UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA
ELÉTRICA E DE COMPUTAÇÃO

Utilização de redes neurais artificiais para detecção e diagnóstico de falhas

Diogo Leite Rebouças

Natal – RN
Junho / 2011

Utilização de redes neurais artificiais para detecção e diagnóstico de falhas

Diogo Leite Rebouças

Orientador: Prof. Dr. Fábio Meneghetti Ugulino de Araújo

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica e de Computação da UFRN (área de concentração: Automação e Sistemas) como parte dos requisitos para obtenção do título de Mestre em Ciências.

Natal, RN, Junho de 2011

Seção de Informação e Referência

Catalogação da Publicação na Fonte. UFRN / Biblioteca Central Zila Mamede

Rebouças, Diogo Leite.

Utilização de redes neurais artificiais para detecção e diagnóstico de falhas / Diogo Leite Rebouças. – Natal, RN, 2011.

76 f. : il.

Orientador: Fábio Meneghetti Ugulino de Araújo.

Dissertação (Mestrado) – Universidade Federal do Rio Grande do Norte. Centro de Tecnologia. Programa de Pós-Graduação em Engenharia Elétrica e de Computação.

1. Sistemas críticos – Dissertação. 2. Detecção de falhas – Dissertação. 3. Diagnóstico de falhas – Dissertação. 4. Redes neurais artificiais – Dissertação. I. Araújo, Fábio Meneghetti Ugulino de. II. Universidade Federal do Rio Grande do Norte. III. Título.

RN/UF/BCZM CDU 004.7

Utilização de redes neurais artificiais para detecção e diagnóstico de falhas

Diogo Leite Rebouças

Dissertação de Mestrado aprovada em 21 de junho de 2011 pela banca examinadora composta pelos seguintes membros:

Prof. Dr. Fábio Meneghetti Ugulino de Araújo (Orientador) ..... DCA/UFRN

Prof. Dr. André Laurindo Maitelli (Examinador Interno) ..... DCA/UFRN

Prof. Dr. Oscar Gabriel Filho (Examinador Externo) ..... UnP

“O único lugar em que o sucesso vem antes do trabalho é no dicionário.”

Sir Albert Einstein

Dedico este trabalho primeiramente a Deus, que me confortou nos momentos mais difíceis e que indicou o caminho certo ao longo de toda minha vida. Ao meu anjo da guarda, pela proteção e iluminação. À toda minha família e, em especial, aos meus pais, Marcondes e Aretusa, pela compreensão e dedicação ao longo desta e de outras jornadas. À minha namorada, Luciana, pelo apoio incondicional, paciência, incentivo e sugestões, durante a realização deste trabalho. Que não lhes falte saúde, paz e tranquilidade. Fica aqui minha gratidão eterna, pois sem vocês nada seria possível.

AGRADECIMENTOS

Aos meus pais, pelo incentivo aos estudos ao longo de toda minha vida. Sem vocês eu não seria nada.

A minha namorada, pela paciência, apoio e compreensão nas horas em que eu precisei me ausentar, em benefício deste e de outros tantos trabalhos durante esses anos. Sem você, talvez eu não tivesse conseguido.

Ao meu orientador, professor Fábio Meneghetti Ugulino de Araújo, sou grato pela orientação.

Aos professores André Laurindo Maitelli, Carlos Eduardo Trabuco Dórea e Oscar Gabriel Filho, pelas contribuições.

Aos colegas de pós-graduação, pelas críticas e sugestões.

À todos os professores e funcionários do Departamento de Engenharia de Computação e Automação e do Programa de Pós-Graduação em Engenharia Elétrica e de Computação, pelo aprendizado e pela ajuda.

Ao professor Adelardo Adelino Dantas de Medeiros, por ter desenvolvido o modelo de tese do Programa de Pós Graduação em Engenharia Elétrica e de Computação.

Aos integrantes do projeto Pró-Engenharias da UFRN, do ITA e da UFPA.

À CAPES, pelo apoio financeiro.

RESUMO

Em um processo real, todos os recursos utilizados, sejam físicos ou desenvolvidos em *software*, estão sujeitos a interrupções ou a comprometimentos operacionais. Contudo, nas situações em que operam os sistemas críticos, qualquer tipo de problema pode vir a trazer grandes consequências. Sabendo disso, este trabalho se propõe a desenvolver um sistema capaz de detectar a presença e indicar os tipos de falhas que venham a ocorrer em um determinado processo. Para implementação e testes da metodologia proposta, um sistema de tanques acoplados foi escolhido como modelo de estudo de caso. O sistema desenvolvido deverá gerar um conjunto de sinais que notifiquem o operador do processo e que possam vir a ser pós-processados, possibilitando que sejam feitas alterações nas estratégias ou nos parâmetros dos controladores. Em virtude dos riscos envolvidos com relação à queima dos sensores, atuadores e amplificadores existentes na planta real, o conjunto de dados das falhas será gerado computacionalmente e os resultados coletados a partir de simulações numéricas do modelo do processo, não havendo risco de dano aos equipamentos. O sistema será composto por estruturas que fazem uso de Redes Neurais Artificiais, treinadas em modo *offline* pelo *software* matemático Matlab®.

Palavras-chave: Sistemas Críticos, Detecção de Falhas, Diagnóstico de Falhas, Redes Neurais Artificiais.

ABSTRACT

In a real process, all used resources, whether physical or developed in software, are subject to interruptions or operational commitments. However, in situations in which operate critical systems, any kind of problem may bring big consequences. Knowing this, this paper aims to develop a system capable to detect the presence and indicate the types of failures that may occur in a process. For implementing and testing the proposed methodology, a coupled tank system was used as a study model case. The system should be developed to generate a set of signals that notify the process operator and that may be post-processed, enabling changes in control strategy or control parameters. Due to the damage risks involved with sensors, actuators and amplifiers of the real plant, the data set of the faults will be computationally generated and the results collected from numerical simulations of the process model. The system will be composed by structures with Artificial Neural Networks, trained in offline mode using Matlab®.

Keywords: Critical Systems, Fault Detection, Fault Diagnosis, Artificial Neural Network.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS E ABREVIATURAS

δ_i	*i*-ésima saída da rede neural
$\widehat{y_i}$	*i*-ésimo valor estimado pelo sistema
$\widehat{y}(t)$	Saída estimada pela rede neural
$\mu_{\mathrm{H_2O}}$	Coeficiente de viscosidade da água
$\mu_{\mathbf{ar}}$	Coeficiente de viscosidade do ar
$\omega_{j,l}$	Peso sináptico que parte do neurônio l e chega no neurônio j
ϕ	Vetor de entradas da rede neural
ϕ_l	*l*-ésima entrada da rede neural
ρ	*biases*
$\rho_{i,j}$	*biases* que partem da camada j e chegam na camada i
θ	Vetor de parâmetros ajustáveis da rede neural
a_i	Orifício de saída i
B_i	Bomba i
d	Atraso de transporte
F, f	Funções de ativação dos neurônios da rede neural
f_l	Função de ativação linear
f_s	Função de ativação sigmoidal

f_t	Função de ativação tangente hiperbólica
L_1	Nível do tanque superior
L_2	Nível do tanque inferior
m	Ordem de entrada
N	Número de amostras de validação
n	Ordem de saída
n_ϕ	Número de entradas da rede
n_h	Número de neurônios da camada oculta
T_1	Tanque superior
T_2	Tanque inferior
u	Entrada da planta
y	Saída da planta
y_i	i-ésimo valor real de saída da planta
ARMAX	*AutoRegressive Moving Average eXogenous*
ARX	*AutoRegressive eXogenous*
BIBO	*Bounded Input, Bounded Output*
CLP	*Controlador Lógico Programável*
DDF	*Detecção e Diagnóstico de Falha*
DIF	*Detecção e Isolamento de Falhas*
EMQ	*Erro Médio Quadrático*
FADG	*Falha do Atuador por Descalibramento de Ganho*
FADO	*Falha do Atuador por Descalibramento de Offset*
FAQ	*Falha do Atuador por Queima*
FASR	*Falha do Atuador por Sensibilidade à Ruído*
FAVK	*Falha do Atuador por Variação da Constante da Bomba (K_m)*
FIR	*Finite Impulse Response*
FSeDG	*Falha do Sensor por Descalibramento de Ganho*

FSeDO	*Falha do Sensor por Descalibramento de Offset*
FSeQ	*Falha do Sensor por Queima*
FSeSR	*Falha do Sensor por Sensibilidade à Ruído*
FSiEOS	*Falha do Sistema por Entupimento do Orifício de Saída*
FSiVrGMP	*Falha do Sistema por Variação do Ganho do Módulo de Potência*
FSiVrOS	*Falha do Sistema por Variação do Orifício de Saída*
FSiVzT	*Falha do Sistema por Vazamento do Tanque*
IHC	*Interface Humano-Computador*
ISA	*Industry Standard Architecture*
LMA	*Levenberg-Marquardt*
MIMO	*Multiple Input and Multiple Output*
NNARMAX	*Neural Network AutoRegressive Moving Average eXogenous inputs*
NNARX	*Neural Network AutoRegressive eXogenous inputs*
NNOE	*Neural Network Output Error*
NNSSIF	*Neural Network State Space Innovations Form*
OE	*Output Error*
PMC	*Perceptron de Múltiplas Camadas*
RK4	*Método de Runge-Kutta de 4ª ordem*
RNA	*Rede Neural Artificial*
Simddef	*Sistema modular para detecção e diagnóstico de falhas*
SISO	*Single Input and Single Output*
SSIF	*State Space Innovations Form*
XML	*eXtensible Markup Language*

CAPÍTULO 1

INTRODUÇÃO

Segundo Ribeiro (1999), o termo automação está relacionado com a substituição da mão-de-obra humana ou animal por uma máquina que realize função equivalente. Partindo desse princípio, pode-se dizer que a automação surge na sociedade em meados do século X, com os moinhos hidráulicos que produziam farinha. Tal mecanismo, capaz de substituir o trabalho de dez a vinte homens, fez com que a produção de alimentos passasse por uma fase de crescimento nunca antes observada.

Desde então, o homem tem direcionado seu conhecimento para o desenvolvimento de tecnologias que o auxiliem em suas atividades. Com a Revolução Industrial, a partir da segunda metade do século XVIII, o processo de transformação e desenvolvimento dessas tecnologias foi acelerado, de tal forma que o homem foi capaz de produzir uma máquina a vapor para movimentar equipamentos industriais e de fazer um martelo de 60 quilos dar 150 golpes por minuto [Goeking 2010].

Por outro lado, a utilização de sistemas de controle remete a tempos ainda mais antigos, por volta de 300 a.C. a 250 a.C., quando foram desenvolvidas as primeiras bóias flutuadoras, o relógio de água de Ktesíbios e uma lamparina a óleo que matinha o nível de óleo combustível constante [Mayr 1970, Mayr 1971, Mayr 1975].

Já na Europa Moderna, o primeiro sistema com realimentação a ser inventado foi o regulador de temperatura de Cornelis Drebbel, entre o final do século XVI e início do século XVII [Mayr 1975]. No final do século XVII e início do século XVIII, Dennis Papin inventou o primeiro regulador de pressão para caldeiras a vapor. Tal dispositivo tinha funcionalidade semelhante a uma válvula de segurança de uma panela de pressão [Dorf e Bishop 2009].

1.1 Aspectos históricos do controle automático

A junção dos conceitos de automação e sistemas de controle se dá no final do século XVIII, quando, em 1769, James Watt desenvolve o primeiro controlador automático com

realimentação usado em um processo industrial, conhecido como *regulador de esferas*.

Ainda em 1769, segundo Faccin (2004), Richard Arkwright, um inventor inglês considerado um dos precursores das técnicas de produção em série, acelerou o processo de industrialização ao desenvolver uma máquina de tecer movimentada pela força da água corrente.

Segundo o autor, foi o tear mecânico que impulsionou a Revolução Industrial na Europa, contribuindo diretamente para a mudança dos hábitos de trabalho e das relações sociais da Idade Contemporânea.

De acordo com Dorf e Bishop (2009), o século seguinte foi caracterizado pelo desenvolvimento de sistemas de controle automático através da intuição e da invenção. Esforços para aumentar a exatidão dos sistemas de controle levaram a atenuações mais lentas das oscilações transitórias e até mesmo a sistemas instáveis, tornando-se necessário o desenvolvimento da teoria de controle automático.

Por volta de 1868, J. C. Maxwell formulou a teoria matemática, através das equações diferenciais do regulador de esferas de James Watt, relacionando os efeitos dos parâmetros do sistema com o seu desempenho [Maxwell 1964]. Com o seu trabalho, Maxwell demonstrou a importância e a utilidade de modelos e métodos matemáticos para a compreensão dos processos industriais e da teoria de controle.

Nos anos seguintes, E. J. Routh (1877) e A. Hurwitz (1885) criaram seus critérios de estabilidade de maneira independente [Routh 1877, Bennett 1996]. Mais tarde, em 1896, A. M. Lyapunov também desenvolveu seu critério de estabilidade baseado em equações diferenciais não-lineares de movimento [Faccin 2004].

No início do século XX, por volta de 1907, foi desenvolvido o primeiro controlador de temperatura pneumático (do tipo liga-desliga), o qual foi instalado em uma unidade de pasteurização de leite em Nova York. Poucos anos depois, em 1914, Edgar H. Bristol, fundador da *Foxboro Instrument Company*, contribuiu significativamente para o desenvolvimento de sistemas de controle ao protocolar o pedido de patente de um amplificador denominado *flapper-nozzle amplifier*, capaz de prover a ação proporcional. Em seguida, por volta de 1920, Morris E. Leeds obteve uma patente de um controlador eletromecânico que provia a ação integral [Faccin 2004].

Em 1922, Nicholas Minorsky apresentou uma análise sobre a teoria de controle envolvida no controle de posição, formulando a lei de controle de três termos, hoje conhecida como controle PID. Entretanto, até 1930 seu trabalho não havia sido amplamente reconhecido [Bennett 1996].

Somente em 1935, Ralph Clarridge criou o controlador de três termos, ao utilizar um controlador que antecipava a variação no sinal de erro para solucionar um problema de oscilação de uma malha de controle de temperatura em uma indústria de celulose [Faccin 2004].

1.2 Introdução da automação na indústria

Durante o período entre guerras, a teoria e a prática de sistemas de controle nos Estados Unidos e na Europa Ocidental se desenvolveram de modo diferente do que na Rússia e no Leste Europeu. O principal incentivo para o uso da realimentação nos Esta-

dos Unidos foi o desenvolvimento do sistema telefônico e dos amplificadores eletrônicos com realimentação por Bode, Nyquist e Black nos Laboratórios Telefônicos Bell [Dorf e Bishop 2009].

Durante a segunda guerra mundial fez-se necessário projetar e construir pilotos automáticos para aeronaves, sistemas de posicionamento de armas, sistemas de controle de antenas de radares e outros sistemas militares baseados na abordagem do controle com realimentação. A complexidade e o desempenho esperados desses sistemas militares fizeram com que houvesse uma extensão das técnicas de controle disponíveis, além de promoverem o interesse em sistemas de controle e o desenvolvimento de novos critérios e métodos. O projeto de sistemas de controle, até então, era uma arte envolvendo a abordagem da tentativa e erro [Dorf e Bishop 2009].

A primeira tentativa de sucesso que não levava em consideração métodos de tentativa e erro foi proposta em Ziegler e Nichols (1942). Neste artigo, foram apresentados dois procedimentos para sintonia dos controladores através de regras simples baseadas nas características dinâmicas do processo.

Após a disseminação dos controladores PID na indústria nos anos seguintes à publicação do artigo de Ziegler e Nichols, aconteceu o grande marco da indústria eletrônica: a criação, em 1947, do transistor. A partir de então, começaram a ser desenvolvidos os primeiros "computadores industriais". Segundo Goeking (2010), embora o microprocessamento tenha sido comercializado apenas a partir da década de 60, foi nesse período que surgiram os primeiros robôs mecânicos que incorporavam os sistemas de microprocessamento e uniam tecnologias mecânicas e elétricas.

Considerando que até o final da década de 60, as empresas automobilísticas, apesar de produzirem em grande escala com rapidez e qualidade, não ofereciam muitas opções de personalização para os clientes (incluindo as cores dos acessórios), a General Motors, solicitou à empresa Allen-Bradley que confeccionasse um produto que conferisse versatilidade à produção. A empresa, que já produzia outros dispositivos elétricos, desenvolveu em 1968, o primeiro Controlador Lógico Programável (CLP), substituindo os antigos painéis de relés, permitindo fazer modificações rápidas no processo produtivo [Goeking 2010].

De acordo com Isermann (2006), a partir de 1975, com a disponibilidade de microcomputadores "baratos", o grau de automação da industria foi drasticamente elevado. Tal aumento ocorreu em paralelo com os avanços dos sensores, atuadores, das redes de comunicação e das Interfaces Humano-Computador (IHC).

Desde então, a demanda cada vez mais crescente pela necessidade de maior desempenho dos processos ou de maior qualidade dos produtos, bem como a independência do funcionamento da planta para com os operadores humanos, vem despertando interesse de diversos membros da comunidade científica.

1.3 A automação e supervisão de processos

A Fig. 1.1 mostra um esquema simplificado da automação de dois processos acoplados. O esquema foi dividido em três níveis: nível inferior, o qual contempla as estratégias de controle mais comuns na industria; nível intermediário, onde estarão os sistemas de

supervisão e o nível superior, no qual estão contidas as atividades de gerenciamento, coordenação e otimização dos processos.

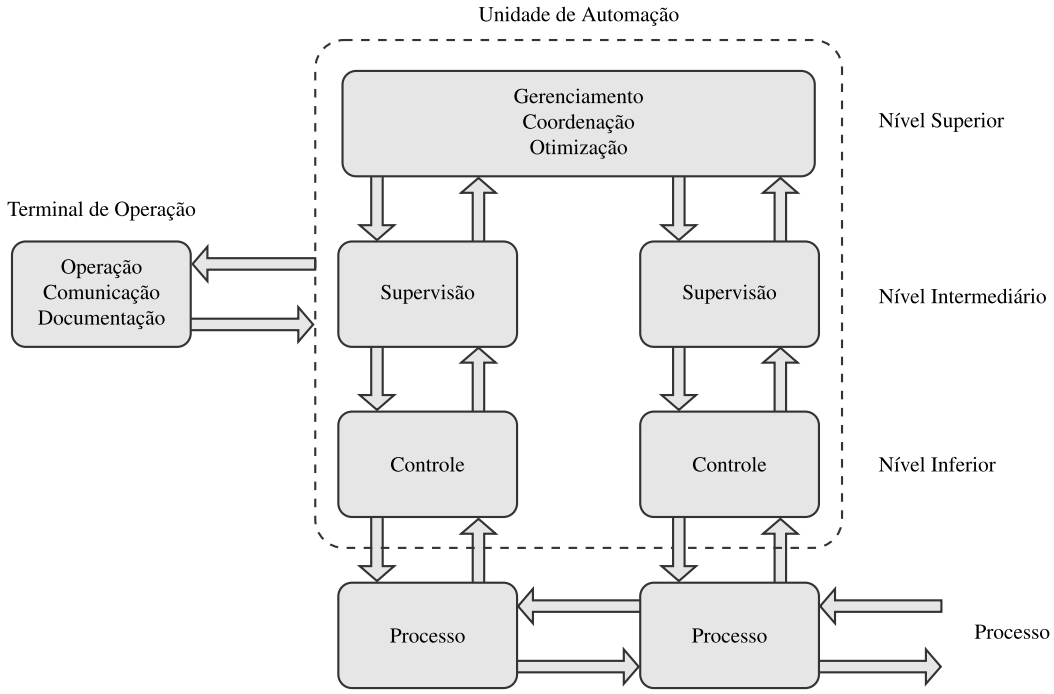


Figura 1.1: Diagrama esquemático da automação de dois processos acoplados.

Segundo com Isermann (2006), ao longo dos anos diversos estudos foram desenvolvidos sobre a teoria de controle com realimentação. Especialmente nos sistemas de controle que incluem os observadores de estado e estimadores de parâmetros, pode-se observar uma evolução significativa. Para o autor, com a evolução da teoria de controle e de processamento de sinais, diversos processos que antes possuíam um comportamento que dificultava a elaboração de estratégias de controle, agora podem ser controlados mais facilmente.

Entretanto, quanto mais eficiente forem as estratégias de controle no nível inferior, melhor deverão ser as estratégias de supervisão do nível intermediário, uma vez que os operadores são removidos do processo. Pode-se dizer que isso ocorre porque os operadores humanos não se limitam a controlar o processo mudando suas referências nos horários previstos, eles também realizam a atividade de supervisão, especialmente quando estão em contato direto com o processo. Logo, com a evolução das técnicas de controle no nível inferior, as estratégias de supervisão também deverão ser aperfeiçoadas.

No passado, o supervisionamento automático dos processos, em sua maioria, era composto por algum tipo de sistema que possuía a simples tarefa de verificar se uma determinada variável, tal como *força*, *velocidade*, *pressão*, *nível* ou *temperatura*, ultrapassava um certo limite ou limiar especificado para o processo. Caso isso viesse a ocorrer, um tipo de alarme disparava, notificando o operador do ocorrido, fazendo com que este agisse de maneira a corrigir o problema. Algumas vezes o problema podia ser corrigido também de maneira automática por algum subsistema de proteção. Tal procedimento, em

muitos dos casos, era suficiente para evitar que houvessem falhas ou danos graves ao processo. Por outro lado, as falhas ou os erros só eram detectados após um certo intervalo de tempo, o que impossibilitava a obtenção de um diagnóstico detalhado sobre o ocorrido [Isermann 2006].

Os desafios desse segmento estão, portanto, em se utilizar modelos matemáticos do processo, modelos de sinais, métodos de identificação e estimação e técnicas de inteligência artificial para se desenvolver um sistema capaz de detectar e diagnosticar falhas em um processo. Para se desenvolver esses tipos de sistemas, deve-se levar em consideração diversos aspectos, destacando-se: a detecção antecipada de pequenas falhas (abruptas ou incipientes); o diagnóstico de falhas nos sensores, atuadores e componentes de um processo; o supervisionamento de estados transientes; o reparo e a manutenção baseados no comportamento do processo; o rigoroso controle de qualidade em processos de fabricação; a detecção e diagnóstico de falhas remotas; o suporte para gerenciamento de falhas e o suporte para sistemas tolerantes a falhas e sistemas reconfiguráveis.

Considerando tais aspectos, pode-se dizer que as primeiras publicações na área de Detecção e Diagnóstico de Falhas (DDF) estão relacionadas com sistemas aeroespaciais [Beard 1971, Jones 1973, Willsky 1976, Clark 1978] e processos químicos [Himmelblau 1978]. Os primeiros conceitos discutidos podem ser classificados como abordagens de relação de paridade. Em tais documentos são demonstrados procedimentos de verificação da consistência da leitura dos instrumentos ou do balanço de massas. Outras abordagens, as quais utilizavam o erro residual do balanço de massas foram aplicadas, por exemplo, na verificação de vazamento em dutos [Billman e Isermann 1987]. A abordagem de relações de paridade foi também investigada em Gertler e Singer (1985).

Nessa mesma época, também foram desenvolvidos métodos baseados em observadores de estado ou filtros de Kalman [Beard 1971, Mehra e Peschon 1971, Jones 1973]. Em Clark (1978), por exemplo, um banco de observadores é utilizado para detectar falhas em sensores através de redundância de informações. Uma variação dos modelos de observadores de estado foi mostrada em Patton e Chen (1991), no qual se faz uso de uma estrutura de autovalores e autovetores para detecção de falhas.

Uma outra abordagem para esse tipo de sistema fez uso de estimadores de parâmetros, como pode ser observado em Baskiotis et al. (1979) para aplicações com turbinas de aeronaves, em Isermann (1982), Isermann (1984) e Isermann (1993) para processos de maneira geral, bombas e motores de corrente contínua, ou ainda em Filbert e Metzger (1982) e Filbert (1985) para aplicações envolvendo motores elétricos.

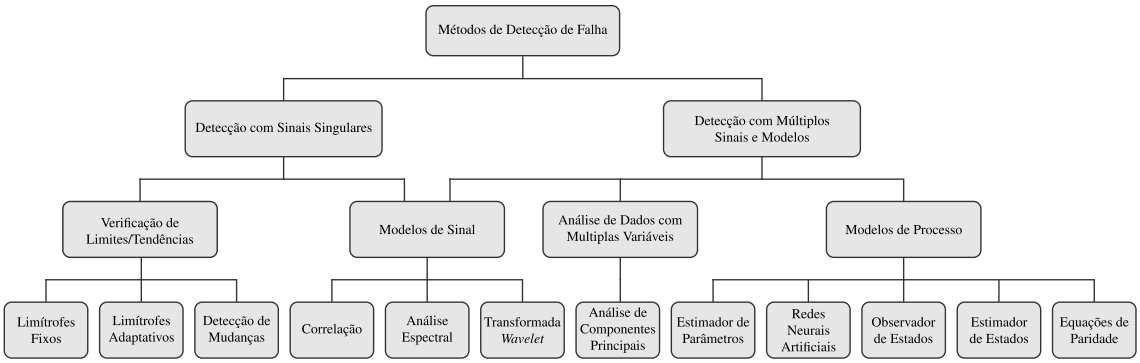
Mais recentemente, diversas contribuições foram feitas na área de DDF. Pode-se citar, por exemplo, as publicações relacionadas ao gerenciamento de falhas e supervisão de processos, tais como Russel et al. (2000), Patton et al. (2000), Higham e Perovic (2001), Simani et al. (2003), Ericson et al. (2005), Blanke et al. (2006), Erdenetsetseg e Ulemj (2007), Fesq (2009) e Hang e Lei (2009).

Uma outra contribuição importante é mostrada em VDI (2003), no qual é exposto um "Diagrama em V", que lista passos importantes para o desenvolvimento de sistemas de DDF. Isermann (2006) diz que apesar de seguir uma sequência lógica, muitas vezes os passos do diagrama são realizados de maneira paralela ou iterativa.

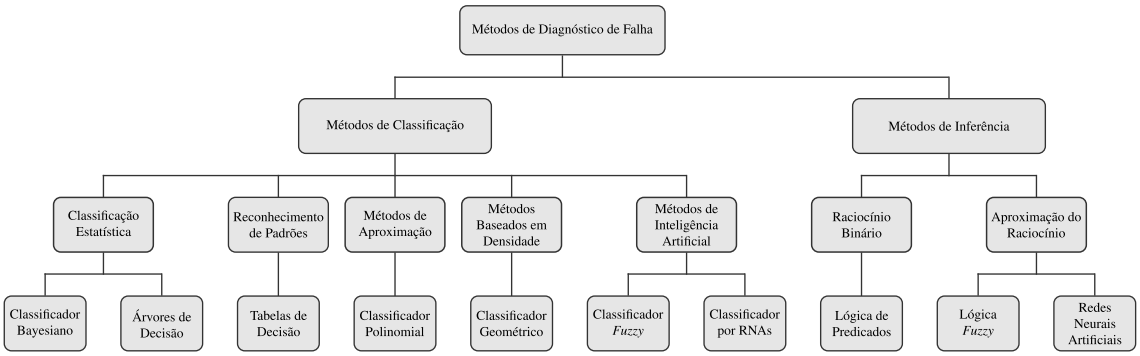
Segundo Zhang e Jiang (2008), apesar das diversas pesquisas existentes desde 1970,

os conceitos sistemáticos, os métodos de desenvolvimento e até mesmo as terminologias relacionadas à área de DDF ainda não estão bem estabelecidas, o que faz com que existam certos conflitos com a utilização correta dos termos em cada situação específica. Algumas das contribuições sobre o assunto podem ser observadas em Laprie (1996), Avižienis et al. (2000), Chiang et al. (2001), Wu (2004) e Isermann (2006).

