & \text{se} \quad \mathbf{h}_j^{\mathrm{T}} \mathbf{P}_{m-1} \mathbf{h}_j > \dfrac{c\alpha - b\beta}{b\alpha - a\beta} \quad \text{e} \quad a\beta < b\alpha \\ \dfrac{c\alpha - b\beta}{b\alpha - a\beta} - \mathbf{h}_j^{\mathrm{T}} \mathbf{P}_{m-1} \mathbf{h}_j & \text{se} \quad \mathbf{h}_j^{\mathrm{T}} \mathbf{P}_{m-1} \mathbf{h}_j \leq \dfrac{c\alpha - b\beta}{b\alpha - a\beta} \end{cases}$$

Em cada otimização individual é garantida a redução, ou no mínimo, não aumentar o erro GCV. Todos os *m* parâmetros são otimizados um após o outro e, se necessário, repetidamente até que o GCV pare de decrescer ou decresça em uma quantidade muito pequena. Naturalmente, a ordem com que os parâmetros são otimizados bem como os valores iniciais são importantes, podendo o erro atingir um mínimo local. O esforço computacional deste método é alto perto do GRR pois a matriz de projeção muda em cada otimização individual e precisa ser recalculada. Algum aumento de eficiência pode se obtido recalculando somente os valores usados nos cinco coeficientes do polinômio racional, tais como $\mathbf{P}\,\boldsymbol{\psi}$ e $\mathbf{P}\,\mathbf{H}$.

Apêndice A7

O método *forward selection*

Sabe-se que a soma dos erros quadráticos para o neurônio de saída *k* é dada por

$$C_k = \sum_{n=1}^{n_p} \left(\psi_k\ (n) - \hat{\psi}_k\ (n)\right)^2 .$$

Considerando apenas um neurônio de saída (q=1), passando para a notação vetorial a equação acima, tem-se

$$C = \left(\boldsymbol{\psi} - \hat{\boldsymbol{\psi}}\right)^{\mathrm{T}} \left(\boldsymbol{\psi} - \hat{\boldsymbol{\psi}}\right).$$

Substituindo a Eq. (A3.1), tem-se

$$\hat{C} = \left(\mathbf{P}\,\boldsymbol{\psi}\right)^{\mathrm{T}} \mathbf{P}\,\boldsymbol{\psi}$$

$$\hat{C} = \boldsymbol{\psi}^{\mathrm{T}} \mathbf{P}^{\mathrm{T}} \mathbf{P}\,\boldsymbol{\psi}$$

$$\hat{C} = \boldsymbol{\psi}^{\mathrm{T}} \mathbf{P}^{2}\,\boldsymbol{\psi} .$$

Considerando $\hat{C}_M$ como a soma dos erros quadráticos antes de a *n*-ésima unidade radial ser acrescentada e $\hat{C}_{M+1}$ a soma dos erros quadráticos depois de a *n*-ésima unidade radial ser acrescentada, tem-se

$$\hat{C}_M = \boldsymbol{\psi}^{\mathrm{T}} \mathbf{P}_M^2\,\boldsymbol{\psi}$$

$$\hat{C}_{M+1} = \boldsymbol{\psi}^{\mathrm{T}} \mathbf{P}_{M+1}^2\,\boldsymbol{\psi} .$$

Quando não existe regularização (como em FS) as matrizes **P** são matrizes de projeção verdadeiras e também são idempotentes (isto é, $\mathbf{P}^2 = \mathbf{P}$). Portanto

$$\hat{C}_M = \boldsymbol{\psi}^{\mathrm{T}} \mathbf{P}_M\,\boldsymbol{\psi}$$

$$\hat{C}_{M+1} = \boldsymbol{\psi}^{\mathrm{T}} \mathbf{P}_{M+1} \boldsymbol{\psi} .$$

Fazendo a diferença dos dois erros, tem-se

$$\hat{C}_M - \hat{C}_{M+1} = \boldsymbol{\psi}^{\mathrm{T}} \left(\mathbf{P}_M - \mathbf{P}_{M+1} \right) \boldsymbol{\psi} . \qquad \text{(A7.1)}$$

A Eq. (A6.3) relaciona as matrizes de projeção antes e depois de uma unidade radial ser retirada. Pode-se utilizar a mesma equação para relacionar as matrizes de projeção antes e depois que a *n*-ésima unidade radial é acrescentada. Assim (lembre-se que para FS $\lambda_j = 0$)

$$\mathbf{P}_{M+1} = \mathbf{P}_M - \frac{\mathbf{P}_M \mathbf{h}_n \mathbf{h}_n^{\mathrm{T}} \mathbf{P}_M}{\mathbf{h}_n^{\mathrm{T}} \mathbf{P}_M \mathbf{h}_n} .$$

Substituindo a equação acima na Eq. (A7.1), tem-se

$$\hat{C}_M - \hat{C}_{M+1} = \boldsymbol{\psi}^{\mathrm{T}} \frac{\mathbf{P}_M \mathbf{h}_n \mathbf{h}_n^{\mathrm{T}} \mathbf{P}_M}{\mathbf{h}_n^{\mathrm{T}} \mathbf{P}_M \mathbf{h}_n} \boldsymbol{\psi}$$

$$\hat{C}_M - \hat{C}_{M+1} = \frac{\left(\boldsymbol{\psi}^{\mathrm{T}} \mathbf{P}_M \mathbf{h}_n \right)^2}{\mathbf{h}_n^{\mathrm{T}} \mathbf{P}_M \mathbf{h}_n} .$$

Apêndice A8

Programas Utilizados

A seguir, encontra-se uma descrição dos programas construídos em MATLAB e em C.