Já com relação aos métodos e algoritmos de detecção e diagnóstico de falhas, pode-se encontrar em Isermann (2006), duas estruturas em árvore que classificam os diversos métodos e algoritmos existentes segundo determinados critérios. Tais estruturas podem ser observadas pelas Figs. 1.2(a) e 1.2(b).



(a) Métodos de Detecção

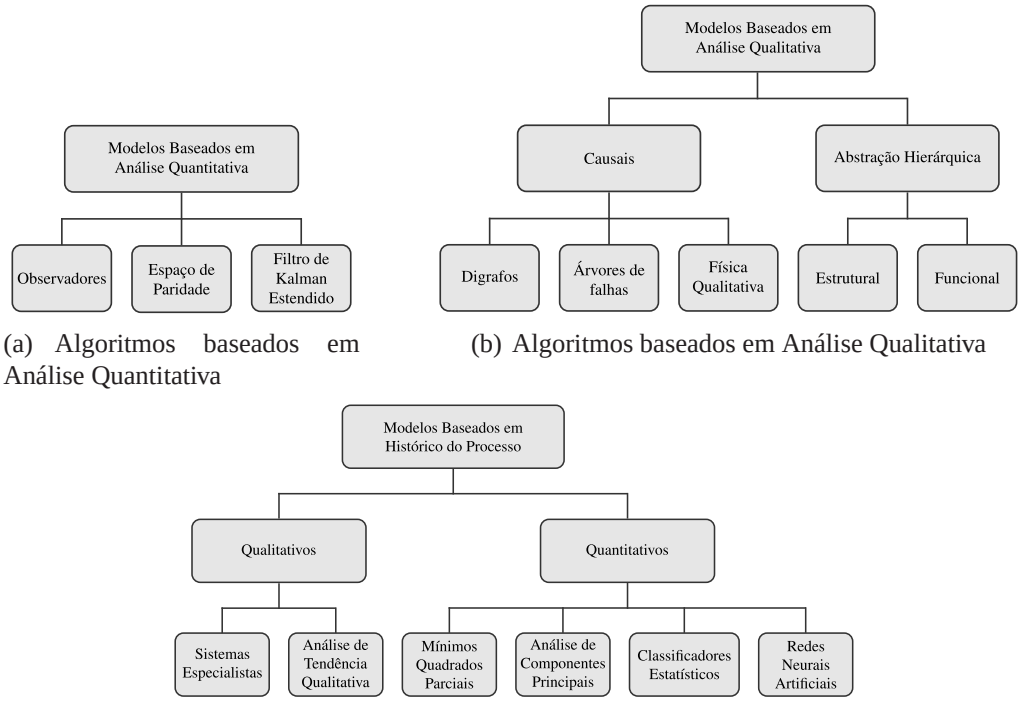


(b) Métodos de Diagnóstico

Figura 1.2: Métodos de detecção e diagnóstico de falhas.

De maneira complementar, Venkatasubramanian et al. (2003*a*) subdividem os algoritmos de diagnóstico de falhas em três classes, conforme Figs. 1.3(a) a 1.3(c). Cada uma das classes é abordada em detalhes nos três artigos que compõem uma revisão bibliográfica sobre o assunto [Venkatasubramanian et al. 2003*a*, Venkatasubramanian et al. 2003*b*, Venkatasubramanian et al. 2003*c*].

Tendo conhecido alguns dos temas mais abordados ao longo de todos esses anos, o trabalho a ser aqui apresentado se propõe a desenvolver um sistema que faça uso de Redes Neurais Artificiais (RNAs) para detectar e diagnosticar as falhas que venham a ocorrer em um determinado processo dinâmico. O sistema deverá ser capaz de detectar a presença das falhas, gerando sinais de alarme que notifiquem o operador e que possam vir a ser pós-processados.



(a) Algoritmos baseados em Análise Quantitativa

(b) Algoritmos baseados em Análise Qualitativa

(c) Algoritmos baseados em Histórico do Processo

Figura 1.3: Classificação dos algoritmos de diagnóstico de falhas.

Para que isso seja possível, o modelo de estudo de caso será identificado através de redes neurais, de tal modo que a partir do valor real mensurado e da inferência realizada pelo modelo identificado, seja gerado um resíduo que, em conjunto com outros valores, irão compor a entrada da estrutura neural de detecção e diagnóstico de falhas.

Considerando então os métodos de DDF que fazem uso de RNAs, pode-se destacar uma série de contribuições. Em Sreedhar et al. (1995), por exemplo, é elaborado um sistema que possui uma rede neural adaptativa capaz de detectar falhas em sistemas não-lineares. Vemuri et al. (1998), por sua vez, propõe um sistema de detecção de falhas inteligente para manipuladores robóticos. Já em Chang et al. (2003) é desenvolvido um sistema que faz uso de RNAs para detectar curto-circuitos. Em Talebi e Patel (2005) é mostrada uma estrutura capaz de detectar falhas em sistemas de controle de satélites. Jia-li et al. (2010) mostra ainda um sistema capaz de diagnosticar falhas em sistemas de câmbio automotivo.

Diversas outras contribuições foram feitas através da utilização de técnicas híbridas. Em Gao et al. (2000), por exemplo, é proposto um sistema que utiliza uma rede neural de Elman, com treinamento assistido por algoritmo genético para a detecção de falhas em unidades de sistemas de motor. Guo et al. (2005) combinam as propriedades das transformadas *wavelet* com RNAs para detecção de falhas em máquinas rotativas Já em Tian et al. (2007), é proposto um sistema Neuro-*Fuzzy* para detecção de falhas em oleodutos. Por fim, Khaled et al. (2010), mostra um sistema composto por um método que combina RNAs com análise de componentes principais, capaz de identificar e isolar falhas em processos de fabricação.

Assim sendo, os capítulos seguintes do presente texto serão divididos conforme descrição a seguir. O Capítulo 2 irá abordar os conceitos sobre as RNAs, mostrando suas arquiteturas e modelos de identificação que serão utilizados no trabalho. Em seguida, o Capítulo 3 irá tratar sobre as terminologias utilizadas e os métodos de detecção e diagnóstico de falhas. Já o Capítulo 4, irá conter a descrição detalhada do processo escolhido como estudo de caso e mostrará algumas possibilidades de como o sistema final poderá ser composto. Por fim, nos dois últimos capítulos, serão apresentados e discutidos os resultados obtidos para cada uma das estruturas idealizadas no Capítulo 4, encerrando com uma breve conclusão e mostrando as possibilidades de trabalhos futuros.

CAPÍTULO 2

REDES NEURAIS ARTIFICIAIS

Ao longo das últimas décadas, as pesquisas na área de Redes Neurais Artificiais (RNAs) evoluiram de maneira significativa, principalmente após a década de 80, com o avanço da tecnologia e o fracasso da escola simbolista na solução de determinados tipos de problemas [Braga et al. 2007].

No setor industrial a história não foi muito diferente. As RNAs vem sendo utilizadas em diversos trabalhos, seja de maneira isolada ou em sistemas híbridos, os quais combinam outras características de sistemas inteligentes, tais como técnicas *Fuzzy* ou Algoritmos Genéticos.

Neste capítulo serão mostrados, de maneira resumida, os conceitos que envolvem as RNAs, destacando algumas de suas propriedades e atributos. Por fim a atenção será voltada para o modelo que será utilizado para a identificação da dinâmica do processo e das falhas.

2.1 Conceitos fundamentais

Segundo Haykin (2000), as RNAs são estruturas paralelas, maciçamente distribuídas, constituídas por unidades simples de processamento conhecida como neurônios. Essas estruturas se assemelham ao cérebro humano devido a sua capacidade de "adquirir conhecimento" a partir do ambiente em que se encontra. Esse aprendizado ocorre através de um ajuste das forças de conexões, ou pesos sinápticos, que existe entre os neurônios. São essas conexões que armazenam os conhecimentos adquiridos pela rede.

Dentre as diversas aplicações das RNAs, podem ser citados exemplos para a classificação de padrões, filtragem de sinais, análise de imagens, identificação e controle de sistemas dinâmicos. De acordo com Haykin (2000), algumas das justificativas para a utilização dessas estruturas são: sua característica intrínseca de não-linearidade, sua capacidade de generalização e adaptabilidade, a tolerância à falhas e a facilidade para realizar o mapeamento de relações entrada-saída.

Devido à grande complexidade existente em muitos problemas físicos reais, desenvolver um modelo matemático que represente adequadamente a dinâmica do processo é uma tarefa praticamente impossível. Para Rebouças (2009), as RNAs, através de seu processo de aprendizagem e de sua capacidade de aproximação universal, conseguem representar a função correspondente à dinâmica do sistema com relativa simplicidade.

Apesar de se conhecer as equações que regem a dinâmica do processo escolhido como estudo de caso deste trabalho, conforme será mostrado no Capítulo 4, justifica-se desde já a utilização de RNAs para identificação do modelo em virtude da necessidade de simulação das falhas. Maiores detalhes serão esclarecidos ao longo do texto do referido capítulo.

2.2 Arquitetura das redes

Dentre as diversas arquiteturas de redes neurais existentes, tais como as redes de funções de base radial, as redes de Kohonen, máquinas de vetor de suporte, dentre outras, a arquitetura escolhida para este trabalho foi a das redes Perceptron de Múltiplas Camadas (PMC), treinadas com o algoritmo Levenberg-Marquardt (LMA) disponível no *software* matemático Matlab®. A Fig. 2.1 representa um modelo esquemático de uma rede PMC.

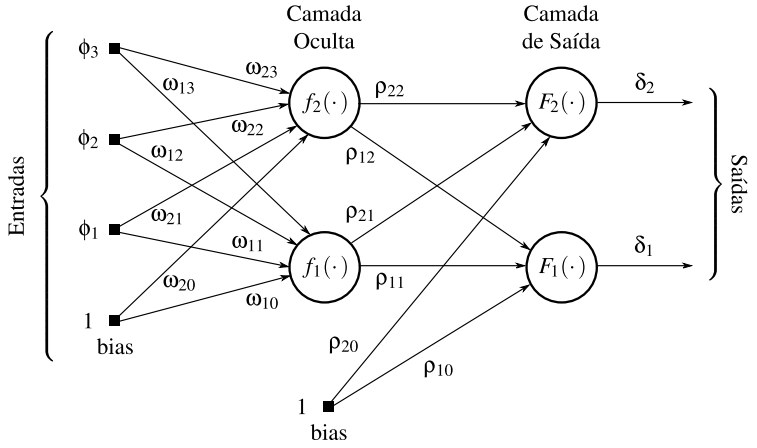


Figura 2.1: Diagrama esquemático de uma rede PMC.

A opção pela estrutura de PMC se dá devido a sua simplicidade e capacidade de aplicação em diversas áreas. Já a opção pelo treinamento com o algoritmo LMA pode ser justificada por se tratar de um método Quase-Newton que acelera o processo de convergência da rede, uma vez que leva em consideração termos de ordens mais elevadas que o algoritmo *backpropagation*.

Ademais, pode-se comentar que, apesar de ser possível realizar o treinamento das redes através de uma linguagem de programação convencional, implementar tais soluções, tão difundidas em *softwares* mundialmente reconhecidos como o Matlab® e amplamente aceitas no meio acadêmico, está fora do escopo deste trabalho.

Segundo Nørgaard et al. (2000), uma rede PMC básica possui seus neurônios dispostos em camadas, recebendo como entrada as saídas dos neurônios da camada imediata-

mente anterior ou, no caso da primeira camada, as entradas da rede. Por possuir essa configuração essas redes são conhecidas como redes *feedforward*.

Como mostrado na Fig. 2.1, a segunda camada é conhecida como camada de saída, pois produz as saídas da rede neral. A primeira camada é conhecida como camada oculta ou intermediária por estar "escondida" entre as entradas da rede (ϕ_1, ϕ_2 e ϕ_3) e a camada de saída. Em Cybenko (1989) foi demonstrado que qualquer função contínua pode ser aproximada por uma rede neural PMC que possua uma camada oculta com funções de ativação sigmoidal ou tangente hiperbólica.

A Eq. 2.1 expressa matematicamente o funcionamento de uma rede neural PMC de duas camadas, como na Fig. 2.1, em que δ_i representa a i-ésima saída da rede.

$$\delta_i(t) = g_i[\phi,\theta] = F_i\left[\sum_{j=1}^{n_h} \rho_{i,j} f_j\left(\sum_{l=1}^{n_\phi} \omega_{j,l}\phi_l + \omega_{j,0}\right) + \rho_{i,0}\right] \tag{2.1}$$

O vetor de parâmetros θ contém os pesos sinápticos e *biases* ($\omega_{j,l}$, $\rho_{i,j}$). O número de neurônios da camada oculta e o número de entradas da rede são, respectivamente, n_h e n_ϕ, enquanto que F_i e f_j são as funções de ativação dos neurônios das camadas de saída e oculta, respectivamente. As funções de ativação mais comumente utilizadas são mostradas nas equações 2.2, 2.3 e 2.4.

$$f_l(x) = x \tag{2.2}$$

$$f_s(x) = \frac{1}{1+e^{-x}} \tag{2.3}$$

$$f_t(x) = \tanh(x) = \frac{1-e^{-x}}{1+e^{-x}} \tag{2.4}$$

2.3 Identificação através de modelo neural

Segundo Rebouças (2009), as estruturas de modelo baseadas em redes neurais, apropriadas para a identificação de sistemas não-lineares, são generalizações de modelos lineares. Para Lucena (2005), essas estruturas são caracterizadas por seu vetor de regressão, que nada mais é do que um vetor que contém as variáveis utilizadas para estimar a saída do sistema. Dependendo da escolha do vetor de regressão, diferentes estruturas de modelo neural podem surgir. Estruturas FIR (*Finite Impulse Response*), ARX (*AutoRegressive eXternal input*), ARMAX (*AutoRegressive Moving Average eXternal input*), OE (*Output Error*) e SSIF (*State Space Innovations Form*) são algumas das estruturas lineares mais conhecidas. Se o vetor de regressão for selecionado para modelos ARX, a estrutura do modelo neural será chamada NNARX (*Neural Network ARX*). Do mesmo modo, existirão também modelos NNFIR, NNARMAX, NNOE e NNSSIF.

Neste trabalho foi utilizado um modelo de rede baseado no NNARX descrito em Nørgaard et al. (2000). A Fig. 2.2 representa um esquema simplificado do modelo adotado. A presença de regressores no modelo, relaciona a saída da rede com seus valores passados

de entrada e saída. A utilização desses regressores é de fundamental importância para a identificação de sistemas.

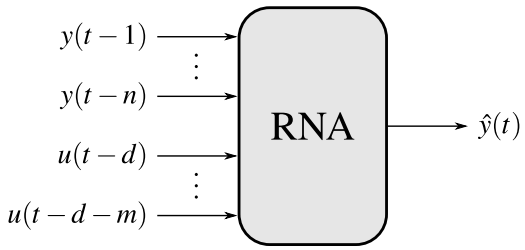


Figura 2.2: Esquema de uma rede neural com estrutura NNARX.

A expressão matemática que descreve o modelo não-linear pode ser descrita conforme Eq. 2.5.

$$\hat{y}(t) = g\left(y(t-1),\ldots,y(t-n),u(t-d),\ldots,u(t-d-m)\right) \tag{2.5}$$

Nessa equação $\hat{y}$ representa a saída estimada, d o atraso de transporte, n a ordem da saída, m a ordem de entrada da planta, $g(\cdot)$ uma função não linear mapeada pela rede neural, y a saída da planta e u a entrada da planta.

A estimativa gerada pela estrutura NNARX é sempre estável, uma vez que representa relações puramente algébricas entre a estimativa e as medições passadas de entradas e saídas do processos, não existindo a realimentação da saída estimada.

2.3.1 Determinação da ordem do modelo

Em sistemas de identificação *blackbox*[1], como o utilizado neste trabalho, é de fundamental importância que a ordem do modelo seja determinada adequadamente. Uma escolha inadequada poderá fazer com que a dinâmica do processo não seja assimilada pela rede neural, fazendo com que as estimativas geradas possuam erro médio consideravelmente alto. Segundo Arruda et al. (2003), existe uma ordem de modelo ótima que permite obter o menor erro entre o modelo estimado e o sistema real.

A ordem de entrada (m) e de saída (n) de um sistema desse tipo pode ser obtida a partir da realização de testes ou simulações com o processo. Por facilidade de representação, considerar-se-á a ordem de entrada igual a ordem de saída ($m = n$). No capítulo 5 poderão ser vistos alguns resultados dos testes que foram realizados para a determinação da ordem das redes de identificação e de detecção e diagnóstico de falhas.

2.3.2 Seleção do modelo

Existem diversas estruturas de modelagem, cada uma com suas vantagens e desvantagens. Segundo Nørgaard et al. (2000), o problema da seleção das estruturas pode ser

[1]Também conhecida como identificação em caixa-preta, a identificação *blackbox* pressupõe que não se dispõe de nenhuma informação sobre a estrutura interna ou sobre o funcionamento do sistema. Nesse tipo de abordagem, faz-se uso de uma análise da relação dos valores de saída obtidos a partir dos estímulos fornecidos nas entradas do sistema.

dividido em duas partes. A primeira delas é a parte da seleção da "família" de estruturas de modelagem. Dependendo do sistema, as estruturas podem ser: modelos lineares, redes PMC, redes de função de base radial, *wavelets* etc. Já a segunda, é a parte da seleção do subconjunto da família escolhida.

A escolha por uma estrutura do tipo NNARX se deu pelas diversas vantagens em se utilizar RNAs para estimar valores e por se tratar de um modelo que não faz uso de realimentação da saída estimada, o que permite dizer que este é estável no sentido BIBO[2] (*Bounded Input, Bounded Output*). Para Nørgaard et al. (2000), devido a estas características, o modelo NNARX é um dos mais utilizados quando o sistema a ser modelado é determinístico ou o nível de ruído não é significativo.

De maneira resumida, após a seleção da estrutura, segue-se uma sequência de etapas até que o modelo possa representar adequadamente o sistema, de acordo com algum critério específico, como por exemplo, o erro médio quadrático (EMQ) de estimativa, que é calculado a partir da Eq. 2.6 e utilizado no algoritmo de treinamento da RNA para determinar o ponto de parada (fim do treinamento).

$$\text{EMQ} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\frac{y_i - \widehat{y_i}}{y_i}\right)^2} \tag{2.6}$$

Nessa equação, N representa o número de amostras de validação, y_i o valor real e $\widehat{y_i}$ o valor estimado.

Tendo conhecido as características das redes neurais que serão utilizadas no desenvolvimento deste trabalho, o capítulo seguinte trará os principais conceitos e terminologias relacionadas ao tema de detecção e diagnóstico de falhas.

[2]Um sistema é dito BIBO estável quando, para qualquer entrada limitada, obtém-se uma saída também limitada.

CAPÍTULO 3

DETECÇÃO E DIAGNÓSTICO DE FALHAS

Com a demanda cada vez mais crescente com relação a eficiência, a qualidade dos produtos e a integração dos processos no setor industrial, aliada aos altos custos envolvidos e as mais diversas necessidades de segurança, torna-se evidente a importância dos sistemas de supervisão e dos sistemas de Detecção e Diagnóstico de Falhas (DDF) [Isermann 2006].

Atualmente, devido ao nível de complexidade envolvido em um processo industrial, os sistemas de supervisão e proteção exigem as mais avançadas técnicas oferecidas pela engenharia moderna e, ainda assim, demandam um esforço contínuo por parte dos pesquisadores e engenheiros para que sejam desenvolvidas novas tecnologias que venham a suprir suas necessidades [Silva 2008].

Considerando tais aspectos, este capítulo mostrará os conceitos relacionados ao desenvolvimento de um sistema de DDF que fará uso de RNAs. Inicialmente, serão mostrados os principais conceitos e terminologias utilizadas na área, destacando alguns dos métodos para a realização de DDF.

3.1 Conceitos e terminologias

Segundo Kaâniche et al. (2002), os sistemas computacionais podem ser caracterizados por cinco propriedades fundamentais: funcionalidade, usabilidade, desempenho, custo e *dependabilidade*. Para Laprie et al. (1992), o termo *dependabilidade*, que nada mais é do que a tradução literal do termo inglês *dependability*, está relacionado com a capacidade de um sistema prestar um serviço que possa ser, justificadamente, confiável. O serviço prestado se refere ao comportamento do sistema percebido por seus usuários, os quais também serão sistemas (máquinas físicas ou seres humanos), que interagem com o anterior.

De acordo com Laprie et al. (1994), dependendo das aplicações envolvidas, o termo

dependabilidade pode ainda ser visto a partir de suas diferentes, mas, complementares, propriedades:

- **Disponibilidade:** Prontidão para o uso;
- **Confiabilidade:** Continuidade do serviço;
- **Proteção (*safety*):** Não ocorrência de consequências catastróficas para o meio;
- **Confidencialidade:** Não ocorrência da divulgação não-autorizada da informação;
- **Integridade:** Não ocorrência de alterações indevidas da informação;
- **Manutenção:** Aptidão para sofrer reparos e evolução.

A associação da integridade com a disponibilidade e a confidencialidade leva à **Segurança (*security*)**.

Apesar do termo dependabilidade ser utilizado na maioria das vezes dessa forma, o dicionário Michaelis, traduz tal termo como *confiabilidade* ou *garantia de funcionamento*. Assim sendo, daqui para frente, a palavra *confiável*, ou suas variações linguísticas, também será utilizada para se referir ao termo *dependable*. Se for necessário fazer referência a propriedade da confiabilidade de um sistema, tal intenção será claramente especificada pelo texto.

3.2 Sistemática da dependabilidade

Avižienis et al. (2000) e Kaâniche et al. (2002) mostram que para se desenvolver um Sistema Computacional Confiável (SCC, do inglês *Dependable Computing System* – DCS), faz-se uso de diferentes técnicas, tais como as técnicas de: **prevenção**, **tolerância**, **supressão** e **previsão** de falhas.

O estudo sobre a *prevenção* de falhas envolve a seleção de metologias de projeto e de tecnologias adequadas para a escolha e a aplicação dos componentes no sistema, ou seja, busca não introduzir ou evitar que as falhas aconteçam. A *tolerância* a falhas está mais relacionada com a continuidade da prestação do serviço quando uma falha vier a ocorrer. Algumas das aplicações mais conhecidas são: mascaramento de falhas, redundância de dispositivos/sistemas e a detecção/recuperação de falhas. A *supressão* das falhas procura minimizar as consequências relativas à presença de uma falha através de mecanismos de verificação, diagnóstico e correção. A *previsão* de falhas está relacionada com a estimativa que pode ser feita sobre a presença, a criação e a consequência que as falhas venham a causar no sistema. Para isso, tal técnica faz uso de critérios avaliativos, técnicas de injeção de falhas e testes de resistência.

Além dessas técnicas, é válido também fazer referência aos termos **avaria** (*failure*), **erro** (*error*) e **falha** (*fault*), conceitualmente abordados de diferentes maneiras por vários autores. Os conceitos relativos a essa nomenclatura serão melhor explicados na seção 3.3.

A partir dessas informações, Avižienis et al. (2000) classificam os termos acima relacionados em três grupos, denominados atributos (propriedades), ameaças e meios pelos quais a dependabilidade é atingida, conforme pode ser observado na Fig. 3.1.

O primeiro grupo, cujos elementos foram brevemente explicados na seção 3.1, permite expressar as propriedades esperadas e analisar a qualidade de um sistema confiável. Já o

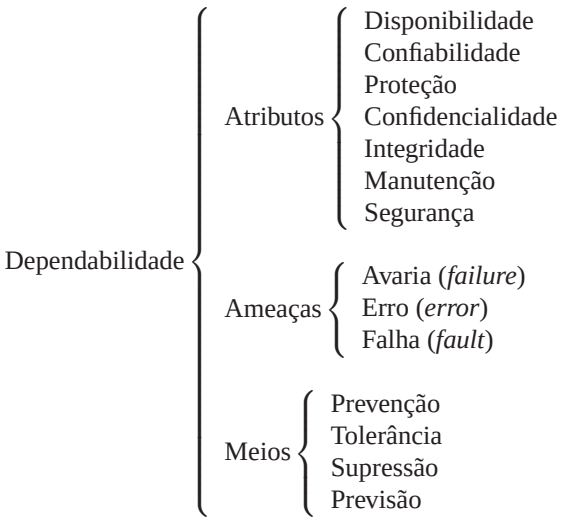


Figura 3.1: Classificação sistemática da dependabilidade.

segundo grupo traz os termos utilizados para expressar características indesejadas – mas, em princípio, não inesperadas – que causam ou fazem com que um sistema passe a ser não-confiável. Por fim, o terceiro grupo exibe os meios ou as técnicas pelas quais torna-se possível oferecer um serviço confiável.

De posse dessas informações e sabendo que muitas vezes os autores utilizam traduções diferentes para *failure*, *error* e *fault*, a seção seguinte tentará esclarecer o significado de cada um desses termos.

3.3 Avarias, erros e falhas

Em um processo real, todos os recursos utilizados, sejam físicos ou implementados em *software*, estão sujeitos a interrupções ou a comprometimentos operacionais.

Em sistemas críticos, tais como as aeronaves ou as usinas nucleares, essas situações fazem com que pequenos "deslizes operacionais" venham a trazer grandes e indesejáveis consequências. Como exemplo dessas consequências, pode-se destacar o acidente do *Airbus 320* da TAM em 2007, que não conseguiu parar ao aterrissar no aeroporto de Congonhas, São Paulo, no qual 199 pessoas vieram a falecer (12 em solo e 187 no avião) e o desastre da usina nuclear de Chernobil, que liberou cerca de 400 vezes mais contaminação do que a bomba nuclear que foi lançada em Hiroshima.

Esses exemplos fazem com que se reflita sobre como os sistemas de controle podem evoluir para evitar que catástrofes ainda maiores venham a ocorrer. Ou seja, como as propriedades de um sistema de controle confiável poderão ser mantidas mesmo na presença de avarias, erros e falhas no processo.

Segundo Pereira Filho (2002), apesar do termo falha ser utilizado, em muitos dos casos, como um termo vago, abrangendo também o significado de avarias e erros, existem certas diferenças conceituais que devem ser observadas.

O termo **avaria** (*failure*) deve ser utilizado para indicar que houve um desvio do comportamento no sistema, o que o torna incapaz de fornecer o serviço para o qual foi designado. Um **erro** (*error*), entretanto, está relacionado com estado do sistema e pode levar a uma avaria. De maneira resumida, se há um erro no estado do sistema, então existe uma sequência de ações que podem ser executadas e que levarão as avarias, a não ser que medidas corretivas venham a ser tomadas. Por fim, mas não menos importante, o termo **falha** (*fault*) é a causa de um erro e está associado à noção de defeitos. Normalmente, diz-se que o termo falha pode ser definido como sendo um defeito que possui o potencial de gerar erros [Pereira Filho 2002, Weber 2002].

Conforme dito anteriormente, alguns autores costumam traduzir os termos *failure*, *error* e *fault* de maneira diferente. O termo *failure*, por exemplo, também é traduzido como falha e o termo *fault* como falta. Entretanto, como é mais comum se referir a sistemas de controle tolerantes a "falhas" ao invés de sistemas de controle tolerantes a "faltas", preferiu-se aqui traduzir o termo *failure* como avaria e o termo *fault* como falha, até mesmo porque as avarias (*failures*) não podem ser toleradas.

Tendo conhecido o significado de cada um desses termos, pode-se mostrar que existe uma certa relação entre eles. Considerando que os sistemas reais são normalmente compostos por subsistemas, é comum se observar que uma falha leva a um erro, que por sua vez pode levar a uma avaria, podendo vir a gerar novas falhas e dar início a uma reação em cadeia, tal como mostra a Fig. 3.2. Contudo, nem sempre uma falha conduz a um erro, assim como nem sempre um erro conduz a uma avaria, mas todos os erros resultam de falhas e todas as avarias resultam de erros.

$$\ldots \implies \text{Falha} \longrightarrow \text{Erro} \longrightarrow \text{Avaria} \implies \text{Falha} \longrightarrow \ldots$$

Figura 3.2: Reação em cadeia das falhas, erros e avarias.

Uma outra maneira de visualizar as relações existentes entre cada um desses termos pode ser observada na Fig. 3.3.

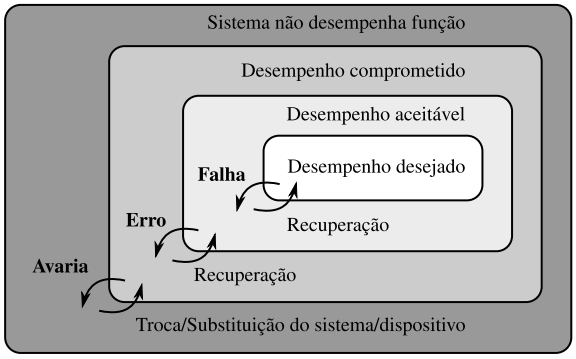


Figura 3.3: Mapa de conceitos relacionados a avarias, erros e falhas em um sistema.

Weber (2002) afirma que as falhas são inevitáveis, uma vez que os componentes físicos do sistema envelhecem e estão sempre sujeitos as interferências externas, ambientais

ou humanas. Mostra ainda que os *softwares*, assim como os sistemas físicos, também são vítimas de falhas, pois estão a mercê da alta complexidade dos processos e da fragilidade humana em trabalhar com grande volume de detalhes de especificação e operação.

3.3.1 Tipos de falhas

De acordo com Silva (2008), as falhas em um processo industrial podem ser classificadas em relação a vários aspectos. Em se tratando da classificação quanto ao tempo, as falhas podem ser abruptas, incipientes ou intermitentes, tal como mostra a Fig. 3.4.

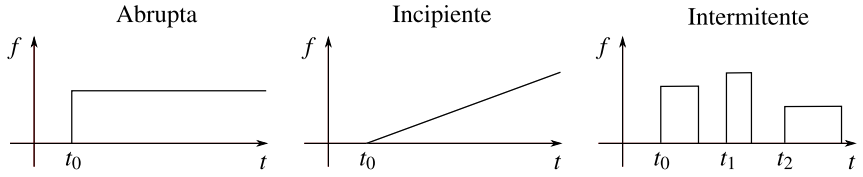


Figura 3.4: Características temporais das falhas quanto a sua persistência.

As *falhas abruptas* surgem repentinamente, podendo ser decorrentes de imprevistos ou até mesmo de acidentes. Essas falhas mudam o comportamento do processo rapidamente, exigindo contra-ações velozes e eficazes que possam minimizar as consequências do ocorrido.

As *falhas incipientes* iniciam a partir de pequenos desvios comportamentais do sistema, podendo ser mascaradas pelos controladores. Muitas vezes essas falhas acabam passando despercebidas pelos operadores ou até mesmo pelos sistemas de detecção e diagnóstico de falhas.

As *falhas intermitentes* são aquelas que ocorrem durante um certo período de tempo e, em seguida, desaparecem, voltando a aparecer após um novo intervalo. Podem ser causadas por alguma perturbação periódica ou por alguma situação que se repita ciclicamente.

Com relação a localização das falhas, estas podem ocorrer nos sensores, nos atuadores ou na estrutura do sistema [Silva 2008]. As *falhas nos sensores* são observadas através de variações específicas nas medições, as quais podem ser descaracterizadas como variações válidas do sistema. As *falhas nos atuadores* podem ser vistas como qualquer mau funcionamento do equipamento que atua no sistema. As *falhas na estrutura*, ou *falhas estruturais*, ocorrem quando alguma alteração do sistema muda, de alguma forma, a relação original de entrada e saída do processo ou quando ocorre algum problema com algum dos dispositivos, desde que não sejam sensores ou atuadores, como por exemplo, os transmissores de sinal. Considerando então um processo genérico, pode-se citar como exemplos de falhas aquelas exibidas pela Tab. 3.1.

Maiores detalhes sobre as propriedades e a classificação das falhas podem ser encontradas em Laprie et al. (1992), Laprie et al. (1994), Weber (2002) e Isermann (2006).

Tabela 3.1: Exemplos de falhas para um processo genérico.

Sensores	Atuadores	Estrutura
Erro de leitura	Erro de escrita	Erro de transmissão
Descalibramento	Erro de leitura	Perda de comunicação
Sensibilidade a ruído	Sensibilidade a ruído	Sensibilidade a ruído (transmissor)
Queima	Queima	Queima (transmissor)
-	Atraso de transporte	Atraso de propagação de sinais

3.4 Detecção e diagnóstico de falhas

Considerando que os processos industrias estão se tornando cada vez mais integrados e complexos, a ocorrência de falhas passa a ser mais um fator de complicação. As falhas que antes poderiam ser detectadas facilmente por medições diretas de determinada variável dependem agora de um conjunto de variáveis que atuam simultaneamente no sistema.

Além disso, se uma falha é detectada em um determinado ponto de um sistema, a causa do problema pode estar próxima ou distante, dependendo do grau de complexidade envolvido. Como exemplo, uma simples falha em uma aeronave pode causar indicações de falha em diversos subsistemas de segurança [Vachtsevanos et al. 2006]. Dessa maneira, processar as informações disponíveis pode vir a contribuir de maneira significativa para o processo de detecção e diagnóstico de falhas.

Com o intuito de garantir o sucesso das operações planejadas e reconhecer as anomalias comportamentais dos processos, muitos sistemas de supervisão e monitoramento estão sendo desenvolvidos. Para Chiang et al. (2001), dentre diversas outras funções, esses sistemas podem auxiliar nas atividades de detectar, diagnosticar e suprimir falhas, garantindo que as operações do processo satisfaçam as especificações de desempenho.

De maneira complementar, destaca-se que a informação a ser disponibilizada por um sistema de monitoramento não deve, tão somente, informar ao operador do sistema sobre o que está acontecendo, mas auxiliá-lo a tomar medidas corretivas com o intuito de sanar o problema. Como resultado disso, o tempo em que o sistema estará inoperante será reduzido, a proteção do sistema aumentará e os custos relacionados diminuirão.

Para Chiang et al. (2001), existem quatro fases envolvidas no monitoramento dos processos: **detecção**, **isolamento**, **diagnóstico** e **recuperação** de falhas, conforme mostrado pela Fig. 3.5.

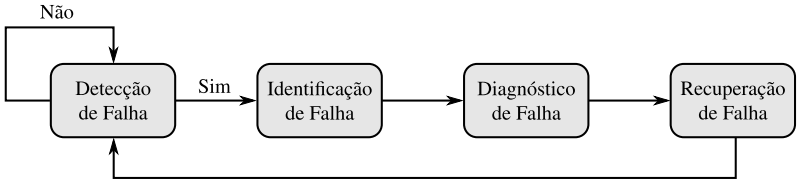


Figura 3.5: Fases envolvidas no monitoramento de processos.

A *detecção da falha* se resume a determinar se ocorreu ou não uma falha. Quando realizada de maneira antecipada, a detecção de falhas pode fornecer informações valiosas

com relação aos problemas que estão por acontecer. Assim, através de ações apropriadas, pode-se evitar grandes perturbações no processo.

A *identificação da falha* tem por objetivo identificar as variáveis mais importantes para que se realize um diagnóstico apropriado. Para isso, essa fase procura concentrar as atenções do operador nos subsistemas mais pertinentes à falha, de tal modo que o efeito desta possa ser eliminado de maneira mais eficiente.

A fase de *diagnóstico da falha* determina que falha ocorreu. De acordo com Isermann (2004), essa fase deverá indicar o maior número de detalhes possíveis a respeito da falha, tais como a intensidade, a localização e o momento em que a falha foi detectada. Essa fase é considerada essencial para uma contra-ação ou eliminação da falha.

Por fim, a fase de *recuperação da falha*, também conhecida como fase de *intervenção*, tem por objetivo remover o efeito da falha para com o sistema, dando início a um novo ciclo de detecção e diagnóstico.

Apesar de estarem explicitamente colocados como uma sequência de ações, nem sempre todas as fases são estritamente necessárias [Chiang et al. 2001]. Os sistemas automatizados fazem com que a mudança de uma fase para outra seja transparente para operador, exibindo apenas as informações cruciais para que se possa tomar as medidas cabíveis.

Venkatasubramanian et al. (2003*a*) mostram que a automatização das formas de detecção e diagnóstico é o primeiro passo a ser tomado para a elaboração de um sistema de gestão de eventos anormais. Afirmam também que ao longo de vários anos muitas soluções vem sendo desenvolvidas e que um grande número de técnicas já foram utilizadas. Para isso, citam como exemplos as técnicas que utilizam árvores de falhas, digrafos, abordagens analíticas, sistemas especialistas e RNAs.

Do ponto de vista de modelagem, existem métodos que exigem grande precisão com relação ao modelo de processo, outros que exigem modelos semi-quantitativos ou ainda modelos qualitativos. Por outro lado, existem métodos que não necessitam de nenhuma informação do modelo, os quais utilizam somente as informações de histórico do processo [Venkatasubramanian et al. 2003*a*].

Conforme mostrado no Capítulo 1, a classificação dos métodos de DDF pode variar de autor para autor [Venkatasubramanian et al. 2003*a*, Angeli e Chatzinikolaou 2004, Zhang e Jiang 2008, Isermann 2006]. A abordagem aqui escolhida irá considerar a classificação adotada em Isermann (2006), até então considerada mais intuitiva.

3.5 Métodos de detecção

Nas seções seguintes será feita uma breve análise dos métodos de detecção de falhas abordados em Isermann (2006). Vale salientar que existem na literatura vários outros métodos de detecção de falhas. Deseja-se aqui apenas trazer alguns exemplos.

3.5.1 Detecção de falhas com verificação de limites

Considerado como um dos métodos mais simples e intuitivos, a detecção de falhas com verificação de limites baseia-se na medição direta de uma determinada variável e

a comparação de seu valor absoluto ou de sua tendência com um limítrofe previamente especificado.

A verificação de valores absolutos baseia-se na utilização de dois limiares ou limítrofes fixos, denominados Y_{min} e Y_{max}. O funcionamento normal do sistema consiste em verificar se a variável Y está ou não contida no intervalo:

$$Y_{min} < Y < Y_{max} \tag{3.1}$$

Essa abordagem considera que o processo está funcionando normalmente quando a variável monitorada encontra-se dentro de uma zona de tolerância. Quando a variável monitorada excede um dos limiares estabelecidos, deduz-se que haverá uma falha em algum ponto do processo. Por mais simples que pareça, este método é aplicado na maioria dos sistemas de automação

A outra abordagem utiliza a tendência da variável medida para detectar as falhas. Como exemplo, um dos métodos mais conhecidos faz uso da primeira derivada da variável ($\dot{Y} = dY(t)/dt$) e testa se a variação observada está ou não dentro de uma faixa previamente estabelecida:

$$\dot{Y}_{min} < \dot{Y} < \dot{Y}_{max} \tag{3.2}$$

Exemplos dessas duas abordagens podem ser observados na Fig. 3.6.

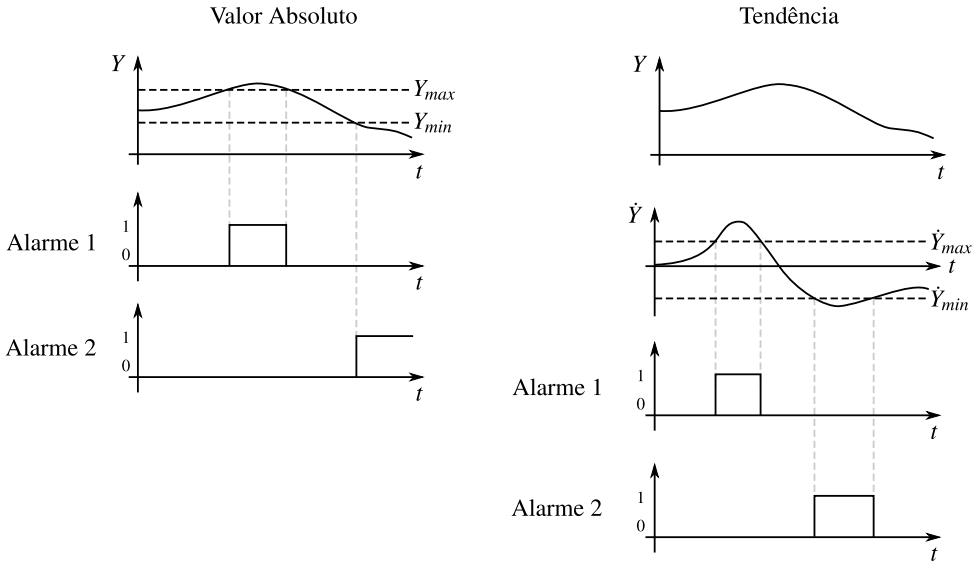


Figura 3.6: Exemplo de detecção com verificação de limites.

3.5.2 Detecção de falhas com modelos de sinais

Diversos sinais medidos em um processo apresentam oscilações de natureza harmônica ou estocástica. Se as mudanças nesse sinal estiverem relacionadas com as falhas nos sensores, atuadores ou estruturais, pode-se detectar essas falhas através do uso de modelos de sinais.

Considera-se, portanto, que as características do sinal medido, tais como amplitude, fase, espectro de frequências, dentre outras, são calculadas a partir de modelos matemáticos do sinal e comparadas com as características observadas durante o funcionamento normal. As diferenças comportamentais geradas pela comparação, denominadas de *sintomas analíticos*, são utilizadas para realizar a detecção das falhas.

Na Fig. 3.7 é exibido um diagrama esquemático que resume os princípios da detecção de falhas com modelos de sinais.

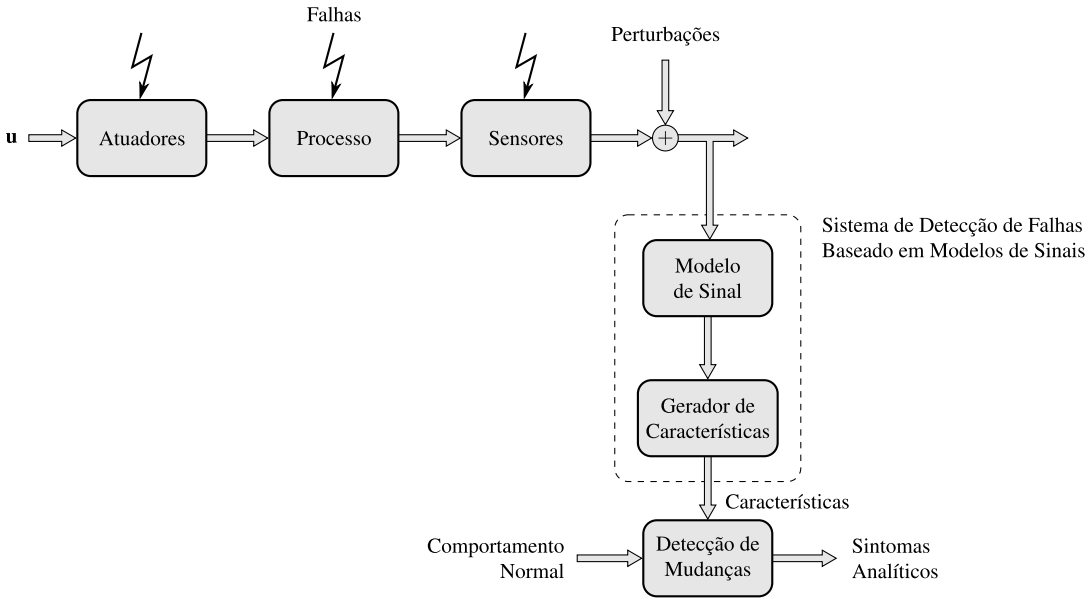


Figura 3.7: Diagrama esquemático da detecção de falhas com modelos de sinais.

3.5.3 Detecção de falhas com equações de paridade

Uma das formas de se detectar falhas de maneira direta em um processo baseia-se na comparação do comportamento real com o comportamento nominal. A diferença entre os sinais de saída do processo real e os sinais de saída do modelo matemático que descreve sua dinâmica geram os *resíduos* utilizados para detecção da falha. De acordo com Silva (2008), os resíduos são a base de várias abordagens de sistemas de Detecção e Isolamento de Falhas (DIF). A Fig. 3.8 e a Eq. 3.3 mostram como os resíduos são obtidos para um sistema de múltiplas entradas e múltiplas saídas (*Multiple Input and Multiple Output* – MIMO).

$$\begin{aligned} e(s) &= y_p(s) - y_m(s) = y_p(s) - G_m(s)u(s) \\ &= G_p(s)\left[u(s) + f_u(s)\right] + n(s) + f_y(s) - G_m(s)u(s) \\ &= \left[G_p(s) - G_m(s)\right]u(s) + G_p(s)\,f_u(s) + n(s) + f_y(s) \\ &= \Delta G_m(s)u(s) + G_p(s)\,f_u(s) + n(s) + f_y(s) \end{aligned} \tag{3.3}$$

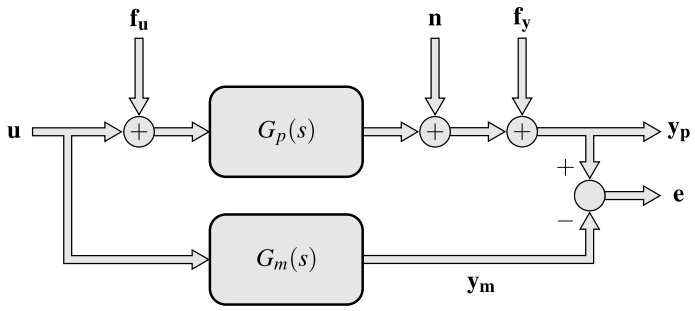


Figura 3.8: Obtenção dos resíduos para um sistema MIMO.

Nessa equação, f_u corresponde às falhas aditivas na entrada, f_y às falhas aditivas na saída, n aos ruídos, y_p à saída do processo, y_m à saída do modelo, G_p à função de transferência do processo, G_m à função de transferência do modelo, u às entradas e e aos resíduos.

Apesar da capacidade em indicar anormalidades no processo através das discrepâncias, essa abordagem possui a desvantagem de ser necessário se ter o conhecimento prévio das equações que regem a dinâmica do processo. Devido a dificuldade existente em se utilizar um modelo fenomenológico preciso, que leve em consideração, inclusive, os ruídos naturalmente presentes no processo, esse método não é muito utilizado.

3.5.4 Detecção de falhas com métodos de identificação

Os modelos matemáticos de processos descrevem uma relação entre os sinais de entrada $u(t)$ e os sinais de saída $y(t)$. Em muitos dos casos não se conhece o processo por completo, o que inviabiliza a elaboração de um modelo preciso que leve em consideração os desvios comportamentais. Dessa forma, os métodos de identificação de processos precisam ser frequentemente aplicados antes da utilização de qualquer técnica de detecção de falhas.

Conforme mostra a Fig. 3.9, existem diversos métodos de identificação de processos dinâmicos. A opção pela utilização de cada um deles varia de acordo com a aplicação.

3.5.5 Detecção de falhas com observadores e estimadores de estado

Considerando que os observadores de estado utilizam o erro de saída, dado pela diferença entre a medição da variável no processo e o modelo ajustável, pode-se dizer que eles são bons candidatos a sistemas de detecção baseados em modelos. Assume-se que, assim como no caso das abordagem que utiliza equações de paridade, a estrutura e os parâmetros do modelo precisam ser conhecidos. Dessa forma, os observadores de estado poderão ajustar as variáveis do modelo de acordo com as condições iniciais e com a evolução do processo, considerando os sinais medidos das variáveis de entrada e saída.

Um tipo de abordagem proposta para sistemas MIMO, leva em consideração um conjunto de observadores de estado tal como mostra a Fig. 3.10. De acordo com Isermann (2006), este esquema pode ser utilizado para detectar falhas aditivas em sensores. Outras

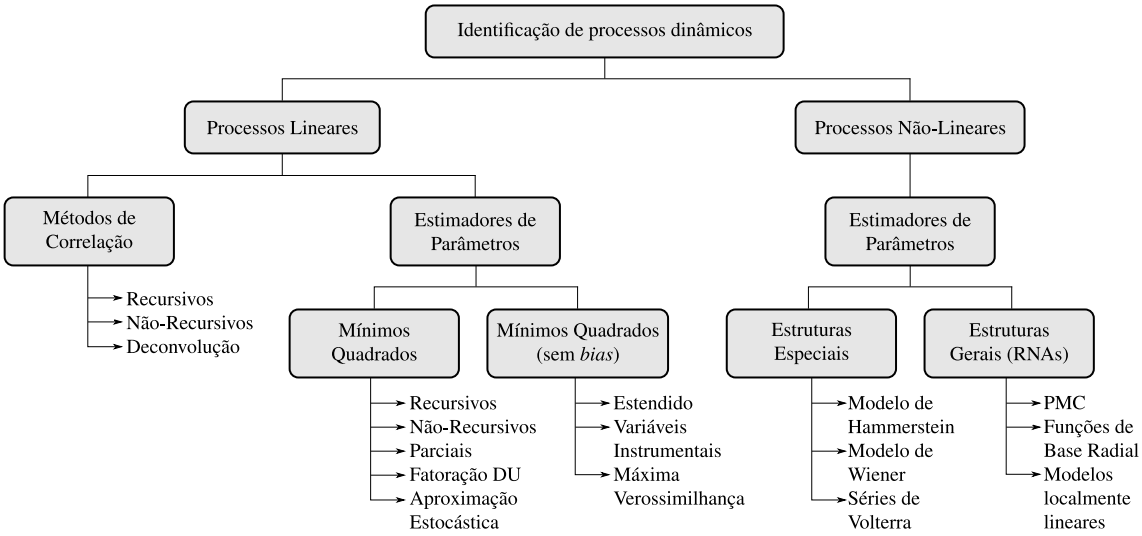


Figura 3.9: Árvore de métodos de identificação de processos dinâmicos.

abordagens fazem uso do Filtro de Kalman (estimador de estados) e de observadores de saída.

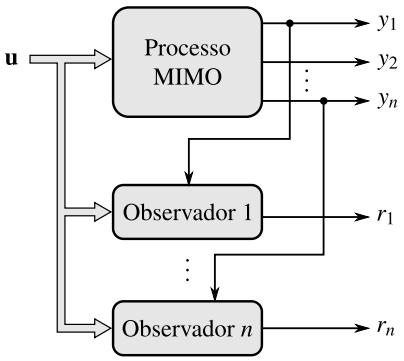


Figura 3.10: Detecção de falhas através de um conjunto de observadores de estado.

3.6 Métodos de diagnóstico

Para Silva (2008) e Isermann (2006) a tarefa de diagnosticar uma falha consiste em determinar que uma falha ocorreu, indicando o maior número de detalhes possíveis, tais como o momento de detecção, o tipo, a localização e a intensidade da falha. Este procedimento de diagnóstico baseia-se nos sintomas analíticos e nos conhecimentos heurísticos do processo.

A Fig. 3.11 sintetiza os passos necessários para o diagnóstico de falhas, tanto para variáveis medidas de maneira automática, quanto para variáveis observadas pelos operadores do processo. Em ambos os casos, a extração de características relativas as alterações comportamentais sobre o processo são necessárias. Os sintomas analíticos e heurísticos

devem ser representados de maneira unificada a fim de se realizar o diagnóstico de maneira adequada.

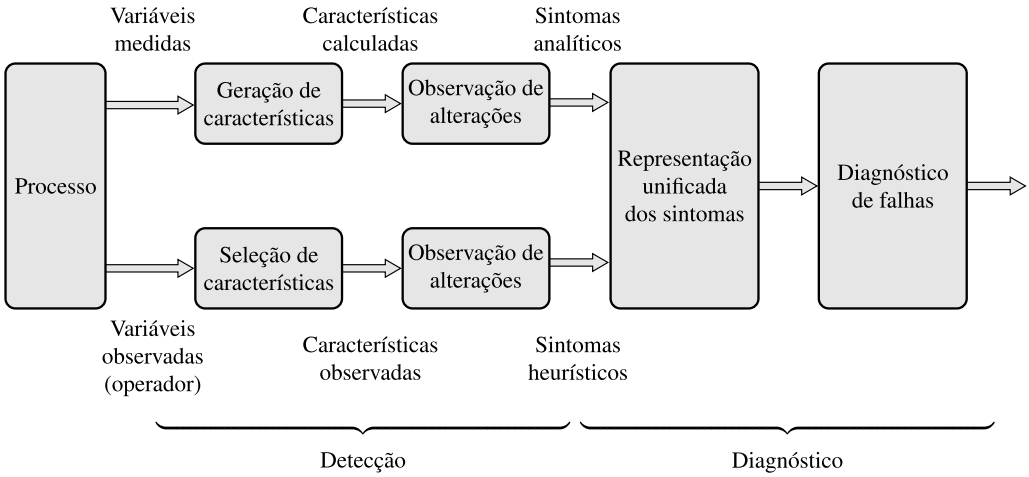


Figura 3.11: Sequências de passos que levam ao diagnóstico de falhas.

Vale observar que as *características* são os valores obtidos a partir dos sinais ou de modelos de sinais do processo, os quais descrevem o seu estado atual. Já os *sintomas* são as possíveis alterações das características com relação a seus valores normais.

As relações entre uma falha e seus sintomas seguem normalmente as relações físicas de causa e efeito. De maneira geral, pode-se dizer que uma falha produz "eventos" que por sua vez geram sintomas, conforme observado na Fig. 3.12. O diagnóstico de falhas normalmente é realizado de maneira inversa, pois o sistema deve indicar as falhas a partir dos sintomas observados.

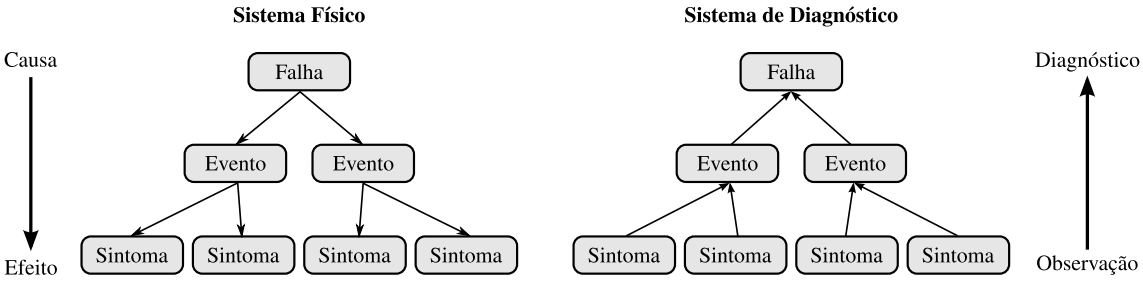


Figura 3.12: Relações de falhas e sintomas.