Programas para o sistema de DDF do manipulador com 2 graus de liberdade

Programas em MATLAB:

- *r2g_falb*: a) simula o manipulador, gerando os padrões normalizados para treinamento (ou teste) do MLP; b) treinamento do MLP via retropropagação do erro; c) teste do mapeamento realizado pelo MLP.
- *residuo2*: gera os resíduos para uma ou várias trajetórias com os pesos do MLP gerados no programa anterior. Para cada condição de operação (falhas e operação normal), gera uma simulação diferente.
- *aux_koho*: utilizando os dados gerados no programa anterior, a) separa os padrões de entrada para uso do programa *rbf_koho.c*; b) unifica os centros gerados pelo programa *rbf_koho.c* em um único subconjunto.
- *rbf_2g*: utilizando os dados gerados no programa *residuo2.m*, treina (ou testa) a rede RBF por: FS, GRR, LRR e MAOK (nesta opção, os centros determinados pelo programa anterior são utilizados para cálculo da matriz de pesos).

Programa em C:

- *rbf_koho*: determina os centros da rede RBF para cada classe utilizando os dados gerados pelo programa *aux_koho.m*.

Programas para o sistema de DDF do Puma 560

Programas em MATLAB:

- *r6g_falb*: a) simula o manipulador, gerando os padrões normalizados para treinamento (ou teste) do MLP; b) teste do mapeamento realizado pelo MLP.
- *res_6g*: gera os resíduos para uma ou várias trajetórias com os pesos do MLP gerados no programa *r_falb.c*. Para cada condição de operação (falhas e operação normal), gera uma simulação diferente.

- *aux_koho*: utilizando os dados gerados no programa anterior, a) separa os padrões de entrada para uso do programa *rbf_koho.c*; b) unifica os centros gerados pelo programa *rbf_koho.c* em um único subconjunto.
- *rbf_6g*: utilizando os dados gerados no programa *residuo2.m*, treina (ou testa) a rede RBF por: FS, GRR, LRR e MAOK (nesta opção, os centros determinados pelo programa anterior são utilizados para cálculo da matriz de pesos).

Programa em C:

- *r_falb*: treinamento do MLP utilizando os dados gerados no programa *r6g_falb.m*.
- *rbf_koho*: determina os centros da rede RBF para cada classe utilizando os dados gerados pelo programa *aux_koho.m*.

Apêndice A9

Trajetórias utilizadas na simulação do manipulador com 2 graus de liberdade

Tabela A9.1. Trajetórias utilizadas no treinamento do MLP(manipulador com 2 elos).

	Posições iniciais		Posições finais	
Trajetória	Junta 1	Junta 2	Junta 1	Junta 2
1	π / 6	0	2 π / 3	π / 2
2	π / 6	π / 2	2 π / 3	0
3	7 π / 24	0	13 π / 24	π / 2
4	π / 6	3 π / 8	2 π / 3	π / 8
5	2 π / 3	3 π / 8	5 π / 12	0
6	13 π / 24	0	π / 6	π / 4
7	2 π / 3	π / 4	7 π / 24	π / 2
8	5 π / 12	π / 2	π / 6	π / 8
9	7 π / 24	0	π / 6	π / 8
10	13 π / 24	π / 2	2 π / 3	3 π / 8

Tabela A9.2. Trajetórias utilizadas no treinamento da rede RBF (manipulador com 2 elos).

	Posições iniciais		Posições finais	
Trajetória	Junta 1	Junta 2	Junta 1	Junta 2
1	π / 6	0	2 π / 3	π / 2
2	5 π / 12	π / 6	7 π / 24	π / 4
3	2 π / 3	3 π / 8	5 π / 12	0
4	13 π / 24	π / 4	5 π / 12	π / 8
5	2 π / 3	π / 8	13 π / 24	0
6	13 π / 24	0	π / 6	π / 4
7	1,83	1,09	2,08	1,23
8	7 π / 24	π / 4	5 π / 12	3 π / 8
9	π / 6	π / 2	2 π / 3	0

Tabela A9.3. Trajetórias utilizadas no teste do sistema de DDF (manipulador com 2 elos)

	Posições iniciais		Posições finais	
Trajetória	Junta 1	Junta 2	Junta 1	Junta 2
1	7 π / 24	0	13 π / 24	π / 2
2	7 π / 24	0	2 π / 3	π / 4
3	5 π / 12	π / 2	π / 6	π / 8
4	π / 6	3 π / 8	2 π / 3	π / 8
5	2 π / 3	π / 4	7 π / 24	π / 2
6	7 π / 24	0	π / 6	π / 8
7	13 π / 24	π / 2	2 π / 3	3 π / 8
8	7 π / 24	π / 2	π / 6	3 π / 8
9	2 π / 3	π / 8	5 π / 12	π / 2
10	5 π / 12	3 π / 8	13 π / 24	π / 4
11	5 π / 12	π / 8	7 π / 24	π / 4
12	13 π / 24	0	2 π / 3	π / 8
13	2 π / 3	π / 2	5 π / 12	0
14	5 π / 12	0	13 π / 24	π / 2
15	7 π / 24	3 π / 8	π / 6	0
16	13 π / 24	3 π / 8	7 π / 24	π / 4
17	π / 6	π / 4	2 π / 3	π / 2