3.7 Detecção e diagnóstico de falhas com RNAs

Com a introdução das novas tecnologias de medição e o consequente aumento do número de instrumentos nos processos industrias, o número de informações disponíveis para um dado sistema tornou-se demasiadamente grande. Tal situação acaba por aumentar

ainda mais a complexidade de processamento, tendo em vista que cada vez mais os procedimentos de DDF utilizam dados de histórico do processo. Dentre os exemplos desses métodos que fazem uso de histórico do processo estão aqueles são baseados em RNAs.

De acordo com a Fig. 3.9, as RNAs estão classificadas como sendo estimadores de parâmetros de características gerais, utilizadas para a identificação de sistemas dinâmicos baseados em modelos de processos não-lineares. Já na Fig. 1.2(b), as RNAs aparecem como métodos de inferência que aproximam o raciocínio humano.

Unindo essas características com aquelas expostas no Cap. 2, serão utilizadas nesse trabalho duas estruturas compostas por redes neurais, com o objetivo de identificar o processo de estudo de caso e realizar a detecção e o diagnóstico de falhas a partir de seu histórico. As estruturas propostas serão apresentadas no Cap. 4, no momento em que forem mostrados maiores detalhes sobre o processo em questão.

CAPÍTULO 4

SISTEMA PROPOSTO

Existem diversos métodos consolidados para detecção e isolamento de falhas, sendo alguns deles baseados em redundância física de componentes de hardware, como sensores, atuadores e controladores. Entretanto, a duplicação de componentes de hardware nem sempre é possível, uma vez que os custos relacionados com a adição de novos componentes podem elevar o orçamento demasiadamente.

Devido a esse elevado custo, existe uma fronteira clara e inerente à aplicação de técnicas de tolerância a falhas. A escolha adequada dos dispositivos físicos e dos *softwares* devem levar em consideração as exigências específicas de cada sistema, de tal maneira que se possa contornar o custo associado ao emprego dessas técnicas [Weber 2002].

Assim sendo, a primeira parte deste capítulo descreverá em detalhes o modelo de estudo de caso escolhido, mostrando suas características e limitações. Em seguida a atenção será voltada para as estruturas neurais de identificação do modelo e de detecção das falhas, mostrando ao final como serão realizadas as simulações.

4.1 Estudo de caso

Ao final do Cap. 1, propôs-se que seria desenvolvido um sistema para realizar a DDF em um processo dinâmico. Para que isso fosse possível seriam utilizadas estruturas neurais que fariam uso de determinados valores, obtidos a partir das medições realizadas no processo e de seu histórico de funcionamento.

O processo em questão, é formado por um sistema de tanques acoplados desenvolvidos pela Quanser®, representado esquematicamente na Fig. 4.1.

O sistema original consiste de uma bomba de água de corrente contínua e dois tanques acoplados. A alimentação de água pela bomba se dá de forma vertical através de dois orifícios com conectores normalmente fechados, de diferentes diâmetros, denominados *Out 1* e *Out 2*. Para facilitar o entendimento e evitar confusões com relação aos orifícios de saída dos tanques (a_1 e a_2), esses orifícios serão tratados como orifícios de entrada.

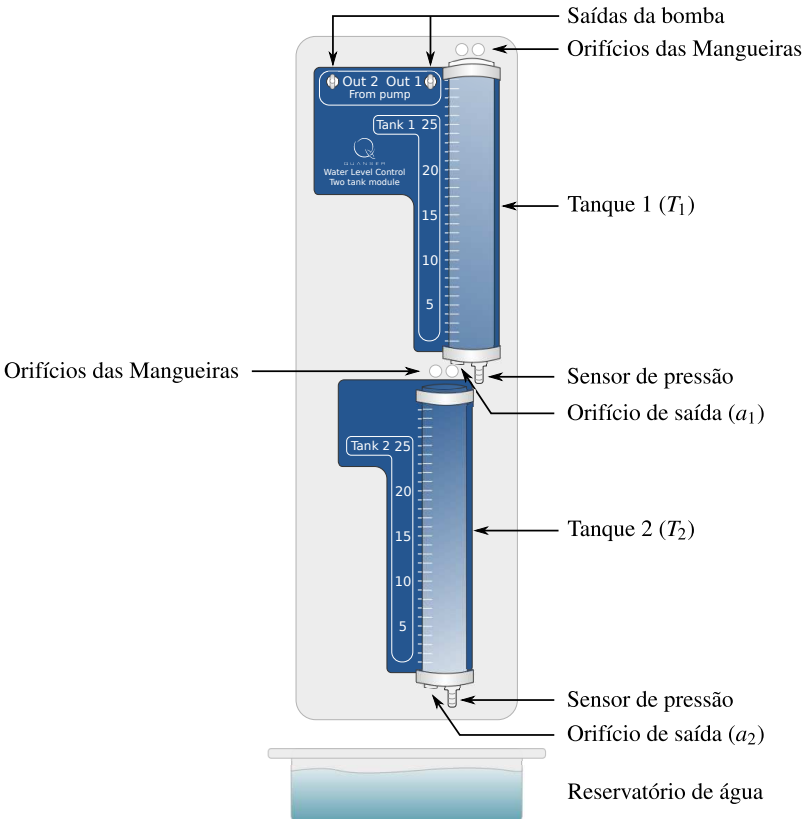


Figura 4.1: Sistema de tanques acoplados da Quanser®.

Os tanques (T_1 e T_2) são montados na parte frontal do suporte de base e posicionados de tal forma que a água flui do tanque superior (T_1) para o tanque inferior (T_2) através do orifício a_1, e do tanque inferior para o reservatório através do orifício a_2. As vazões de saída dos tanques variam de acordo com a mudança desses orifícios, disponibilizados em três diâmetros diferentes pelo fabricante.

Além dos diferentes orifícios de saída, o fabricante também disponibiliza um conjunto de mangueiras, permitindo a utilização dos dois orifícios de entrada e o bombeio de água para os dois tanques simultaneamente. As dimensões de cada um dos orifícios e os demais parâmetros do sistema podem ser visualizados na Tab. 4.1.

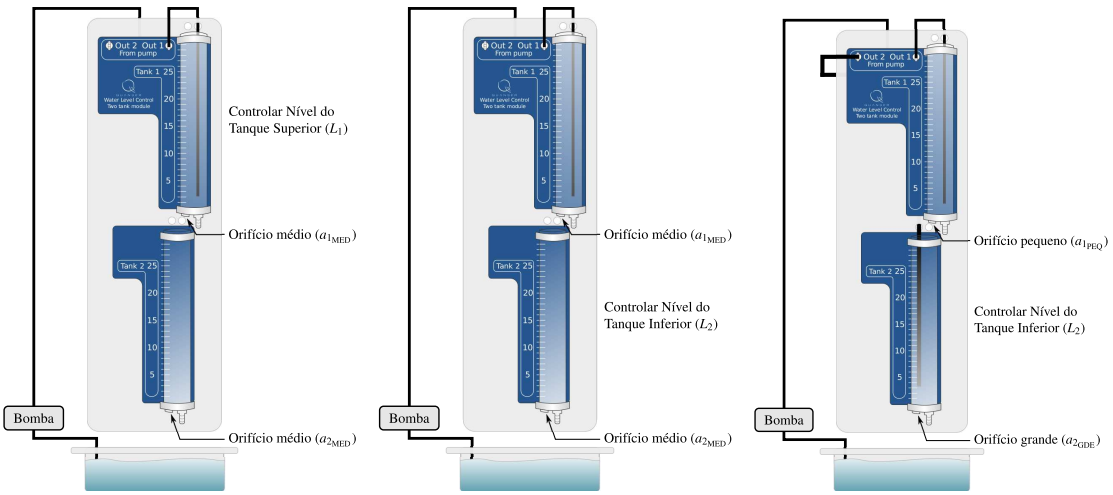
Dentre as várias combinações possíveis obtidas a partir da mudança dos orifícios de entrada e saída, as três configurações sugeridas pelo manual do fabricante encontram-se representadas na Fig. 4.2. Cada uma dessas configurações modifica a dinâmica do processo, permitindo que sejam projetados e analisados diferentes tipos de controladores.

Na primeira configuração deseja-se controlar o nível de T_1 (L_1) através de uma alimentação direta, não fazendo uso do segundo tanque. Na segunda configuração deseja-se controlar o nível de T_2 (L_2) a partir de uma alimentação indireta em T_1. Assim, T_2 é alimentado a partir da água que escoa de T_1 pelo orifício a_1. Por fim, na terceira configuração, deseja-se controlar o nível de T_2 a partir de uma alimentação indireta em T_1 e da alimentação direta de T_2. Para isso, faz-se uso dos dois orifícios de entrada.

Em todos os casos pode-se perceber que o sistema se comporta como um sistema de

Tabela 4.1: Dimensões e parâmetros do sistema de tanques.

Nome	Símbolo	Valor	Unidade
Bomba			
Constante de fluxo	K_m	4,6	$(cm^3/s)/V$
Limites de Tensão	$V_{p_{MAX/MIN}}$	± 15	Volts
Orifício de entrada 1	*Out 1*	0,635	cm
Orifício de entrada 2	*Out 2*	0,4763	cm
Tanque 1/Tanque 2			
Altura	H_1/H_2	30	cm
Diâmetro interno	–	4,445	cm
Área de secção transversal	A_1/A_2	15,517916547	cm^2
Sensibilidade do sensor	–	5	cm/V
Orifícios de saída – Diâmetros			
Orifício pequeno	$a_{i_{PEQ}}$	0,3175	cm
Orifício médio	$a_{i_{MED}}$	0,47625	cm
Orifício grande	$a_{i_{GDE}}$	0,555625	cm
Orifícios de saída – Áreas			
Orifício pequeno	$a_{i_{PEQ}}$	0,079173044	cm^2
Orifício médio	$a_{i_{MED}}$	0,178139348	cm^2
Orifício grande	$a_{i_{GDE}}$	0,242467446	cm^2
Sensores			
Range de pressão	–	0 - 1	PSI
Range de nível	–	0 - 30	cm
Constante de sensibilidade	K_s	6,25×	cm/V



(a) Configuração 1. (b) Configuração 2. (c) Configuração 3.

Figura 4.2: Configurações sugeridas pelo fabricante.

uma única entrada, que é a tensão de alimentação da bomba, e uma única saída (*Single Input and Single Output* – SISO), que pode ser L_1 ou L_2, dependendo da configuração.

Além dessas observações, deve-se destacar ainda que a planta em questão faz parte de um modelo desenvolvido para fins acadêmicos e que, por esse motivo, existem algumas restrições quanto as interfaces e aos padrões de comunicação.

4.1.1 Modelo matemático

Como os dois tanques possuem mesma área de secção transversal ($A_1 = A_2$), suas dinâmicas serão semelhantes. Entretanto, encontrar um modelo matemático que descreva adequadamente a dinâmica desses tanques não é algo tão simples, visto que as equações gerais de movimento e energia que descrevem o escoamento de fluidos são bastante complicadas [Dorf e Bishop 2009].

Assim sendo, é preciso que sejam feitas algumas hipóteses fundamentais, admitindo-se que a água no tanque é incompressível e que o escoamento é não-viscoso, não-rotacional e regular. Cada uma dessas características serão descritas a seguir baseadas na argumentação exposta em Dorf e Bishop (2009).

Diz-se que um fluido é incompressível quando este possui uma massa específica constante. Entretanto, sabe-se que todo fluido é compressível em certo grau, e que, o fator de compressibilidade k é uma medida de tal característica. Quanto menor o valor de k, menor é a compressibilidade indicada para aquele fluido. O ar, que é um fluido compressível, possui um fator de compressibilidade $k_{\text{ar}} = 0{,}98\ \text{m}^2/\text{N}$, enquanto que a água tem um fator de compressibilidade de $k_{\text{H}_2\text{O}} = 4{,}9 \times 10^{-10}\ \text{m}^2/\text{N} = 50 \times 10^{-6}\ \text{atm}^{-1}$. Em outras palavras, um dado volume de água diminui em 50 milionésimos de seu volume original para um aumento de uma atmosfera de pressão. Desse modo, a hipótese de que a água é incompressível é valida para o sistema em questão.

Já a viscosidade de um fluido é dada pelo seu coeficiente de viscosidade μ $(\text{N}\cdot\text{s}/\text{m}^2)$. Quanto maior esse coeficiente, mais viscoso é o fluido. Como exemplo, o coeficiente de viscosidade em condições normais à 20°C para o ar é $\mu_{\text{ar}} = 0{,}178 \times 10^{-4}\ \text{N}\cdot\text{s}/\text{m}^2$ enquanto que para a água $\mu_{\text{H}_2\text{O}} = 1{,}054 \times 10^{-3}\ \text{N}\cdot\text{s}/\text{m}^2$. Assim, a água é cerca de 60 vezes mais viscosa que o ar. Vale salientar que a viscosidade depende principalmente da temperatura e não da pressão. Para efeitos de comparação, a água à 0°C é 2 vezes mais viscosa que a água à 20°C. Com fluidos de baixa viscosidade como o ar e a água, os efeitos do atrito são importantes apenas nas camadas de fronteira e em uma fina camada adjacente à parede do reservatório e da tubulação de saída. Assim, pode-se desprezar a viscosidade no desenvolvimento do modelo.

Se cada elemento do fluido em cada ponto do escoamento não tem velocidade angular com relação a esse ponto, o fluxo é denominado não-rotacional. Admita que a água no tanque é não-rotacional. Logo, para um fluido não-viscoso, um fluxo inicialmente não-rotacional permanece não-rotacional.

Por fim, o fluxo de água é dito regular se sua velocidade em cada ponto é constante com o tempo. Isso não implica necessariamente que a velocidade seja a mesma em cada ponto, mas sim que em qualquer ponto a velocidade não mude com o tempo. Condições de regime regular podem ser atingidas em velocidades baixas do fluido. Admita então que há condições de fluxo regular. Observa-se entretanto que se a área de abertura de saída fosse muito grande, o fluxo através do reservatório não seria lento o suficiente para

o estabelecimento das condições de regime regular o que faria com que o modelo não conseguisse predizer o fluxo do fluido de maneira exata.

Tendo esclarecido essas condições, para se obter o modelo matemático do fluido dentro do reservatório empregam-se os princípios básicos da ciência e da engenharia, como o princípio da conservação da massa. Assim, a massa de água em qualquer instante de tempo é dada pela Eq. 4.1.

$$m = \rho_{H_2O} AL \tag{4.1}$$

em que ρ_{H_2O} é a massa específica da água, A a área de secção transversal do reservatório e L o nível de água no reservatório. Tomando a derivada temporal de m na Eq. 4.1, tem-se:

$$\dot{m} = \rho_{H_2O} A\dot{L} \tag{4.2}$$

na qual utilizou-se o fato do fluido ser incompressível (ou seja, $\dot{\rho} = 0$). Como a área de secção transversal não varia com o tempo, a mudança de massa no reservatório é igual a massa que entra menos a massa que deixa o tanque, logo:

$$\dot{m} = \rho_{H_2O} A\dot{L} = \rho_{H_2O} Q_{IN} - \rho_{H_2O} \underbrace{a\upsilon_{OUT}}_{Q_{OUT}} \tag{4.3}$$

em que Q_{IN} é a vazão de entrada de massa em regime permanente, a é a área do orifício de saída e υ_{OUT} é a velocidade de saída do fluido, que é uma função da altura da água. Da equação de Bernoulli [Houghton e Carpenter 2002], tem-se:

$$P_{IN} + \frac{1}{2}\rho_{H_2O}\,{\upsilon_{IN}}^2 + \rho_{H_2O}\, gL = \frac{1}{2}\rho_{H_2O}\,{\upsilon_{OUT}}^2 + P_{OUT} \tag{4.4}$$

em que υ_{IN} é a velocidade da água na entrada do reservatório, g é a aceleração da gravidade e P_{IN} e P_{OUT} são as pressões na entrada e na saída, respectivamente. Mas P_{IN} e P_{OUT} são iguais à pressão atmosférica e a área de a é suficientemente pequena ($a \approx A/85$, considerando o orifício de saída médio) para que a água escoe vagarosamente e a velocidade υ_{IN} seja desprezível. Assim, a equação de Bernoulli fica reduzida a:

$$\upsilon_{OUT} = \sqrt{2gL} \tag{4.5}$$

Além disso, segundo Apkarian (1999), a vazão de entrada Q_{IN} depende apenas da constante da bomba e da tensão a ela aplicada, podendo ser positiva, quando a água é sugada do reservatório e injetada no tanque, ou negativa quando a água é sugada do tanque e devolvida ao reservatório. Assim:

$$Q_{IN} = K_m V_p \tag{4.6}$$

Substituindo então as Eqs. 4.5 e 4.6 na Eq. 4.3 e resolvendo para $\dot{L}$, tem-se:

$$\dot{L} = -\left[\frac{a}{A}\sqrt{2g}\right]\sqrt{L} + \frac{1}{\rho_{H_2O} A} K_m V_p \tag{4.7}$$

Considerando que a massa específica da água é $\rho_{H_2O} \approx 1{,}0$ à 20°C, então a Eq. 4.7 passa a ser:

$$\dot{L} = -\left[\frac{a}{A}\sqrt{2g}\right]\sqrt{L} + \frac{K_m}{A}V_p \tag{4.8}$$

Assim, as duas únicas variáveis do modelo serão o nível L do tanque e a tensão V_p aplicada à bomba. Perceba que essa é uma equação diferencial ordinária não-linear.

4.1.2 Modificações do modelo

Observa-se que Eq. 4.8 representa adequadamente a dinâmica do tanque superior, contudo, a dinâmica do tanque inferior age um pouco diferente. Em primeiro lugar, dependendo da configuração escolhida T_2 poderá ou não receber uma alimentação direta. Considerando, por exemplo, a primeira configuração sugerida pelo manual do fabricante, mostrada na Fig. 4.2(a), a dinâmica de T_2 irá depender apenas da vazão de água que escoa de T_1 e da vazão de água que retorna ao reservatório. Assim sendo, a dinâmica dos dois tanques pode ser representada pelas Eqs. 4.9 e 4.10:

$$\dot{L_1} = \frac{K_m}{A}V_p - \left[\frac{a_1}{A}\sqrt{2g}\right]\sqrt{L_1} \tag{4.9}$$

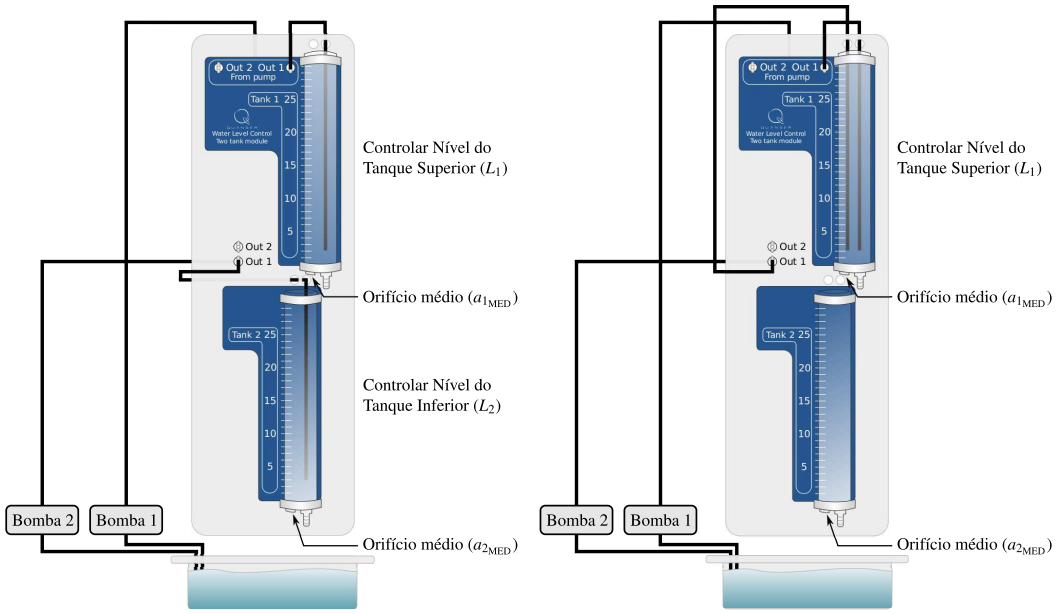
$$\dot{L_2} = \left[\frac{a_1}{A}\sqrt{2g}\right]\sqrt{L_1} - \left[\frac{a_2}{A}\sqrt{2g}\right]\sqrt{L_2} \tag{4.10}$$

em que a_1 representa o orifício de saída de T_1, a_2 o orifício de saída de T_2, L_1 o nível de água em T_1 e L_2 o nível e água em T_2. Perceba que não há necessidade em se discriminar a área de secção transversão através de um subíndice, pois $A_1 = A_2 = A$.

Com o intuito de deixar o sistema proposto com um caráter mais genérico e possibilitar que sejam realizados novos estudos relacionados com o tema de tolerância à falhas, optou-se por modificar o sistema de tanques original introduzindo uma outra bomba com as mesmas características da primeira. Dessa forma, pôde-se obter novas configurações, dentre as quais destacam-se aquelas expostas pela Fig. 4.3.

Na configuração exposta pela Fig. 4.3(a), pode-se controlar o nível dos dois tanques de maneira individual através das duas bombas de água. Já na configuração exposta pela Fig. 4.3(b), a segunda bomba está sendo utilizada para garantir que o processo continue funcionando normalmente quando a primeira bomba vier a falhar. Ou seja, possui-se dois atuadores com um deles em *standby*. Ademais, é válido ressaltar que essa última configuração pode ser facilmente adaptada para qualquer uma das três anteriores sugeridas pelo fabricante.

É evidente que, no caso da Fig. 4.3(a), o sistema deixa de ser SISO e passa a ser MIMO. Assim sendo, esse sistema será o utilizado como modelo de estudo de caso no trabalho. A única modificação existente nas equações que regem a dinâmica do modelo é a introdução de uma nova variável na equação do segundo tanque, em virtude da adição da segunda bomba. Assim, as equações finais do modelo de estudo de caso serão:



(a) Nova configuração 1. (b) Nova configuração 2.

Figura 4.3: Novas configurações sugeridas após a modificação do modelo.

$$\dot{L_1} = \frac{K_m}{A} V_{p_1} - \left[\frac{a_1}{A}\sqrt{2g}\right]\sqrt{L_1} \tag{4.11}$$

$$\dot{L_2} = \frac{K_m}{A} V_{p_2} + \left[\frac{a_1}{A}\sqrt{2g}\right]\sqrt{L_1} - \left[\frac{a_2}{A}\sqrt{2g}\right]\sqrt{L_2} \tag{4.12}$$

em que V_{p_1} se refere à tensão aplicada na Bomba 1 (B_1) e V_{p_2} à tensão aplicada na Bomba 2 (B_2).

4.1.3 Limitações do processo

Conforme dito anteriormente, a planta do estudo de caso foi desenvolvida para fins acadêmicos e possui algumas limitações com relação as interfaces e aos padrões de comunicação. Por causa dessas limitações a planta é normalmente controlada por um sistema desenvolvido computacionalmente em que a comunicação se dá através de um barramento ISA (*Industry Standard Architecture*). Tal sistema deve calcular as ações de controle proporcional, integral e derivativa e produzir um sinal de controle em Volts que será enviado aos atuadores.

Entretanto, devido aos baixos níveis de tensão intrínsecos à esse tipo de barramento, faz-se necessário utilizar um amplificador, também fornecido pela Quanser®, cujo fator de amplificação é de cinco vezes. Assim, para que a bomba não venha a queimar, os sistemas de controle devem implementar uma rotina de segurança que garanta que o sinal de controle não supere os limites de ± 3 Volts. A representação esquemática do funcionamento do sistema como um todo está representada na Fig. 4.4.

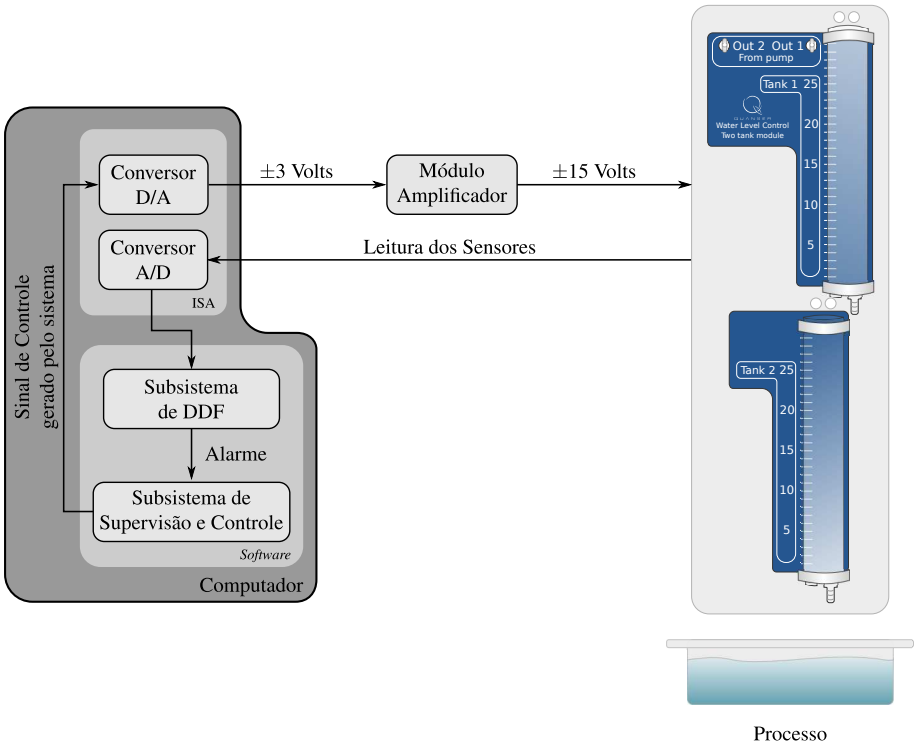


Figura 4.4: Diagrama esquemático de funcionamento do sistema proposto.

A partir dessa representação esquemática, percebe-se facilmente que as únicas variáveis disponíveis são os níveis dos tanques (L_1 e L_2) e as tensões das bombas (V_{p_1} e V_{p_2}).

Além das limitações físicas relacionadas aos sinais enviados ou recebidos pelo processo, destaca-se ainda o fato de se tratar de uma planta real em que todos os dispositivos físicos e mecânicos estão sujeitos a deteriorações relacionadas ao tempo de uso. Assim sendo, não se considera uma boa prática estimular ou induzir falhas no sistema real. Logo, todas as falhas a serem analisadas serão induzidas através do modelo matemático simulado em um sistema computacional.

4.2 Simulações do sistema e das falhas

Considerando os fatores expostos ao final da seção 4.1.3, o sistema de tanques será simulado a partir das Eqs. 4.11 e 4.12, utilizando o método de Runge-Kutta de 4ª ordem (RK4).

De maneira resumida, pode-se dizer que o RK4 faz parte de uma família de métodos iterativos para a aproximação numérica de soluções de equações diferenciais ordinárias. O funcionamento do método pode ser esclarecido conforme explicação a seguir.

Seja um problema de valor inicial dado por:

$$y' = f(t,y) \qquad y(t_0) = y_0 \tag{4.13}$$

Então, o método RK4 para este problema é dado pelas seguintes equações:

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \quad (4.14)$$

$$t_{n+1} = t_n + h \quad (4.15)$$

em que y_{n+1} é a aproximação por RK4 de $y(t_{n+1})$, e:

$$k_1 = f(t_n, y_n) \quad (4.16)$$

$$k_2 = f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_1\right) \quad (4.17)$$

$$k_3 = f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_2\right) \quad (4.18)$$

$$k_4 = f(t_n + h, y_n + hk_3) \quad (4.19)$$

Assim, o próximo valor (y_{n+1}) é determinado a partir do valor atual (y_n) somado com o produto do tamanho do intervalo (h) e uma inclinação estimada, obtida a partir da média ponderada das inclinações k_i. Percebe-se ainda que a média ponderada dá um maior peso para as inclinações no ponto médio.

O método de RK4 permite que sejam determinados os valores de L_1 e L_2 para cada instante da simulação e, consequentemente, que a dinâmica do processo seja observada. As falhas induzidas no processo serão então computadas a partir de alterações feitas sobre os valores de L_1 e L_2 obtidos. Dependendo do tipo de falha, a alteração pode ocorrer antes, durante ou após a execução do método de RK4.

4.2.1 Falhas simuladas

Assim como já foi visto, devido ao elevado grau de complexidade e à integração inter processos, as falhas que antes poderiam ser detectadas através de medições diretas de determinadas variáveis, passam a depender de um conjunto de variáveis que atuam simultaneamente no sistema. Ademais, as causas das falhas podem nem sempre estar próximas dos locais em que foram identificadas.

Considerando tais aspectos, foram selecionadas somente algumas falhas dentre aquelas que podem ocorrer no processo em estudo. Alguns exemplos de falhas desse sistema podem ser observados na Tab. 4.2. O agrupamento e a classificação das falhas escolhidas se deu conforme Tabs. 4.3 e 4.4.

As siglas expostas nessas tabelas foram introduzidas com o intuito de facilitar, tanto o processo automático de treinamento e validação da estrutura neural, quanto a referenciação textual.

As falhas expostas na Tab. 4.4 são induzidas de diferentes maneiras na simulação. Na FADG, por exemplo, o ganho do atuador é alterado de 1,0 para qualquer outro valor a escolha do usuário, no momento em que é realizada a leitura dos níveis. Já na FSiVzT, por questões de simplicidade, supõe-se que um outro orifício de saída foi adicionado ao lado do orifício de saída do tanque (a_i). Esse orifício, de área arbitrária, terá a mesma dinâmica do orifício de saída do tanque, exceto que para T_1 a água que por ele escoa não irá influenciar na dinâmica de T_2.

Tabela 4.2: Exemplos de falhas que podem ocorrer no sistema de tanques.

Sensores	**Atuadores**	**Estruturais**
Erro de leitura	Erro de escrita	Erro de transmissão
Descalibramento	Erro de leitura	Perda de comunicação
Sensibilidade a ruído	Sensibilidade a ruído	Sensibilidade à ruído
Queima do sensor	Queima do atuador	Queima do transmissor
–	Atraso de transporte	Atraso de propagação

Tabela 4.3: Grupos de falhas.

Grupo	**Tipo de falha**	**Sigla**
0	Não há falhas	SF
1	Falha do sensor	FSe
2	Falha do atuador	FA
3	Falha estrutural ou Falha do sistema	FSi

Tabela 4.4: Classificação das falhas selecionadas.

Grupo	**Classe**	**Denominação**	**Variante**	**Sigla**
1	1	Descalibramento	Ganho	FSeDG
1	1	Descalibramento	Nível DC (*offset*)	FSeDO
1	2	Sensibilidade	Ruído	FSeSR
1	3	Queima	–	FSeQ
2	1	Descalibramento	Ganho	FADG
2	1	Descalibramento	Nível DC (*offset*)	FADO
2	2	Sensibilidade	Ruído	FASR
2	3	Variação	Constante da bomba (K_m)	FAVK
2	4	Queima	–	FAQ
3	3	Vazamento	Tanque	FSiVzT
3	2	Variação	Orifício de saída	FSiVrOS
3	2	Variação	Ganho do módulo de potência	FSiVrGMP
3	1	Entupimento	Orifício de saída	FSiEOS

4.3 Estruturas neurais escolhidas

Tanto para a identificação do modelo quanto para a detecção e o diagnóstico das falhas, deve-se ter bastante cuidado ao se escolher a estrutura e a arquitetura das redes neurais. Em ambos os casos, uma escolha inadequada pode fazer com que o sistema não se comporte de maneira adequada e não realize a função para o qual foi designado.

Assim sendo, para que fosse realizada uma análise comparativa, foram propostas duas estruturas neurais para o processo de identificação do modelo e duas outras estruturas para a detecção e o diagnóstico das falhas. Nos quatro casos foram utilizados modelos NNARX

com uma camada oculta com função de ativação sigmoidal. O número de regressores e o número de neurônios nessa camada foram determinados a partir de testes, conforme mostrado no Cap. 5. Os detalhes relativos a cada uma dessas estruturas serão esclarecidos nas próximas seções.

4.3.1 Propostas de identificação

A primeira estrutura de identificação busca representar a dinâmica do sistema de tanques através de uma única rede neural, a qual receberia como entrada os valores passados dos níveis, $L_1(k-1)$ e $L_2(k-1)$, e os valores de tensão aplicados, $V_{p_1}(k-1)$ e $V_{p_2}(k-1)$, gerando uma estimativa dos níveis em sua saída, denominadas $\widehat{L_1}(k)$ e $\widehat{L_2}(k)$. A Fig. 4.5 representa um diagrama esquemático do funcionamento dessa primeira proposta.

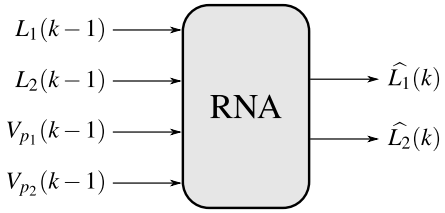


Figura 4.5: Proposta de identificação global.

Já a segunda estrutura de identificação leva em consideração que cada tanque possui uma dinâmica diferente, tendo em vista que o T_1 age de maneira independente, enquanto que a dinâmica de T_2 depende da dinâmica de T_1. Assim, optou-se por utilizar duas redes neurais, como mostrado na Fig. 4.6.

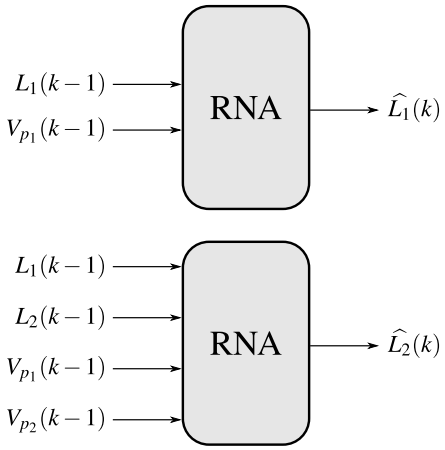


Figura 4.6: Proposta de identificação individual.

A primeira rede seria a responsável pela identificação da dinâmica de T_1, recebendo como entrada o nível anterior deste tanque, $L_1(k-1)$, e a tensão aplicada à bomba que o alimenta, $V_{p_1}(k-1)$, gerando uma estimativa de L_1 em sua saída. Já a segunda rede seria a responsável pela identificação da dinâmica de T_2 e receberia como entrada as mesmas

variáveis da primeira proposta de identificação, ou seja, $L_1(k-1)$, $L_2(k-1)$, $V_{p_1}(k-1)$ e $V_{p_2}(k-1)$. A saída da segunda rede seria, nesse caso, uma estimativa de L_2.

Independentemente da proposta, pode-se visualizar o sistema de identificação do modelo como uma única estrutura neural (simples ou composta), que possui como entrada os níveis anteriores e as tensões aplicadas, e que produz em sua saída as estimativas dos níveis dos dois tanques. A Fig. 4.7 mostra como o sistema de identificação pode ser visto.

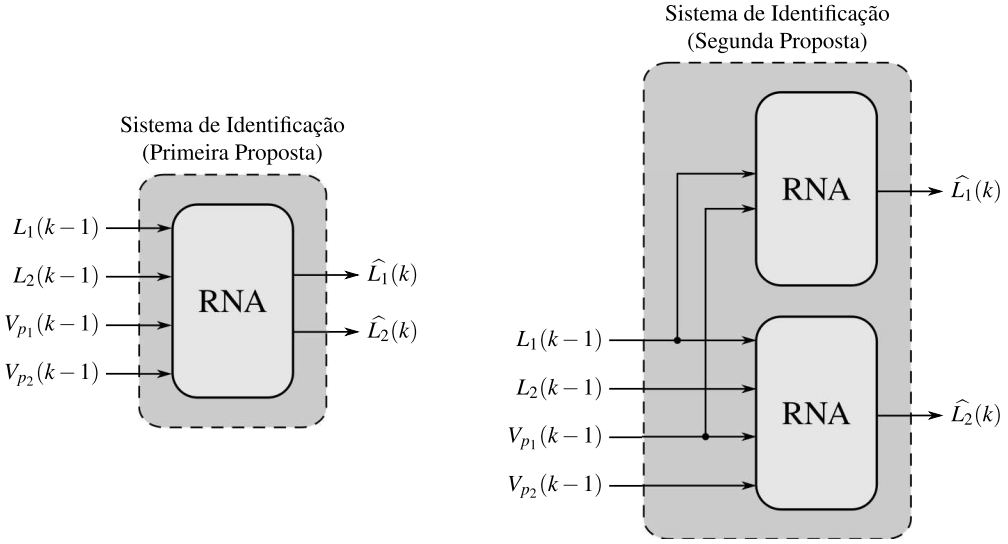


Figura 4.7: Visão geral do sistema de identificação.

4.3.2 Propostas de detecção

As estruturas neurais de DDF seguiram o mesmo raciocínio adotado para as estruturas de identificação. No primeiro caso, buscou-se desenvolver uma única rede neural que fosse capaz de detectar e identificar cada uma das falhas listadas pela Tab. 4.4. Essa rede receberia como entrada os mesmos sinais das redes de identificação e os resíduos e_i, produzidos pela diferença dos níveis entre o modelo simulado (L_i) e o modelo identificado ($\widehat{L}_i$). A saída da rede formaria uma palavra binária que seria decodificada a partir de uma tabela previamente estabelecida. Nessa tabela cada falha teria uma palavra binária correspondente. A Fig. 4.8 representa um diagrama esquemático dessa proposta.

Na segunda proposta, a estrutura de DDF seria composta por várias redes neurais, em que cada uma destas estaria associada a uma única falha, formando um conjunto de "especialistas". Não se trata, entretanto, de uma máquina de comitê de especialistas, uma vez que não existe uma rede que realiza a tomada de decisões.

Inicialmente, as entradas de cada uma das redes seriam as mesmas da primeira proposta de detecção. As saídas, por sua vez, seriam compostas por uma palavra de apenas 2 bits, indicando se aquela falha estaria sendo detectada em T_1 em T_2 ou em T_1 e T_2 simultaneamente. A Fig. 4.9 representa um diagrama esquemático dessa proposta.

Existem diversas vantagens em se utilizar a segunda proposta ao invés da primeira, como por exemplo, a de se utilizar entradas diferentes para cada uma das redes especi-

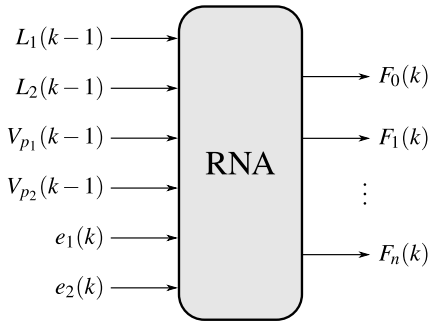


Figura 4.8: Proposta de detecção através de uma única rede neural.

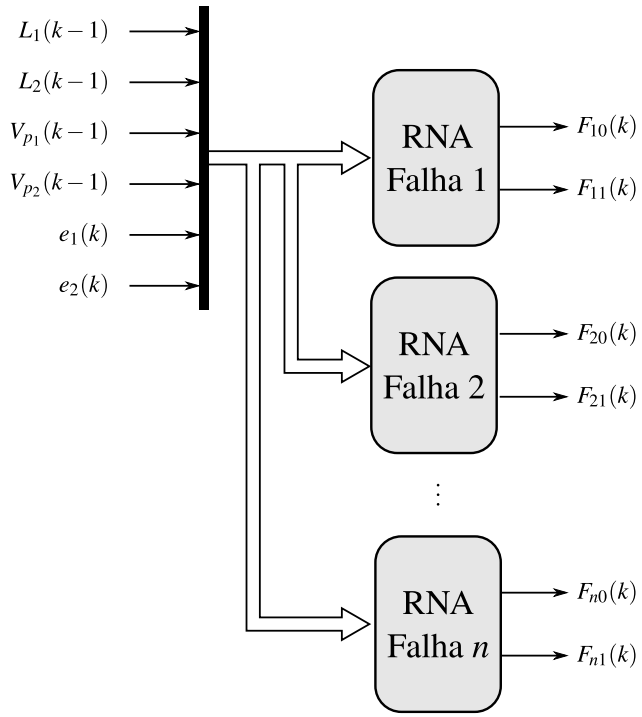


Figura 4.9: Proposta de detecção através de um conjunto de redes especialistas.

alistas. Essa situação poderia ocorrer se fosse percebido que uma determinada entrada, ao ser adicionada, fizesse com que aquela falha fosse detectada de maneira mais precisa ou mais rápida. Essa vantagem também é percebida na situação inversa, em que uma determinada entrada pode estar sendo indiferente com relação ao processo de detecção.

Com relação ao tempo de treinamento e ao processamento dos dados, pode-se dizer que qualquer modificação na estrutura das entradas da primeira proposta faria com que a rede precisasse ser treinada novamente. Por se tratar, provavelmente, de uma estrutura mais complexa, o treinamento poderia ter um grande custo relacionado ao tempo e ao processamento. Já na segunda proposta apenas uma das redes seria retreinada, o que iria reduzir esse custo de maneira considerável. Além disso, se o treinamento fosse realizado em tempo real, não haveria necessidade de se parar o sistema de detecção como um todo, mas somente uma parte deste.

Ainda sobre o processamento dos dados, percebe-se que, em uma estrutura como a da segunda proposta, diversos tipos e classes de módulos de detecção e/ou diagnóstico podem ser implementados. Dependendo do tipo de falha, técnicas híbridas, que envolvam métodos *Fuzzy*, algorítimos genéticos, métodos estatísticos e quaisquer outras técnicas que existam ou que venham a ser criadas, podem vir a melhorar a eficiência de detecção. Dessa forma, o sistema que antes seria composto somente por redes neurais, passaria a ser composto por diversos módulos que utilizariam técnicas distintas.

Uma outra vantagem estaria relacionada com o pós-processamento da informação. Ao se utilizar a segunda proposta percebe-se que não é necessário realizar uma decodificação da saída a cada vez que uma entrada é introduzida na rede. Considerando que o tempo gasto para decodificar a palavra na saída da primeira proposta, é diretamente proporcional ao tamanho da palavra e que este, por sua vez, está diretamente relacionado com número de falhas a serem identificadas, pode-se dizer que quanto maior for o número de falhas a serem identificadas, maior será esse tempo de decodificação.

Além disso, vale ainda salientar que as falhas podem acontecer de maneira simultânea e independente, o que iria produzir um grande número de combinações de falhas. Para um conjunto de 13 (treze) falhas, por exemplo, considerando que somente 2 (duas) destas poderiam atuar simultaneamente, teria-se um total de 78 (setenta e oito) combinações distintas. Se todas as combinações fossem consideradas esse número cresceria de maneira exorbitante.

Como cada uma dessas combinações deve ser identificada de maneira única pelo sistema, há então a necessidade de se representar as combinações dentre as palavras de saída. Assim, além das palavras binárias que representariam cada uma das falhas, precisariam existir as palavras binárias que representariam as combinações de cada uma dessas, aumentando ainda mais o número de bits e, consequentemente o tempo de decodificação.

Um outro ponto que pode ser destacado é que ao se utilizar a segunda proposta, torna-se mais fácil realizar o acoplamento do sistema de DDF com um sistema de monitoramento e geração de alarme, pois cada falha seria acionada de maneira individual. Dessa forma, os sinais de acionamento poderiam conter informações pertinentes para que os respectivos alarmes fossem gerados de maneira mais precisa.

Uma última consideração a ser feita é que, assim como o sistema de identificação, o sistema de detecção também pode ser visto como uma única estrutura neural. A única observação a ser feita é que a saída dessa estrutura poderá ser uma palavra binária de p bits, tal que $2^p \geq n$, em que n representa o número de palavras a serem identificadas; ou n palavras binárias de 2 bits. Nesse último caso n representa simplesmente o número de falhas a serem identificadas, pois as combinações de falhas são obtidas a partir das saídas ativas de cada um dos módulos.

4.4 Composição do sistema

Tendo conhecido todos os subsistemas a serem utilizados, pode-se observar na Fig. 4.10 a estrutura do sistema de DDF. Deve-se destacar entretanto que o trabalho aqui proposto não incorpora o desenvolvimento de um sistema de supervisão e monitoramento completo, mas somente o desenvolvimento do sistema de DDF.

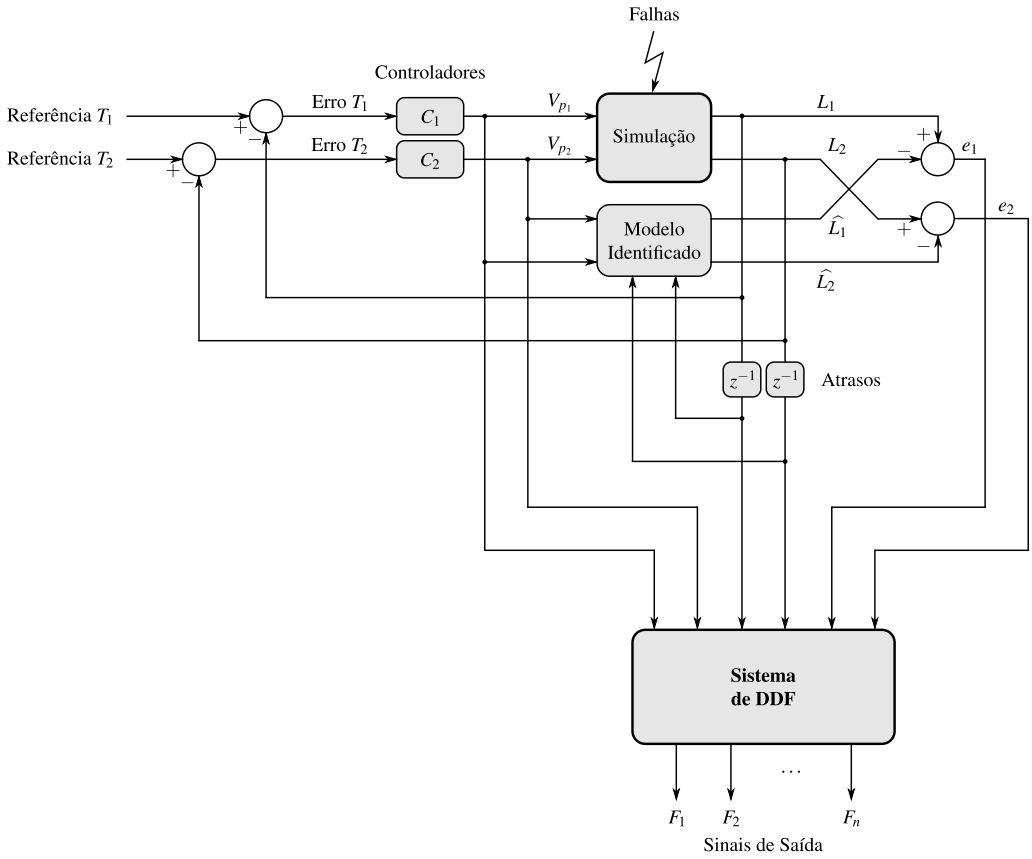


Figura 4.10: Composição do sistema proposto.

Acoplar tal sistema a um ambiente de monitoramento e supervisão, ou ainda associá-lo a técnicas de controle tolerante à falhas, consiste simplesmente em processar as informações disponíveis na interface de saída do módulo de DDF. Contudo, isso não faz parte do escopo deste trabalho.

4.5 Sistemas computacionais desenvolvidos

Para dar mais liberdade de configuração à simulação e ao sistema de DDF, foram desenvolvidos dois sistemas computacionais, utilizando C++ como linguagem de programação, a biblioteca Qt para o desenvolvimento das interfaces gráficas e a biblioteca *Flood* para a implementação das redes neurais.

O primeiro sistema, cuja captura de tela (*screenshot*) pode ser vista na Fig. 4.11, tinha por objetivo simular o funcionamento do sistema de tanques. Para isso, fez-se uso do método de RK4 e da utilização de arquivos de texto simples para realizar a configuração de todos os parâmetros referentes a cada uma dessas falhas. Os parâmetros de configuração são especificados em cada uma das colunas do arquivo de entrada, conforme Tab. 4.5, e o número de linhas equivale ao número de amostras a serem simuladas. O período de amostragem utilizado na simulação do sistema é de 100 (cem) milissegundos, o mesmo

utilizado para o controle da planta real.

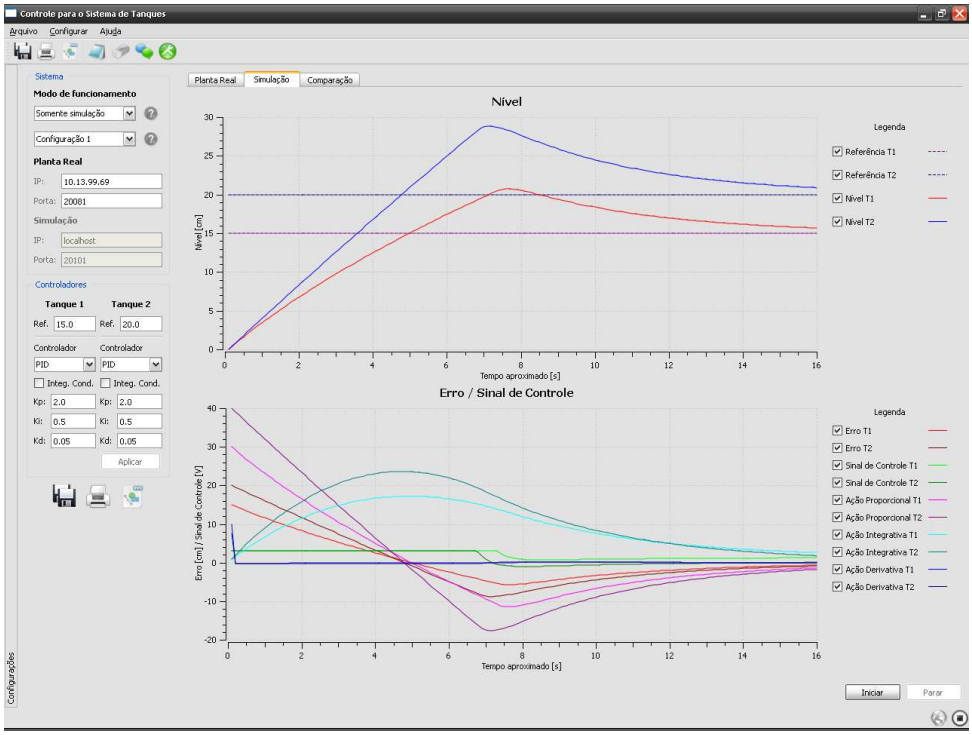
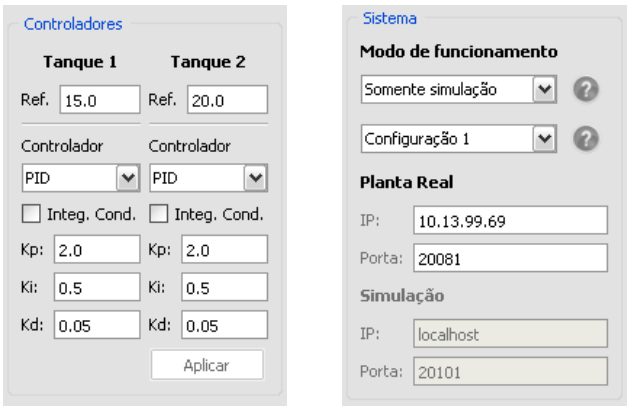


Figura 4.11: Captura de tela do sistema de simulação em funcionamento.

Esse sistema também implementa as rotinas de dois controladores (um para cada bomba) que podem ser configurados para operarem como controladores P, PI, PD, PID ou PI-D. Os ganhos proporcionais, integrais e derivativos são introduzidos na interface do usuário, conforme mostra a Fig. 4.12(a).



(a) (b)

Figura 4.12: Captura de tela dos campos de ajuste dos controladores, IP e porta.

Além disso, o sistema pode ainda ser utilizado para operar com a planta real, fazendo aquisições dos dados através da rede (local ou remota), utilizando *sockets* TCP, se co-

Tabela 4.5: Especificação das colunas do arquivo de configuração da simulação.

Coluna	**Descrição**
1	Tempo
2	Ganho do sensor – T_1
3	Ganho do sensor – T_2
4	Nível DC (*offset*) do sensor – T_1
5	Nível DC (*offset*) do sensor – T_2
6	Porcentagem de ruído do sensor – T_1
7	Porcentagem de ruído do sensor – T_2
8	Ganho do atuador – B_1
9	Ganho do atuador – B_2
10	Nível DC (*offset*) do atuador – B_1
11	Nível DC (*offset*) do atuador – B_2
12	Porcentagem de ruído do atuador – B_1
13	Porcentagem de ruído do atuador – B_2
14	Área do orifício do vazamento – T_1
15	Área do orifício do vazamento – T_2
16	Constante da bomba – K_{m_1}
17	Constante da bomba – K_{m_2}
18	Área do orifício de saída – a_1
19	Área do orifício de saída – a_2
20	Ganho do módulo de potência – Multiplicador de V_{p_1}
21	Ganho do módulo de potência – Multiplicador de V_{p_2}
22	Referência (*setpoint*) – T_1
23	Referência (*setpoint*) – T_2

municando com um servidor implementado conforme descrito em Oliveira (2008). A configuração do endereço IP e da porta de comunicação também é realizada a partir da interface do usuário, conforme Fig. 4.12(b).

A saída desse sistema é composta por até quatro arquivos, dependendo da configuração escolhida, contendo os valores da referência, dos níveis, do erro, das ações de controle (P, I e D) e do sinal de controle enviado para cada uma das bombas.

Já o segundo sistema, denominado *Sistema modular para detecção e diagnóstico de falhas* (Simddef), foi idealizado para processar as informações de saída do primeiro sistema (simulador) e exibir as detecções das falhas de maneira simples e intuitiva. Para isso, as matrizes de saída do simulador deverão ser pré-processadas para que se faça uma adequação das variáveis àquelas utilizadas por cada um dos módulos de detecção e/ou diagnóstico.

Contudo, devido a possibilidade de se introduzir diversos tipos de módulos diferentes e de se inserir e remover falhas no processo (a critério do usuário), esse sistema foi projetado para ser configurado a partir de arquivos XML (*eXtensible Markup Language*). Assim, tanto as falhas quanto os módulos são carregados a partir desses arquivos que seguem uma formatação pré-estabelecida, conforme descrito em detalhes no Apêndice A.

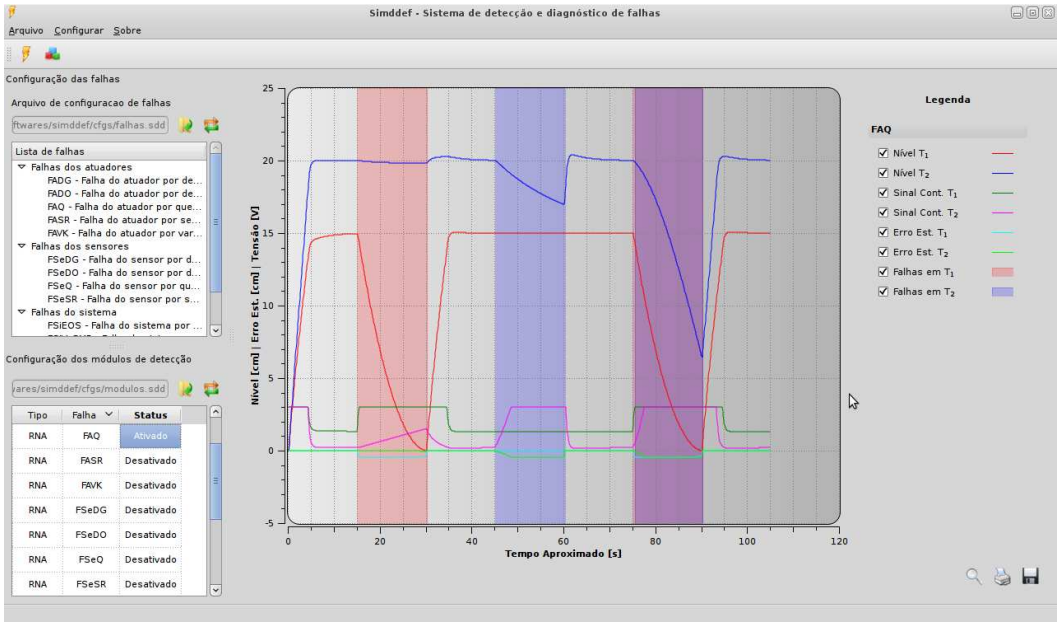


Figura 4.13: Captura de tela do *Simddef* em funcionamento.

Uma captura de tela desse sistema em funcionamento pode ser observada na Fig. 4.13.

Por fim, vale ressaltar que, dependendo do número de falhas e módulos a serem configurados, as informações contidas nesses arquivos poderão ser posteriormente disponibilizadas em um banco de dados. Além disso, em virtude da manipulação das informações estar sendo feita sobre arquivos XML, qualquer tipo de modificação realizada pode ser facilmente implementada na classe que manipula esses dados, facilitando a integração e o aperfeiçoamento do sistema como um todo.

CAPÍTULO 5

RESULTADOS

Neste capítulo serão analisados os resultados obtidos a partir da implementação das propostas de detecção do sistema. Para isso, em um primeiro momento será mostrado como se deu a coleta dos dados de treinamento e validação das redes especialistas. Ao final do capítulo as melhores estruturas neurais serão selecionadas para que se possa realizar uma análise um pouco mais detalhada.

5.1 Coleta dos dados

Tanto para o processo de identificação quanto para o processo de detecção, o primeiro passo a ser dado é a obtenção das amostras experimentais para o treinamento supervisionado das redes neurais de identificação do modelo e de detecção e diagnóstico das falhas.

Dessa maneira, realizou-se a coleta dos dados a partir da estimulação do sistema simulado através da aplicação de sinais binários pseudo aleatórios (*Pseudo Random Binary Signals* – PRBS) à referência de cada um dos tanques e aos parâmetros do sistema que simulam as falhas. Para a identificação, a faixa de valores aplicados se deu do nível mínimo (zero) ao máximo (trinta). Já para a detecção das falhas, os valores foram aplicados conforme Tab. 5.1. Nessa tabela, os valores gerados no intervalo determinado pelos mínimos e máximos de cada parâmetro eram multiplicados pelos respectivos valores padrão e aplicados ao modelo.

Perceba que nas falhas dos atuadores, as tensões a serem aplicadas podem vir a danificar a bomba. Por esse motivo não seria viável obter as amostras do processo real, mas sim a partir de uma simulação.

Os sinais pseudo aleatórios gerados se mantiveram dentro dos limites estabelecidos durante todo o tempo da simulação. Para o processo de identificação foram obtidas 6000 (seis mil) amostras, equivalentes à 10 (dez) minutos de simulação. Já para a detecção o processo foi simulado durante 20 (vinte) minutos, o que correspondeu à obtenção de

Tabela 5.1: Valores aplicados para o treinamento das redes neurais de detecção.

Falha	**Valor padrão**	**Mínimo**	**Máximo**	**Representatividade**
FSeDG	0,16*	0,8	1,2	Até ±6 cm
FSeDO	1,0	-3,0	3,0	Até ±3 cm
FSeSR	1,0	-0,03	0,03	Até ±9 cm
FSeQ	1,0	0,0	0,0	–
FADG	1,0	0,8	1,0	Até -3 Volts
FADO	1,0	-1,0	0,0	Até -1 Volts
FASR	1,0	-0,03	0,03	Até ±0,45 Volts
FAVK	K_m	0,7	1,1	–
FAQ	1,0	0,0	0,0	–
FSiVzT	$a_{i_{\text{MED}}}$	0,25	0,75	25 a 75% de $a_{i_{\text{MED}}}$
FSiVrOS	$a_{i_{\text{MED}}}$	0,75	1,25	±25% de $a_{i_{\text{MED}}}$
FSiVrGMP	5,0*	0,8	1,0	Até -3 Volts
FSiEOS	$a_{i_{\text{MED}}}$	0,0	0,5	–

* Estabelecido pelo manual do fabricante.

12000 (doze mil) amostras.

De posse dos valores obtidos, iniciou-se a fase de treinamento das RNAs. Todas as redes foram treinadas em modo *offline* com o *toolbox* de redes neurais do *software* matemático Matlab®, utilizando o algoritmo LMA. Ao final de cada etapa de treinamento as redes eram submetidas aos testes de validação com o intuito de avaliar suas capacidades de generalização.

5.2 Análise das RNAs

Para que fossem estabelecidos critérios avaliativos significantes, diversas redes neurais foram treinadas com diferentes configurações. Para cada uma dessas configurações, o número de neurônios na camada oculta e a ordem do modelo eram modificadas. O número de redes treinadas corresponde àquele especificado no final da Tab. 5.2.

Na primeira proposta de identificação foram treinadas 6 (seis) redes neurais a cada vez que o número de neurônios na camada oculta era alterado. Ou seja, para cada ordem eram treinadas 18 (dezoito) redes neurais. Como foram testadas três ordens distintas, tem-se um total de 54 (cinquenta e quatro) redes para a primeira proposta.

Observa-se entretanto que, na segunda proposta de identificação, existiam duas redes a serem treinadas em cada etapa de treinamento, conforme Fig. 4.6. Já para a segunda proposta de detecção, considera-se um conjunto de 13 (treze) redes especialistas, sendo uma para cada falha, conforme Fig. 4.9. Em virtude desses aspectos, o número de redes treinadas dobra da primeira para a segunda proposta de identificação e é multiplicado por um fator de treze para a segunda proposta de detecção.

Ainda na Tab. 5.2 pode-se observar que não há nenhuma rede treinada na primeira proposta de detecção. Tal fato se deu em virtude de nenhuma das estruturas neurais terem

Tabela 5.2: Número de redes neurais treinadas de acordo com a ordem do modelo e o número de neurônios na camada oculta.

Proposta	**Ordem**	**Neurônios na camada oculta**	**Número de redes treinadas**	**Total**
Identificação				
1	2	6, 8 e 10	6	54
	3	8, 12 e 16		
	4	10, 16 e 22		
2	2	2/6, 4/8 e 6/10	6	108
	3	4/8, 6/12 e 8/16		
	4	6/10, 8/16 e 10/22		
Detecção				
1	–	–	–	–
2	2	8/12/16	6/falha	702
	3	14/18/22		
	4	20/24/28		
			Total	864

convergido dentro de um limite de 50.000 (cinquenta mil) iterações. Vale ressaltar que, apesar de não ter sido obtido sucesso durante o treinamento, foram feitas diversas tentativas, com vários tipos de redes neurais de ordens distintas e de diferentes arquiteturas de interconexão neuronal.

A não convergência dessas redes se deu, provavelmente, em consequência da ausência de dados representativos no universo das variáveis disponíveis, obtidas direta ou indiretamente a partir do processo. No caso do sistema de tanques acoplados, conforme descrito anteriormente, as seis variáveis disponíveis, mostradas na Fig. 4.8, deveriam identificar não somente as falhas propostas, mas também todas as combinações possíveis entre elas. Entretanto, nem mesmo quando somente as treze falhas propostas foram consideradas como saída, a rede chegou a convergir.

5.3 Melhores redes

Diante do grande número de redes neurais a serem analisadas, foi necessário estabelecer algum tipo de critério avaliativo para que as melhores redes fossem selecionadas. Para as propostas de detecção utilizou-se o EMQ sobre os valores de saída $\widehat{L_1}$ e $\widehat{L_2}$, enquanto que para as redes de detecção fez-se uso do número de erros de tipo I e erros de tipo II. O EMQ e o número de erros de tipo I e II de cada rede foram obtidos a partir da média dos valores de três validações distintas.

5.3.1 Redes para a identificação do modelo

Utilizando o EMQ estabelecido pela Eq. 2.6, foram obtidos os resultados exibidos na Tab. 5.3.

Tabela 5.3: Melhores redes treinadas para a identificação do modelo.

Proposta	**Ordem**	**NCO***	**Treinamento**	**EMQ $\widehat{L_1}$**	**EMQ $\widehat{L_2}$**	**EMQ**
1	2	8	2	4,39e-7	3,29e-6	3,73e-6
	3	12	5	1,38e-5	1,46e-5	2,84e-5
	4	22	4	1,40e-6	2,60e-6	4,01e-6
2	2	2/6	2	5,06e-6	7,26e-6	1,23e-5
	3	6/12	1	5,48e-6	1,81e-7	5,66e-6
	4	10/22	3	5,12e-6	9,46e-6	1,45e-5

* Número de neurônios na camada oculta.

Assim, a melhor rede de identificação, escolhida para estimar os valores de L_1 e L_2, foi a rede de segunda ordem, com oito neurônios na camada oculta, obtida no segundo treinamento. Essa rede, cuja soma dos EMQs foi de $3{,}73 \times 10^{-6}$, será utilizada para gerar os valores dos resíduos e_i que compõem a entrada das redes de detecção.

5.3.2 Redes para a detecção/diagnóstico de falhas

Os erros de tipo I, também conhecidos como *falsos positivos*, ocorrem quando o sistema reconhece uma falha que, na verdade, não houve, enquanto que os erros de tipo II, também conhecidos como *falsos negativos*, ocorrem quando o sistema deixa de reconhecer uma falha que aconteceu.

No sistema proposto, a escolha das melhores redes de detecção se deu a partir do número de incidência de cada um desses erros. Assim, estabeleceu-se que a melhor rede para uma determinada falha seria aquela em que a soma dos erros de tipo I com os erros de tipo II fosse a menor possível.

Os resultados obtidos para cada uma das falhas podem ser observados na Tab. 5.4. Os valores decimais estão presentes em virtude das colunas dos erros se referirem a média obtida a partir das três validações. A última coluna representa o percentual de erro médio total[1] sobre todas as amostras. Assim como na proposta de identificação, as melhores redes estão destacadas em cinza.

Deve-se observar ainda que apesar de o treinamento ter sido feito com 12000 (doze mil) amostras, a rede deve acertar 24000 (vinte e quatro mil) vezes, pois para cada ponto a rede terá que exibir se está ou não havendo aquela falha em T_1 e em T_2. Portanto, para cada ponto de treinamento haverá dois acertos a serem realizados.

Analisando os valores exibidos nessas tabelas, percebe-se que as redes especialistas possuem uma certa dificuldade em identificar a maioria das falhas dos atuadores e algumas falhas do sistema, chegando, em alguns casos, a ter mais de 43% de erro nos pontos de validação.

[1] O percentual de erro médio total é obtido a partir da soma da quantidade de erros de tipo I com os erros de tipo II, dividido pelo número total de amostras.

Tabela 5.4: Melhores redes treinadas para a detecção de falhas.

Falha	Ordem	NCO	Treinamento	Acertos	Erros de tipo I	Erros de tipo II	Total de erros
FSeDG	2	16	1	23430	134	436	2,37%
	3	22	6	23456	176	368	2,26%
	4	28	2	23492	203	305	2,12%
FSeDO	2	16	3	23872	4	124	0,53%
	3	14	6	23876	24	100	0,51%
	4	28	5	23890	9	101	0,46%
FSeSR	2	8	2	23306	201	493	2,89%
	3	14	5	23314	419	267	2,86%
	4	20	3	23317	325	358	2,84%
FSeQ	2	12	5	23972	21	7	0,12%
	3	18	4	23993	1	6	0,03%
	4	20	4	23994	1	5	0,02%
FADG	2	8	3	20710	1627	1663	13,7%
	3	14	6	20465	1734	1801	14,73%
	4	20	2	20116	2076	1808	16,18%
FADO	2	12	2	22828	612	560	4,88%
	3	22	6	22903	607	490	4,57%
	4	28	3	23075	636	289	3,85%
FASR	2	8	6	14153	3407	6440	41,03%
	3	14	6	13977	4064	5959	41,76%
	4	20	3	13578	4394	6028	43,42%
FAVK	2	8	5	20765	1551	1684	13,48%
	3	14	5	20702	1336	1962	13,74%
	4	20	4	20489	1641	1870	14,63%
FAQ	2	12	6	23976	1	23	0,1%
	3	18	2	23979	3	18	0,086%
	4	28	6	23980	2	18	0,083%
FSiVzT	2	16	6	23623	174	203	1,57%
	3	22	2	23742	95	163	1,07%
	4	24	1	23774	74	152	0,94%
FSiVrOS	2	8	3	22465	437	1098	6,39%
	3	14	4	22274	764	962	7,19%
	4	20	3	21781	1165	1054	9,24%
FSiVrGMP	2	16	1	20629	1863	1508	14,04%
	3	18	3	20847	2135	1018	13,14%
	4	20	2	20906	1980	1114	12,89%
FSiEOS	2	12	4	23996	1	3	0,018%
	3	14	5	23995	2	3	0,021%
	4	20	2	23995	2	3	0,021%

5.4 Detecções

Tendo selecionado a melhor rede de identificação e as melhores redes de detecção, pôde-se compor o sistema final, conforme Fig. 4.10. Com o intuito de avaliar o desempenho desse sistema foram realizadas treze simulações, variando cada um dos parâmetros das possíveis falhas individualmente.

Cada simulação teve uma duração de um 1 (um) minuto e 45 (quarenta e cinco) segundos, tendo sido dividida em 7 (sete) intervalos de 15 (quinze) segundos, de acordo com a Fig. 5.1. As referências utilizadas para os tanques foram fixadas em 15 (quinze) e 20 (vinte) centímetros para T_1 e T_2, respectivamente. Os controladores foram configurados para atuarem como PIDs, cujos ganhos também foram mantidos fixos, com valores $K_P = 2{,}0$, $K_I = 0{,}5$ e $K_D = 0{,}05$.

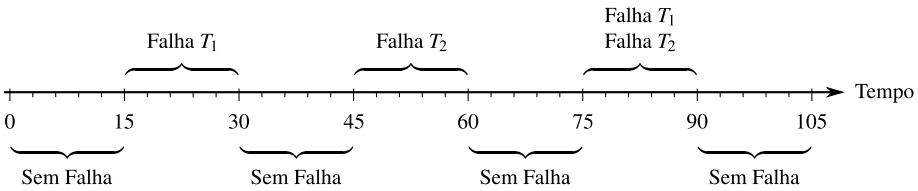


Figura 5.1: Intervalos de simulação do sistema proposto.

Ao contrário do que aconteceu durante o período de treinamento e validação das redes, os valores de cada um dos parâmetros das falhas foram mantidos fixos durante os intervalos em que aquela falha estava agindo sobre o sistema. Os valores utilizados em cada uma das treze simulações podem ser observados na Tab. 5.5.

Tabela 5.5: Valores dos parâmetros modificados para a simulação das falhas.

Simulação	**Falha**	**Valor modificado**
1	FSeDG	Ganho $= 0{,}128$
2	FSeDO	-2,0 cm
3	FSeSR	$\pm 2\%$
4	FSeQ	Ganho $= 0{,}0$
5	FADG	Ganho $= 0{,}8$
6	FADO	-0,5 Volts
7	FASR	$\pm 2\%$
8	FAVK	$K_m = 3{,}45$
9	FAQ	Ganho $= 0{,}0$
10	FSiVzT	$a_{VZ} = \frac{a_{MED}}{2}$
11	FSiVrOS	$a_i = \frac{a_{MED}}{2}$
12	FSiVrGMP	Ganho $= 4{,}5$
13	FSiEOS	$a_i{}' = \frac{a_{MED}}{4}$

Os resultados obtidos para cada uma das falhas podem ser vistos da Fig. 5.2 a 5.14. Nessas figuras, as áreas hachuradas em vermelho representam os intervalos em que o

sistema detectou a falha em T_1, enquanto que as falhas hachuradas em azul representam os intervalos em que a falha foi detectada em T_2.

A primeira falha a ser simulada foi a FSeDG, cujo resultado da detecção pode ser observado na Fig. 5.2. Nessa simulação o valor do ganho do sensor foi reduzido para 80% do valor original.

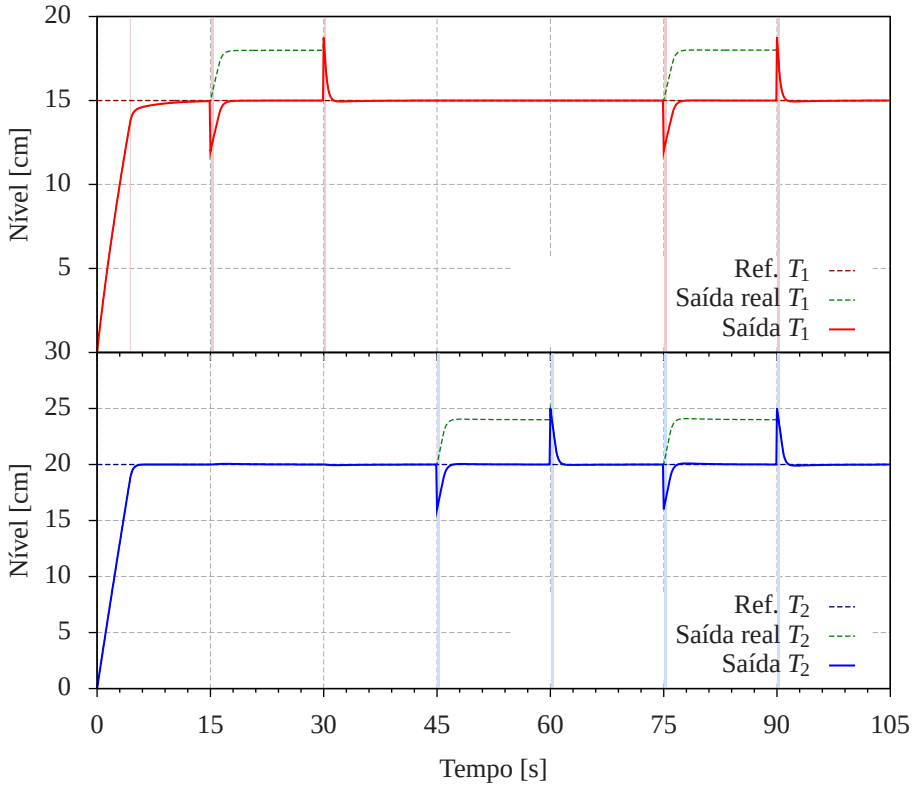


Figura 5.2: Simulação da FSeDG com o ganho reduzido a 80% do valor original.

Observa-se nessa figura que o sistema consegue identificar a presença da falha assim que o valor do parâmetro é modificado. Após esse período, os controladores conseguem "compensar" a falha, enviando mais tensão para a bomba e, consequentemente, fazendo com que o nível retorne à referência. Observa-se, entretanto, que essa "compensação" se dá de maneira inadequada, tendo em vista que a leitura do nível estará com um erro de 20%.

Vale destacar que há uma nítida diferença entre a curva denominada "Saída T_i", a qual representa a saída efetiva do sensor do tanque i, e a curva da saída real do sistema, denominada "Saída real T_i". Essa diferença deverá ser implicitamente interpretada nas demais falhas do sensor.

Assim sendo, quando o valor lido pelos sensores for de 24 cm, o tanque estará prestes a transbordar, atingindo, na verdade, o limite superior de 30 cm. Em uma aplicação acadêmica tal situação pode não representar nenhum risco aos equipamentos além daqueles em que a água venha a causar. Por outro lado, em aplicações críticas, essa "compensação" pode vir a causar diversos prejuízos.

O sistema se comportou de maneira semelhante a essa na FSeDO e na FSiVzT, conforme observado nas Figs. 5.3 e 5.4. Essa semelhança pode vir a dificultar a identificação

dessas falhas quando as redes estiverem agindo paralelamente.

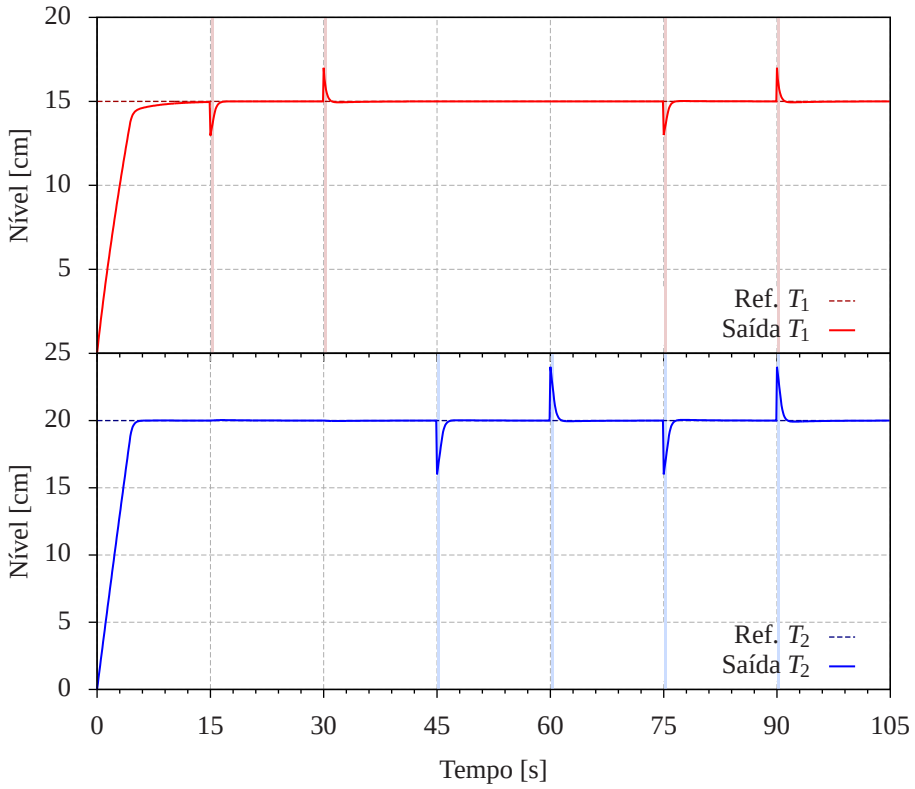


Figura 5.3: Simulação da FSeDO com *offset* de -2 cm.

Os resultados obtidos para essas falhas não estão condizentes com a Tab. 5.4. Tal situação pode estar acontecendo porque as redes identificaram uma dinâmica para variações rápidas através de sinais de excitação pseudo aleatórios, não conseguindo identificar variações bruscas e continuadas.

Assim, uma possível alternativa para contornar esse problema, seria a de se utilizar *flags* binárias, ativadas no momento em que fosse detectada a primeira variação e desativadas na detecção seguinte. Essas *flags* indicariam para o sistema que a falha está agindo durante todo o intervalo de tempo em que estivessem ativas.

Uma outra simulação mostrou que a FSeSR foi facilmente identificada pelo sistema, como pode ser visto na Fig. 5.5. Observa-se entretanto que, por se tratar de um ruído com distribuição uniforme ($\pm 2\%$), o sistema não consegue identificar a falha em alguns pontos. Nesses pontos, pode-se perceber que o valor gerado pela função `rand` mantém o sinal próximo à referência.

Ao contrário da FSeSR, a simulação realizada para a FASR não foi tão facilmente identificada, como pode ser visto na Fig. 5.6. Os resultados obtidos para T_1 podem até ser considerados razoáveis, enquanto que os resultados para T_2 são visivelmente inaceitáveis, tendo em vista que nenhum dos pontos em que a falha deveria ter sido identificada foram reconhecidos pela rede. Apesar disso, observa-se que os resultados estão condizentes com a Tab. 5.4, em que o total de erros de detecção supera 40%.

Com exceção da FSiVrGMP, todas as demais falhas restantes também foram facilmente identificadas pelo sistema, como pode ser observado nas Figs. 5.8 a 5.14. No

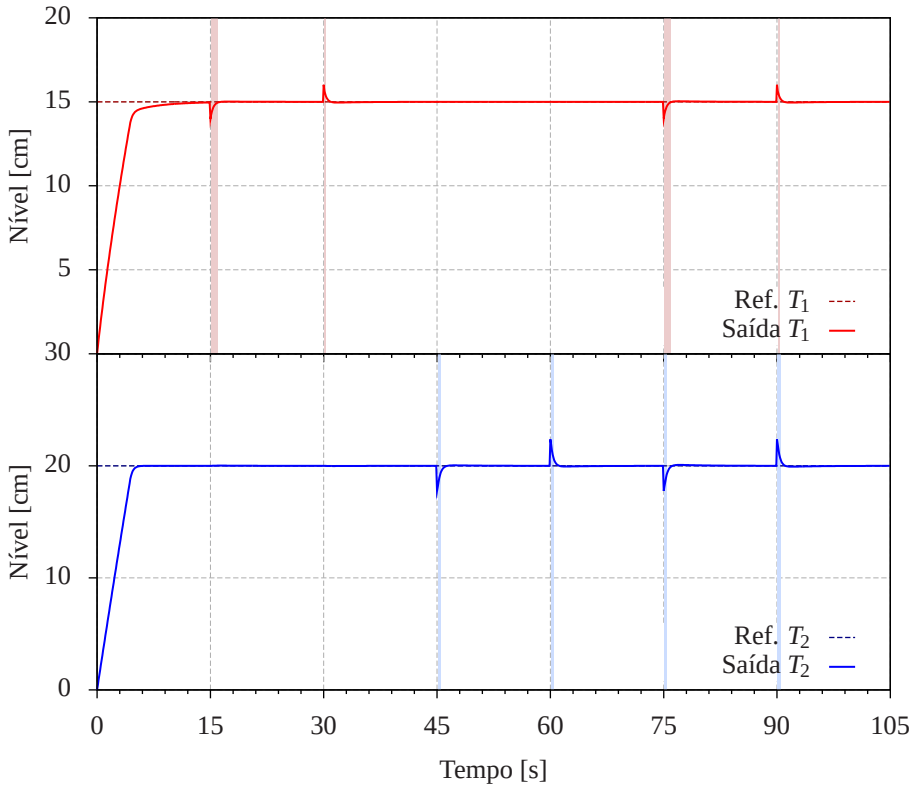


Figura 5.4: Simulação da FSiVzT com $a_{VZ} = a_{MED}/2$.

caso da FSiVrGMP, o sistema não consegue detectar quando a falha acontece somente em T_1, acusando falha também em T_2, como pode ser visto no intervalo de 15 (quinze) a 30 (trinta) segundos da Fig. 5.7. Todavia, as falhas em T_1 são identificadas corretamente. Tal fato justifica a quantidade de erros expostos pela Tab. 5.4.

Uma última observação pode ser feita a respeito da Fig. 5.8, em que há a detecção da falha em T_1 em um pequeno intervalo no final da simulação. Contudo, isso não veio a comprometer o funcionamento do sistema nos intervalos em que a falha deveria realmente ser detectada. Trata-se, portanto, de um falso positivo.

Ressalta-se por fim que as situações adversas podem ter ocorrido no sistema em virtude dos testes terem sido realizados com valores constantes para a variação dos parâmetros. Por esse motivo os resultados obtidos nem sempre estão de acordo com os valores da Tab. 5.4.

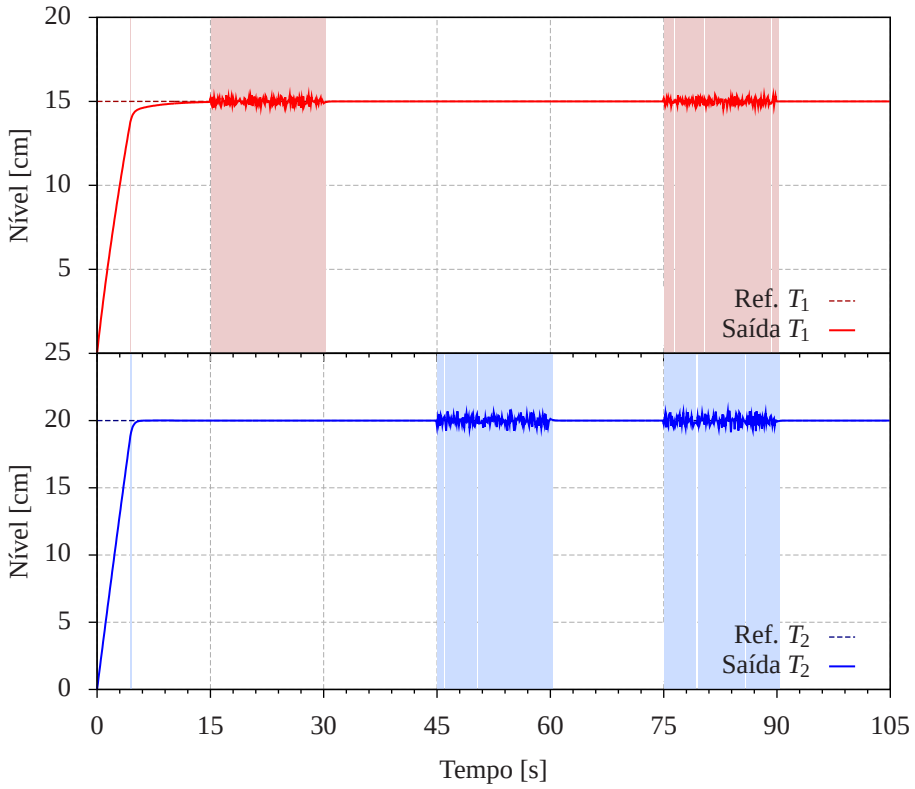


Figura 5.5: Simulação da FSeSR com ruído de distribuição uniforme (±2%).

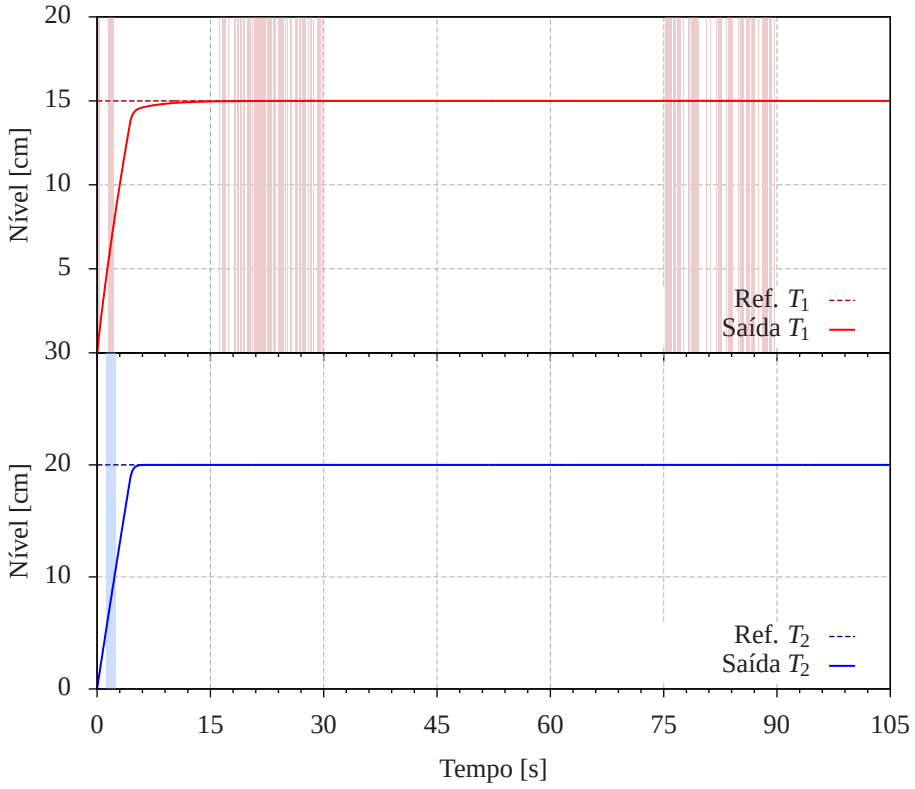


Figura 5.6: Simulação da FASR com ruído de distribuição uniforme (±2%).

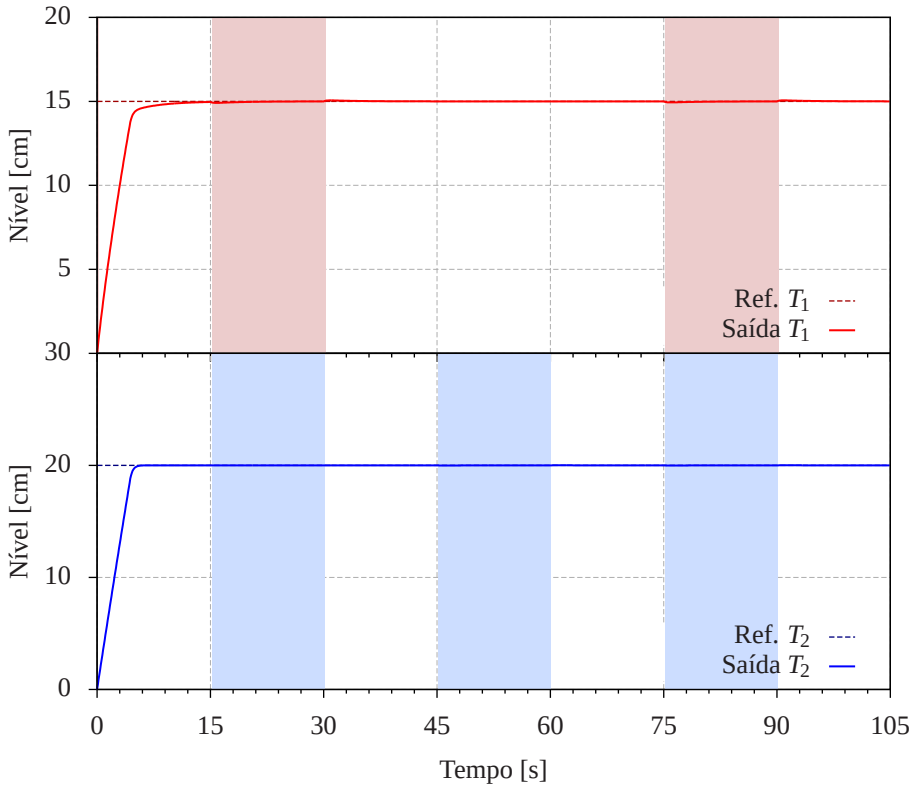


Figura 5.7: Simulação da FSiVrGMP com o ganho reduzido a 90% do valor original.

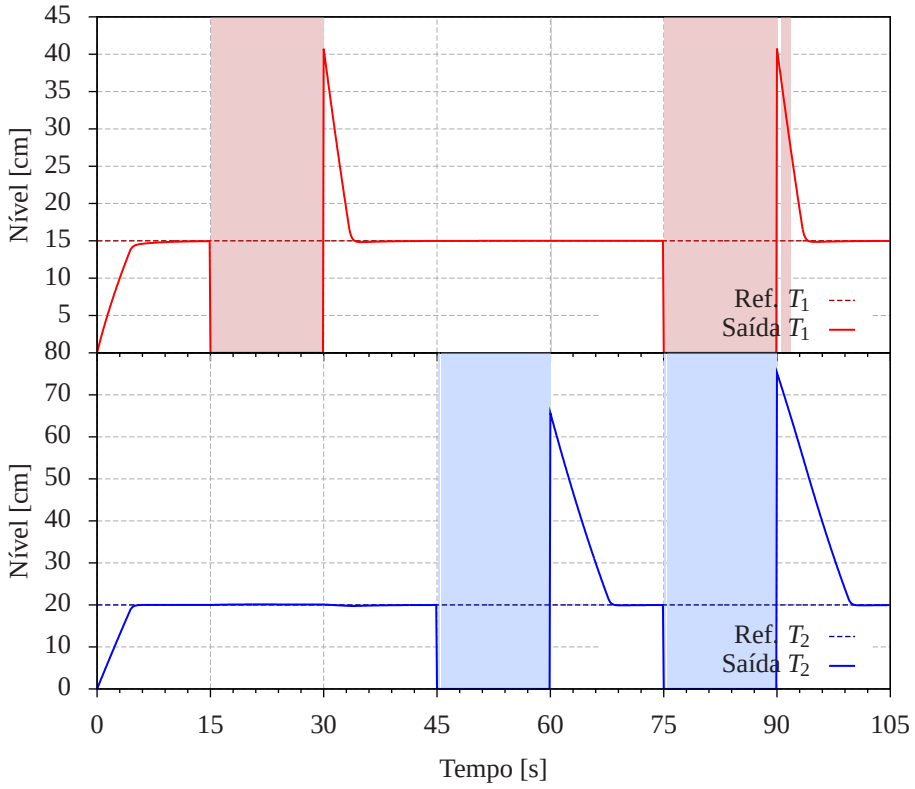


Figura 5.8: Simulação da FSeQ (Ganho = 0).

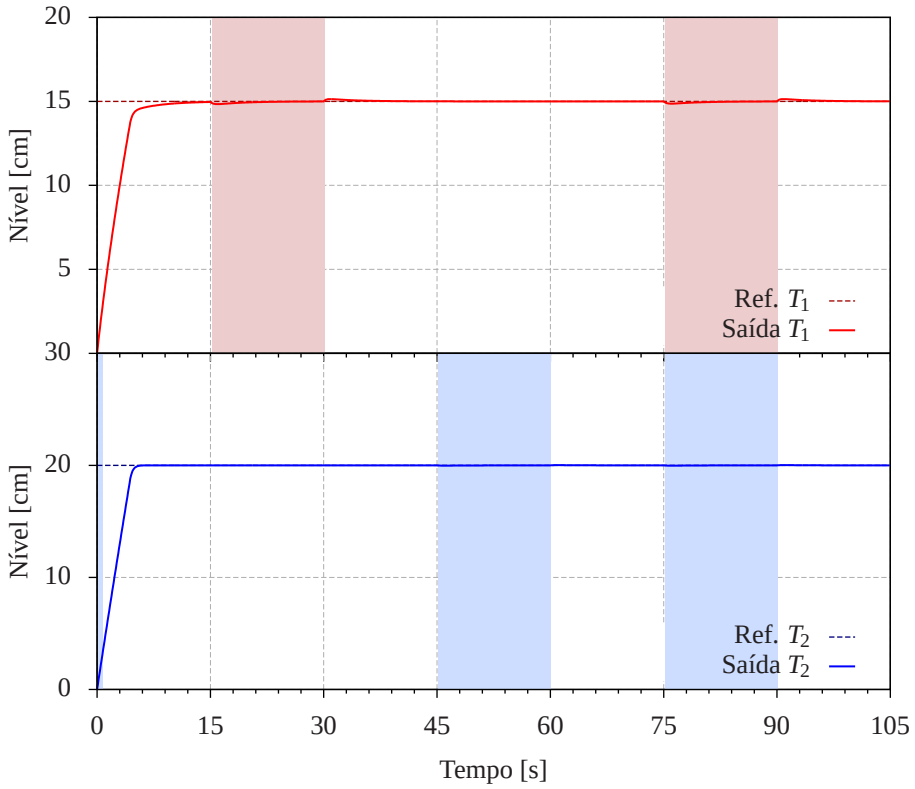


Figura 5.9: Simulação da FADG com o ganho reduzido a 80% do valor original.

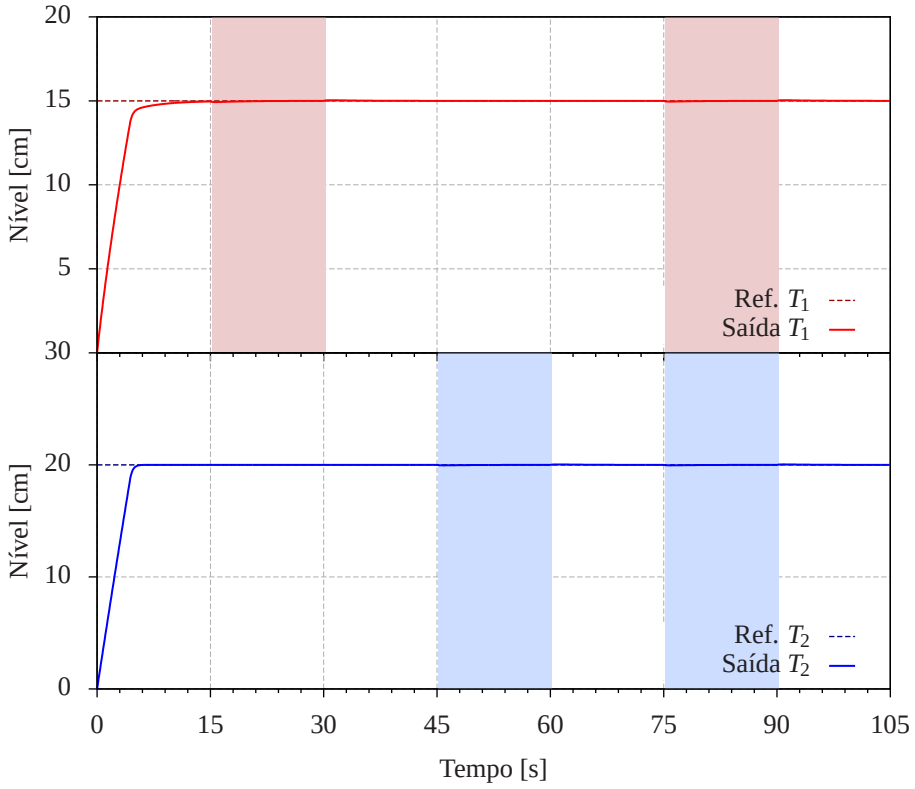


Figura 5.10: Simulação da FADO com *offset* de -0,5 Volts.

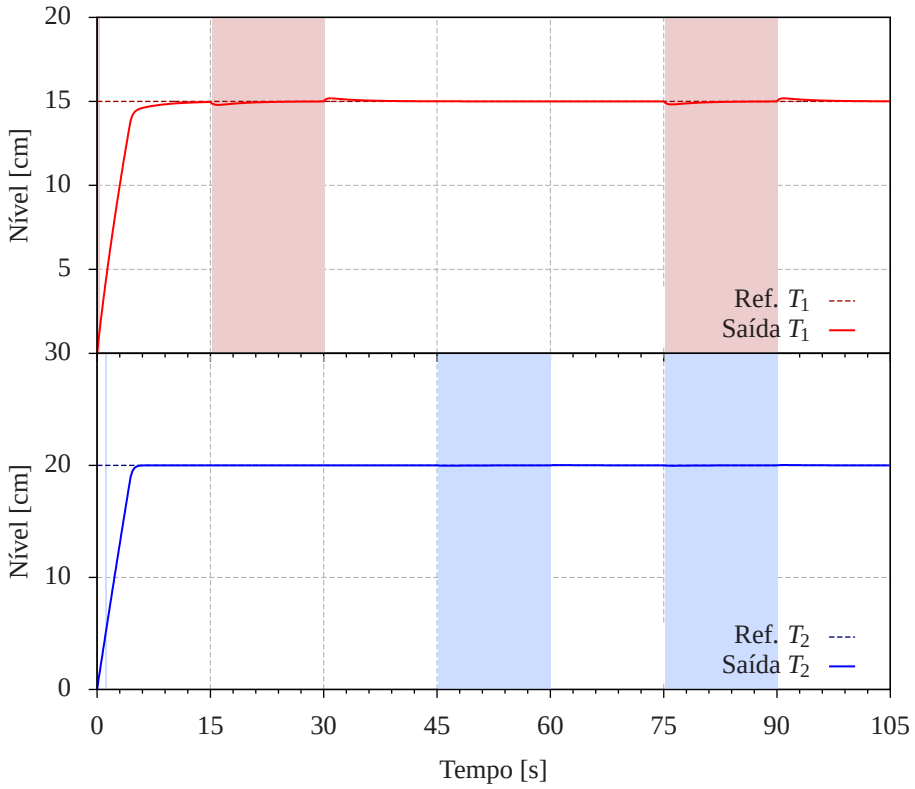


Figura 5.11: Simulação da FAVK com K_m reduzido a 75% do valor original.

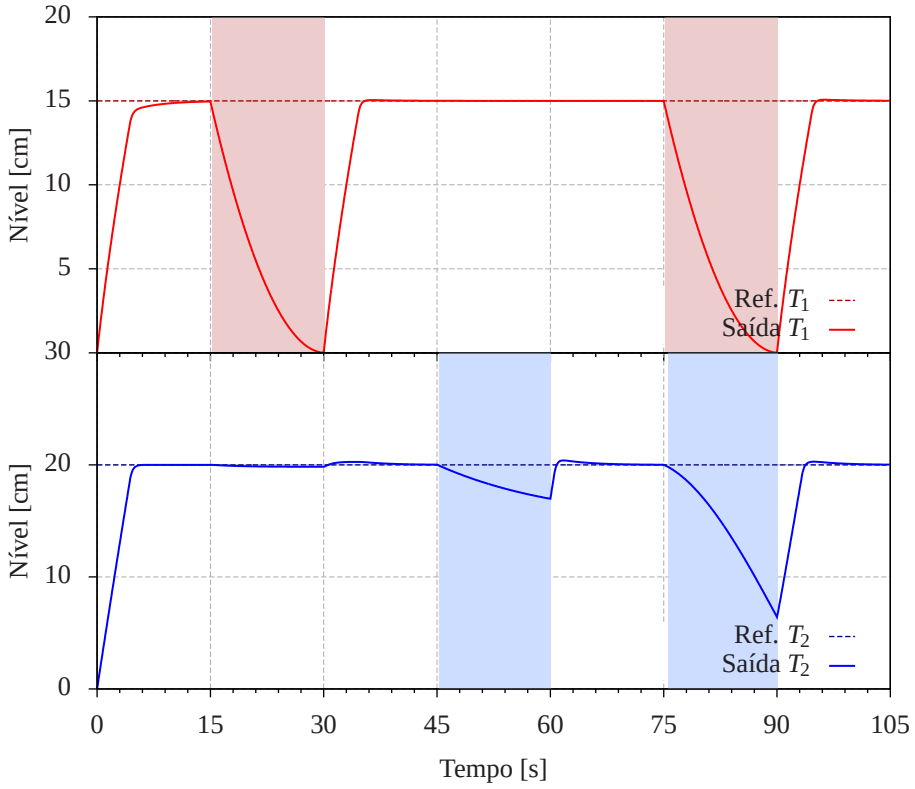


Figura 5.12: Simulação da FAQ (Ganho = 0).

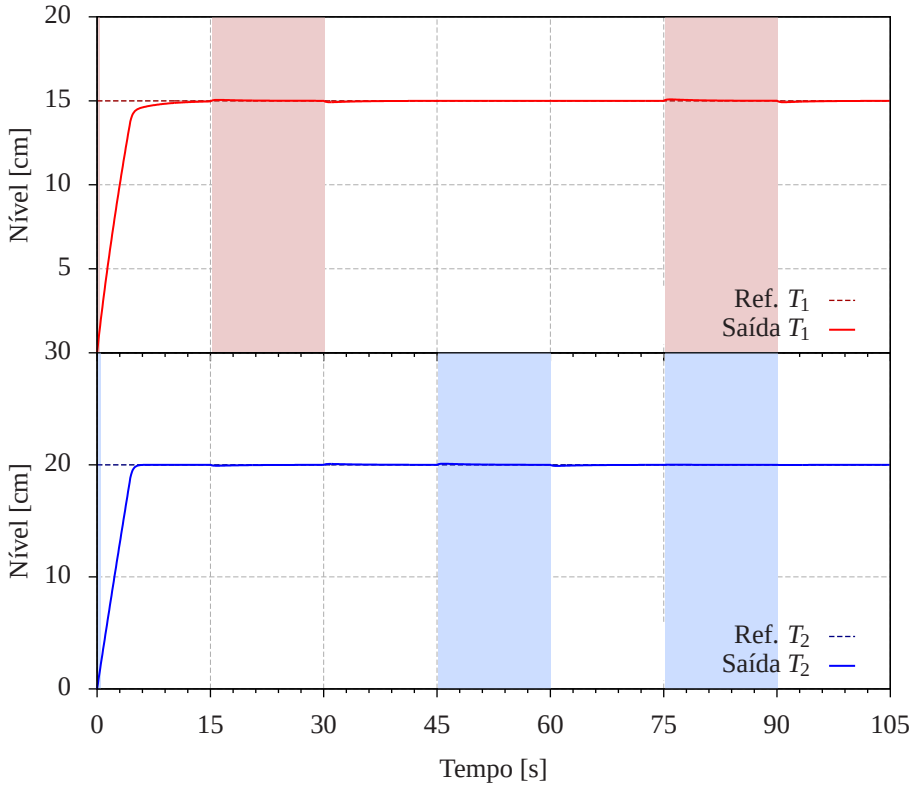


Figura 5.13: Simulação da FSiVrOS com $a_i = a_{MED}/2$.

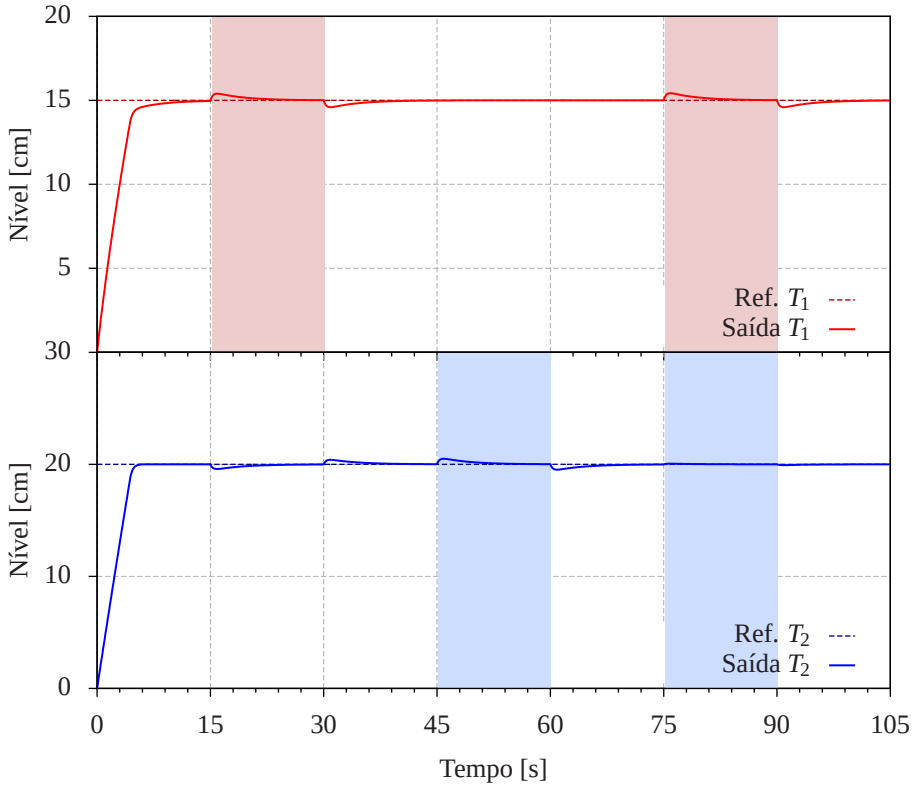


Figura 5.14: Simulação da FSiEOS com $a_i = a_{MED}/4$.

CAPÍTULO 6

CONCLUSÕES

6.1 Conclusões

O presente trabalho foi desenvolvido com o intuito de fornecer um sistema DDF para um sistema de tanques acoplados. Para isso, o sistema fez uso de uma estrutura neural que, a partir dos valores disponíveis, indicava ao usuário os momentos em que as falhas estavam ocorrendo.

O sistema proposto foi constituído por um módulo de identificação, o qual realizava a inferência dos níveis dos tanques, e um conjunto de redes especialistas, capazes de identificar cada uma das treze falhas listadas.

Dentre essas falhas, oito foram identificadas com facilidade e outras três tiveram um desempenho satisfatório, com um pequeno problema de detecção que pode ser facilmente resolvido através de *flags* binárias. As outras duas falhas não foram identificadas corretamente, mas, em ambos os casos, o sistema consegue detectar a falha corretamente para T_1.

Observa-se ainda que os resultados obtidos podem melhorar ainda mais quando forem utilizados valores reais, oscilando dentro da faixa de valores em que a rede foi treinada. Tal situação pode fazer com que não mais ocorra o problema de detecção das FSeDG, FSeDO e FSiVzT, evitando portanto a utilização de *flags* para contorná-lo.

Assim, pode-se dizer que o sistema teve um desempenho satisfatório, conseguindo identificar cerca de 85% das falhas propostas. Comprova-se, portanto, que as redes neurais de múltiplas camadas são estruturas eficientes tanto para a identificação de modelos quanto para a detecção e o diagnóstico de falhas.

Em contraponto, vale salientar que não foram realizados testes que avaliassem o desempenho das redes de detecção atuando de maneira simultânea ou paralelamente. Essa situação pode vir a gerar falsos positivos uma vez que as variáveis de entrada das redes são idênticas. Outro aspecto que reforça essa teoria é que a rede da primeira proposta de detecção não convergiu. Conforme dito anteriormente, a não convergência das redes

dessa proposta se deu, provavelmente, em consequência da ausência de dados representativos no universo das variáveis disponíveis, obtidas direta ou indiretamente a partir do processo.

6.2 Perspectivas

Após obter os resultados, com as falhas agindo de maneira individual sobre cada uma das redes especialistas, o passo seguinte a ser dado envolve a aplicação de sinais de excitação em todas as redes especialistas simultaneamente. Dessa forma, a partir das saídas ativas de cada uma das redes, será possível identificar as falhas que possuem um comportamento semelhante.

Uma outra comparação pode ser realizada a partir da modificação das estruturas neurais, utilizando novos tipos de rede, tais como as redes de função de base radial, máquinas de vetor de suporte ou quaisquer outros modelos neurais que possam ser adaptados ao problema de detecção e diagnóstico de falhas.

Tendo sido selecionada a melhor estrutura de detecção, o ambiente de simulação desenvolvido em C++ poderá ser incrementado, adicionando a possibilidade da simulação das falhas em tempo real. Para isso, faz-se necessário que as estruturas neurais sejam adicionadas ao sistema e que sejam estabelecidas novas formas de entrada dos parâmetros a serem modificados. Além disso, o sistema poderá gerar sinais de saída de tal forma que as detecções das falhas sejam observadas a partir de um sistema de monitoramento e supervisão externo.

Com relação ao *Simddef*, apesar de os testes terem sido realizados apenas com estruturas neurais, nada impede que novas técnicas venham a ser implementadas, tais como técnicas *Fuzzy*, métodos estatísticos, novos tipos de estruturas neurais, métodos de detecção por verificação de limites, técnicas que utilizam observadores de estado, dentre outras. Além disso, a utilização de arquivos de configuração com sintaxe XML facilita a integração desses novos módulos ao sistema. Todo o código desenvolvido está disponível para *download* em `http://code.google.com/p/proeng-ufrn` para todos os participantes do projeto Pró-Engenharias.

Uma vez que todo o sistema tenha sido testado com vários sinais de excitação e diferentes tipos de módulos de detecção, pode-se agregar suas características à um SCTF. Nesse caso os sinais gerados pelo sistema de DDF servirão como “alarmes”. O SCTF, por sua vez, poderá realizar a reconfiguração dos controladores, modificando os valores dos ganhos ou até mesmo suas estruturas, de tal forma que o sistema continue funcionando de maneira correta até que a falha tenha sido corrigida.

APÊNDICE A

ARQUIVOS XML DO *Simddef*

A.1 Arquivos XML

XML (*eXtensible Markup Language*) é uma linguagem de marcação simples e altamente flexível, criada pela W3C e derivada da *Standard Generalized Markup Language* (SGML – ISO 8879). Essa linguagem foi idealizada para gerar linguagens de marcação para necessidades especiais e sua principal característica é que as linguagens desconhecidas ou de pouco uso podem ser facilmente definidas sem que haja necessidade de serem submetidas aos comitês de padronização.

A formatação dos arquivos XML permite que os dados sejam organizados de forma hierárquica, possibilitando que sejam criados arquivos para sua validação. Além disso, o formato não depende de plataformas de *hardware* ou de *software*, de tal forma que um banco de dados pode, através de uma aplicação, escrever ou ler um arquivo desse tipo sem maiores complicações.

Por definição, um documento XML é composto por um conjunto de caracteres que, quando dispostos de forma adequada, compõem uma árvore organizacional dos dados. A estrutura gerada possui, dentre outros, conjuntos de *tags*, *elementos* e *atributos*.

As *tags*, nada mais são do que as marcações que iniciam com < e terminam com >, podendo ser divididas em *start-tags*, como por exemplo `<secao>`, *end-tags*, como por exemplo `</secao>` e *empty-element-tags*, como por exemplo `<secao />`.

Os *elementos*, por sua vez, são as marcações dos componentes lógicos do documento, que iniciam com uma *start-tag* e encerram com uma *end-tag*, podendo ainda serem compostos apenas por uma *empty-element-tag*. Os caracteres entre as *tags*, quando existirem, constituem o conteúdo do elemento e podem conter outros tipos de marcação, inclusive outros elementos, dando origem aos *elementos filhos*. Um exemplo de elemento seria `<fala>Olá Mundo</fala>` ou apenas `<quebra_linha />`

Já os *atributos* são marcações compostas de um par *nome*/*valor*, que existem no contexto das *start-tags* ou das *empty-element-tags*. Como exemplo, no trecho

`<imagem end="foto.jpg" largura="100px" />`, existem dois atributos, denominados `end`, cujo valor é `foto.jpg`, e `largura`, cujo valor é `100px`.

A.2 Estrutura dos arquivos utilizados

A estrutura dos arquivos utilizados pelo *Simddef*, assim como todos arquivo XML tem início com uma declaração do tipo

```
<?xml version="1.0" encoding="UTF-8"?>
```

na qual a versão do XML utilizado e o tipo de codificação do arquivo podem ser modificados a critério da aplicação. Após a *tag* de inicialização do documento, a estrutura dos elementos dos arquivos varia de acordo com o recurso a ser utilizado dentro do sistema.

O primeiro elemento do arquivo é composto pela *tag* `Simddef`, a qual possui dois atributos, denominados `versao` e `tipo`:

```
<Simddef versao="1.0" tipo="falhas">
```

O primeiro atributo desta *tag* poderá ser utilizado, por exemplo, em futuras versões do sistema, para adequar a estrutura do arquivo que está sendo lido às novas versões disponíveis. Já o segundo atributo é utilizado para definir que tipo de arquivo está sendo carregado no sistema.

Nesta primeira versão, existem somente dois valores possíveis para o atributo `tipo`, são eles: `modulos` e `falhas`. Os elementos seguintes a serem lidos dependem diretamente do valor deste atributo. Assim, tendo conhecido esse valor, a classe que manipula os dados que estão sendo lidos poderá seguir um dos dois caminhos distintos. O término da leitura do arquivo encerra quando a *end-tag* `</Simddef>` for lida.

A.2.1 Arquivo de configuração de falhas

O arquivo de configuração de falhas possui a seguinte estrutura:

```
<?xml version="1.0" encoding="UTF-8"?>
<Simddef versao="1.0" tipo="falhas">
    <Falha>
        <local> ... </local>
        <abrv> ... </abrv>
        <descricao> ... </descricao>
    </Falha>
</Simddef>
```

O elemento `local` se refere ao local em que a falha poderá ocorrer, e pode ter três valores possíveis: `Sensor`, `Atuador` ou `Sistema`. Já o elemento `abrv` se refere a uma abreviatura estabelecida para aquela falha, a qual será utilizada em determinadas localizações do sistema para evitar que textos muito grandes apareçam na interface do usuário. Por fim, o elemento `descricao` se refere a uma descrição da falha. Este último elemento

poderá descrever alguns detalhes sobre como a falha ocorre e, possivelmente, algum outro aspecto importante relacionado à falha.

O elemento `Falha` poderá ser repetido tantas vezes quantas forem o número de falhas cadastradas no sistema, desde que cada falha seja especificada a partir dos três elementos citados anteriormente.

A.2.2 Arquivo de configuração de módulos

O arquivo de configuração dos módulos possui estrutura semelhante ao arquivo de falhas. Contudo, sabe-se que poderão ser implementados diversos tipos de módulos e cada um deles deverá estar associado à uma falha. Essa característica faz com que possam existir *tags* diferentes no arquivo, dependendo do tipo de módulo que está sendo carregado. Para um módulo neural, por exemplo, a estrutura do arquivo poderia ser:

```
<?xml version="1.0" encoding="UTF-8"?>
<Simddef versao="1.0" tipo="modulos">
    <Modulo>
      <tipo nome="RNA">
          <ordem>...</ordem>
      </tipo>
      <falha>...</falha>
      <arquivos qtde="3">
          <arq end="..."/>
          <arq end="..."/>
          <arq end="..."/>
      </arquivos>
   </Modulo>
</Simddef>
```

Nesse caso, o elemento `tipo` se refere ao tipo de módulo que está sendo configurado, podendo ter como valor, por exemplo, `RNA`, `Fuzzy`, `Estatístico`, ou ainda qualquer outro nome dentre os módulos que venham a ser implementados. O elemento `ordem`, se refere a ordem da rede neural, a qual determina o número de regressores que serão utilizados nas entradas e saídas da rede. O elemento `falha` associa o módulo à uma falha previamente carregada a partir de sua abreviatura. O elemento `arquivos` possui o atributo `qtde`, que indica o número de arquivos que serão carregados para a configuração do módulo. No caso de uma rede neural, os arquivos poderão, por exemplo, conter os pesos sinápticos e as funções de ativação de cada uma das camadas. As *tags* `arq`, com seu atributo `end`, indicam o endereço do arquivo que será carregado. O valor do atributo `qtde` deverá ser condizente com o número de *tags* `arq` do módulo.

Assim como no arquivo de falhas, o elemento `Modulo` pode se repetir tantas vezes quantos forem o número de módulos cadastrados no sistema. Dependendo do tipo de módulo, outras *tags* poderão ser adicionadas. A manipulação de cada uma dessas *tags* poderá ser implementada na classe de manipulação dos arquivos XML.

É importante ressaltar também que cada módulo deverá ser implementado pelo usuário como uma classe que herda métodos e atributos de uma classe abstrata, de acordo com a interface abaixo. Os métodos virtuais puros são os únicos utilizados pelo sistema para

processar as informações e exibir as falhas detectadas ao usuário.

```
#ifndef MODULO_H_
#define MODULO_H_

#include <QHash>
#include <QString>
#include <QStringList>

#include <Matrix.h>
using Flood::Matrix;

typedef Matrix< double > MatrizD;
typedef Matrix< int > MatrizI;

class Modulo
{
    // Atributos publicos
    public:
        enum TipoModulo{ RNA = 0, Fuzzy, Estatistico, Personalizado };

    // Metodos publicos
    public:
        Modulo( const Modulo::TipoModulo &tipo = Modulo::RNA );
        ~Modulo();

        bool ativo();

        Modulo::TipoModulo tipo();

        QList< MatrizD > dados();

        QString nome_falha();

        QStringList endereco_arquivos();

        virtual QHash< int, QString > curvas_a_exibir() = 0;
        virtual QHash< QString, MatrizI > deteccoes_falhas();

        virtual QString nome_tipo() = 0;

        virtual void ler_arquivos() = 0;

        void configurar_arquivos( const QStringList & );
        void configurar_ativo( const bool & );
        void configurar_falha( const QString & );

    // Metodos protegidos
    protected:
        virtual void processar_saida() = 0;

```

```
50      // Atributos
51      protected:
52          bool ativ;
53
54          MatrizD entrada;
55          MatrizD saida;
56
57          QHash< QString, MatrizI > deteccoes;
58
59          QString falha;
60
61          QStringList arquivos;
62
63          TipoModulo tipo_modulo;
64
65          uint n_entradas;
66          uint n_saidas;
67  };
68
69  #endif
```

Dentre os métodos a serem implementados, o método `curvas_a_exibir` não possui nenhuma entrada e produz como saída uma *hash* que mapeia o número da curva a ser exibida com seu respectivo nome. Esta função tem como principal objetivo fornecer ao sistema uma maneira simples e direta de inserir os valores desejados no gráfico e associar as curvas inseridas à legenda.

O método `nome_tipo` também não possui entrada e fornece como saída o nome do tipo do módulo que foi carregado. Este nome é utilizado na janela de configuração dos módulos e na janela principal do sistema.

O método `ler_arquivos` assume que todos os arquivos foram configurados durante a criação do módulo, na leitura do arquivo XML, de tal maneira que nenhuma entrada é necessária. Este método serve para fazer a leitura dos arquivos e configurar o módulo de acordo com o seu tipo. No caso dos módulos neurais, por exemplo, é este método quem cria a rede, atribui os pesos sinápticos e configura as funções de ativação de cada camada. Como o intuito do método é de apenas ler os arquivos e efetuar a configuração dos módulos, este método não produz nenhuma saída.

Por fim, o método `processar_saida` é quem efetivamente realiza o processamento de todos os dados que foram carregados. Este método não é chamado a partir do sistema, devendo ser utilizado dentro de alguma rotina interna da classe que está sendo implementada como, por exemplo, ao final do método `ler_arquivos`.

A.3 Exemplos

Nas seções seguintes poderão ser visualizados exemplos completos dos arquivos de configuração de falhas e módulos. As quebras de linha que existem em alguns dos elementos `descricao` foram ali colocadas apenas para evitar que a formatação do código neste documento excedesse as margens de impressão.

A.3.1 Arquivo de configuração de falhas

```
<?xml version="1.0" encoding="UTF-8"?>
<Simddef versao="1.0" tipo="falhas">
    <Falha>
        <local>Atuador</local>
        <abrv>FADG</abrv>
        <descricao>Falha do atuador por descalibramento de ganho</descricao>
    </Falha>
    <Falha>
        <local>Atuador</local>
        <abrv>FADO</abrv>
        <descricao>Falha do atuador por descalibramento de offset</descricao>
    </Falha>
    <Falha>
        <local>Atuador</local>
        <abrv>FAQ</abrv>
        <descricao>Falha do atuador por queima</descricao>
    </Falha>
    <Falha>
        <local>Atuador</local>
        <abrv>FASR</abrv>
        <descricao>Falha do atuador por sensibilidade à ruídos</descricao>
    </Falha>
    <Falha>
        <local>Atuador</local>
        <abrv>FAVK</abrv>
        <descricao>Falha do atuador por variação da constante da
                    bomba</descricao>
    </Falha>
    <Falha>
        <local>Sensor</local>
        <abrv>FSeDG</abrv>
        <descricao>Falha do sensor por descalibramento de ganho</descricao>
    </Falha>
    <Falha>
        <local>Sensor</local>
        <abrv>FSeDO</abrv>
        <descricao>Falha do sensor por descalibramento de offset</descricao>
    </Falha>
    <Falha>
        <local>Sensor</local>
        <abrv>FSeQ</abrv>
        <descricao>Falha do sensor por queima</descricao>
    </Falha>
    <Falha>
        <local>Sensor</local>
        <abrv>FSeSR</abrv>
        <descricao>Falha do sensor por sensibilidade à ruídos</descricao>
    </Falha>
```

```
    <Falha>
        <local>Sistema</local>
        <abrv>FSiEOS</abrv>
        <descricao>Falha do sistema por entupimento do orifício de
                    saída</descricao>
    </Falha>
    <Falha>
        <local>Sistema</local>
        <abrv>FSiVrGMP</abrv>
        <descricao>Falha do sistema por variação do ganho do módulo de
                    potência</descricao>
    </Falha>
    <Falha>
        <local>Sistema</local>
        <abrv>FSiVrOS</abrv>
        <descricao>Falha do sistema por variação do diâmetro do orifício de
                    saída</descricao>
    </Falha>
    <Falha>
        <local>Sistema</local>
        <abrv>FSiVzT</abrv>
        <descricao>Falha do sistema por vazamento do Tanque</descricao>
    </Falha>
</Simddef>
```

A.3.2 Arquivo de configuração de módulos

```
<?xml version="1.0" encoding="UTF-8"?>
<Simddef versao="1.0" tipo="modulos">
    <Modulo>
        <tipo nome="RNA">
            <ordem>2</ordem>
        </tipo>
        <falha>FADG</falha>
        <arquivos qtde="3">
            <arq end="cfgs_rnas/FADG/entrada.dat"/>
            <arq end="cfgs_rnas/FADG/O2N8T3_RNA.cfg"/>
            <arq end="cfgs_rnas/FADG/O2N8T3_RNA.lim"/>
        </arquivos>
    </Modulo>
    <Modulo>
        <tipo nome="RNA">
            <ordem>4</ordem>
        </tipo>
        <falha>FADO</falha>
        <arquivos qtde="3">
            <arq end="cfgs_rnas/FADO/entrada.dat"/>
            <arq end="cfgs_rnas/FADO/O4N28T3_RNA.cfg"/>
            <arq end="cfgs_rnas/FADO/O4N28T3_RNA.lim"/>
        </arquivos>
    </Modulo>
```

```
    <Modulo>
        <tipo nome="RNA">
            <ordem>4</ordem>
        </tipo>
        <falha>FAQ</falha>
        <arquivos qtde="3">
            <arq end="cfgs_rnas/FAQ/entrada.dat"/>
            <arq end="cfgs_rnas/FAQ/O4N28T6_RNA.cfg"/>
            <arq end="cfgs_rnas/FAQ/O4N28T6_RNA.lim"/>
        </arquivos>
    </Modulo>
    <Modulo>
        <tipo nome="RNA">
            <ordem>2</ordem>
        </tipo>
        <falha>FASR</falha>
        <arquivos qtde="3">
            <arq end="cfgs_rnas/FASR/entrada.dat"/>
            <arq end="cfgs_rnas/FASR/O2N8T6_RNA.cfg"/>
            <arq end="cfgs_rnas/FASR/O2N8T6_RNA.lim"/>
        </arquivos>
    </Modulo>
    <Modulo>
        <tipo nome="RNA">
            <ordem>2</ordem>
        </tipo>
        <falha>FAVK</falha>
        <arquivos qtde="3">
            <arq end="cfgs_rnas/FAVK/entrada.dat"/>
            <arq end="cfgs_rnas/FAVK/O2N8T5_RNA.cfg"/>
            <arq end="cfgs_rnas/FAVK/O2N8T5_RNA.lim"/>
        </arquivos>
    </Modulo>
    <Modulo>
        <tipo nome="RNA">
            <ordem>4</ordem>
        </tipo>
        <falha>FSeDG</falha>
        <arquivos qtde="3">
            <arq end="cfgs_rnas/FSeDG/entrada.dat"/>
            <arq end="cfgs_rnas/FSeDG/O4N28T2_RNA.cfg"/>
            <arq end="cfgs_rnas/FSeDG/O4N28T2_RNA.lim"/>
        </arquivos>
    </Modulo>
    <Modulo>
        <tipo nome="RNA">
            <ordem>4</ordem>
        </tipo>
        <falha>FSeDO</falha>
        <arquivos qtde="3">
            <arq end="cfgs_rnas/FSeDO/entrada.dat"/>
            <arq end="cfgs_rnas/FSeDO/O4N28T5_RNA.cfg"/>
            <arq end="cfgs_rnas/FSeDO/O4N28T5_RNA.lim"/>
        </arquivos>
    </Modulo>
```

```
    <Modulo>
        <tipo nome="RNA">
            <ordem>4</ordem>
        </tipo>
        <falha>FSeQ</falha>
        <arquivos qtde="3">
            <arq end="cfgs_rnas/FSeQ/entrada.dat"/>
            <arq end="cfgs_rnas/FSeQ/O4N20T4_RNA.cfg"/>
            <arq end="cfgs_rnas/FSeQ/O4N20T4_RNA.lim"/>
        </arquivos>
    </Modulo>
    <Modulo>
        <tipo nome="RNA">
            <ordem>4</ordem>
        </tipo>
        <falha>FSeSR</falha>
        <arquivos qtde="3">
            <arq end="cfgs_rnas/FSeSR/entrada.dat"/>
            <arq end="cfgs_rnas/FSeSR/O4N20T3_RNA.cfg"/>
            <arq end="cfgs_rnas/FSeSR/O4N20T3_RNA.lim"/>
        </arquivos>
    </Modulo>
    <Modulo>
        <tipo nome="RNA">
            <ordem>2</ordem>
        </tipo>
        <falha>FSiEOS</falha>
        <arquivos qtde="3">
            <arq end="cfgs_rnas/FSiEOS/entrada.dat"/>
            <arq end="cfgs_rnas/FSiEOS/O2N12T4_RNA.cfg"/>
            <arq end="cfgs_rnas/FSiEOS/O2N12T4_RNA.lim"/>
        </arquivos>
    </Modulo>
    <Modulo>
        <tipo nome="RNA">
            <ordem>4</ordem>
        </tipo>
        <falha>FSiVrGMP</falha>
        <arquivos qtde="3">
            <arq end="cfgs_rnas/FSiVrGMP/entrada.dat"/>
            <arq end="cfgs_rnas/FSiVrGMP/O4N20T2_RNA.cfg"/>
            <arq end="cfgs_rnas/FSiVrGMP/O4N20T2_RNA.lim"/>
        </arquivos>
    </Modulo>
    <Modulo>
        <tipo nome="RNA">
            <ordem>2</ordem>
        </tipo>
        <falha>FSiVrOS</falha>
        <arquivos qtde="3">
            <arq end="cfgs_rnas/FSiVrOS/entrada.dat"/>
            <arq end="cfgs_rnas/FSiVrOS/O2N8T3_RNA.cfg"/>
            <arq end="cfgs_rnas/FSiVrOS/O2N8T3_RNA.lim"/>
        </arquivos>
    </Modulo>
```

```
135     <Modulo>
136         <tipo nome="RNA">
137             <ordem>4</ordem>
138         </tipo>
139         <falha>FSiVzT</falha>
140         <arquivos qtde="3">
141             <arq end="cfgs_rnas/FSiVzT/entrada.dat"/>
142             <arq end="cfgs_rnas/FSiVzT/O4N24T1_RNA.cfg"/>
143             <arq end="cfgs_rnas/FSiVzT/O4N24T1_RNA.lim"/>
144         </arquivos>
145     </Modulo>
146 </Simddef>
```

REFERÊNCIAS BIBLIOGRÁFICAS

Angeli, C. e A. Chatzinikolaou (2004), 'On-line fault detection techniques for technical systems: A survey', *International Journal of Computer Science & Applications* **1**(1), 12–30.

Apkarian, Jacob (1999), *Coupled water tank experiments manual*, Quanser Consulting Inc., Canada.

Arruda, Lúcia Valéria Ramos, Flávio Neves Júnior e Sílvio Fávaro (2003), 'Identificação genética de modelos por pólos e zeros baseada no compromisso entre os erros de polarização e variância', *Revista Controle & Automação* **14**(2).

Avižienis, Algirdas, Jean-Claude Laprie e Brian Randell (2000), Fundamental concepts of dependability, *em* 'Proceedings of the 3rd Information Survivability Workshop', pp. 7–12.

Baskiotis, C., J. Raymond e A. Rault (1979), Parameter identification and discriminant analysis for jet engine machanical state diagnosis, *em* '18th IEEE Conference on Decision and Control including the Symposium on Adaptive Processes', Vol. 18, pp. 648–650.

Beard, Richard Vernon (1971), Failure accommodation in linear systems through self-reorganization, Tese de doutorado, Massachusetts Institute of Technology, Departament of Aeronautics and Astronautics.

Bennett, Stuart (1996), A brief history of Automatic Control, *em* 'IEEE Control Systems Magazine', Vol. 16(3), pp. 17–25.

Billman, L. e R. Isermann (1987), 'Leak detection methods for pipelines', *Automatica* **23**(3), 381–385.

Blanke, M., M. Kinnaert, J. Lunze, M. Staroswiecki e J. Schröder (2006), *Diagnosis and Fault-Tolerant Control*, Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Braga, Antônio de Pádua, André Ponce de Leon F. De Carvalho Ludermir e Teresa Bernarda (2007), *Redes Neurais Artificiais – Teoria e Prática*, LTC.

Chang, C.S., Z. Xu e A. Khambadkone (2003), 'Enhancement and laboratory implementation of neural network detection of short circuit faults in dc transit system', *IEE Proceedings – Electric Power Applications* **150**(3), 344–350.

Chiang, L. H., E. L. Russel e R. D. Braatz (2001), *Fault detection and diagnosis in industrial systems*, Springer.

Clark, Robert N. (1978), 'A simplified instrument failure detection scheme', *IEEE Transactions on Aerospace and Electronic Systems* **14**(4), 558–563.

Cybenko, George (1989), 'Approximation by superpositions of a sigmoidal function', *Mathematics of Control, Signals, and Systems (MCSS)* **2**(4), 303–314.

Dorf, Richard C. e Robert H. Bishop (2009), *sistemas de controle modernos*, 11ª edição, Rio de Janeiro: Editora LTC.

Erdenetsetseg, T. e D. Ulemj (2007), Supervision and fault management of process-tasks and terminology, *em* 'International Forum on Strategic Technology', pp. 444–447.

Ericson, S., N. Grip, E. Johannson, L. Persson, R. Sjöberg e J. Strömberg (2005), Towards automatic detection of local bearing defects in rotating machines, *em* 'Mechanical Systems and Signal Processing', Vol. 9, pp. 509–535.

Faccin, Flávio (2004), Abordagem inovadora no projeto de controladores PID, Dissertação de mestrado, Universidade Federal do Rio Grande do Sul, Porto Alegre.

Fesq, L. M. (2009), Current fault management trends in nasa's planetary spacecraft, *em* 'IEEE Aerospace conference', pp. 1–9.

Filbert, D. (1985), Fault diagnosis in nonlinear electromechanical systems by continous-time parameter estimation, *em* 'ISA Transactions', Vol. 24(3), pp. 23–27.

Filbert, D. e K. Metzger (1982), Quality test of systems by parameter estimation, *em* '9th IMEKO Congress', Berlin, Germany.

Gao, X. Z., S. J. Ovaska e Y. Dote (2000), Motor fault detection using elman neural network with genetic algorithm-aided training, *em* 'IEEE International Conference on Systems, Man, and Cybernetics', Vol. 4, pp. 2386–2392.

Gertler, Janos e David Singer (1985), Augmented models for statistical fault isolation in complex dynamic systems, *em* 'American Control Conference', pp. 317–322.

Goeking, Weruska (2010), 'Da máquina a vapor aos softwares de automação', Publicação online – O Setor Elétrico/Memória da Eletricidade. [Acessado em 20/08/2010].
URL: *http://www.osetoreletrico.com.br/web/automacao.html*

Guo, Qian-jin, Hai bin Yu e Ai dong Xu (2005), Modified morlet wavelet neural networks for fault detection, *em* 'International Conference on Control and Automation', Vol. 2, pp. 1209–1214.

Hang, L. e M. Lei (2009), Definition of managed objects for fault management of satellite network, *em* 'International Forum on Information Technology and Applications', Vol. 1, pp. 107–110.

Haykin, Simon (2000), *Redes Neurais: princípios e prática*, Bookman.

Higham, E. e S. Perovic (2001), Predictive maintenance of pumps based on signal analysis of pressure and differential pressure (flow) measurements, *em* 'Transactions of Institute of Measurement and Control', Vol. 23(4), pp. 226–248.

Himmelblau, David M. (1978), *Fault detection and diagnosis in chemical and petrochemical processes*, Elsevier Scientific Pub. Co.

Houghton, Edward Lewis e Peter W. Carpenter (2002), *Aerodynamics for Engineering Students*, Butterworth Heinemann.

Isermann, Rolf (1982), 'Parameter-adaptative control algorithms', *Automatica* **18**(5), 513–528.

Isermann, Rolf (1984), 'Process fault detection based on modeling and estimation methods – a survey', *Automatica* **20**(4), 387–404.

Isermann, Rolf (1993), 'Fault diagnosis of machines via parameter estimation and knowledge processing: tutorial paper', *Automatica* **29**(4), 815–835.

Isermann, Rolf (2004), Model-based fault detection and diagnosis: status and applications, *em* 'Proceedings of the 16th IFAC Symposium on Automatic Control in Aerospace', pp. 71–85.

Isermann, Rolf (2006), *Fault-diagnosis systems: an introduction from fault detection to fault tolerance*, Springer-Verlag.

Jia-li, Tang, Liu Yi-jun e Wu Fang-sheng (2010), Levenberg-marquardt neural network for gear fault diagnosis, *em* '2nd International Conference on Networking and Digital Society', Vol. 1, pp. 134–137.

Jones, Harold Lee (1973), Failure detection in linear systems, Tese de doutorado, Massachusetts Institute of Technology, Departament of Aeronautics and Astronautics.

Kaâniche, Mohamed, Jean-Claude Laprie e Jean-Paul Blanquart (2002), 'A framework for dependability engineering of critical computing systems', *Safety Science* **40**(9), 731–752.

Khaled, O., D. Hedi, N. Lotfi, H. Messaoud e Z. S-abazi (2010), Fault detection and localization with neural principal component analysis, *em* '18th Mediterranean Conference on Control Automation (MED)', pp. 880–885.

Laprie, Jean-Claude (1996), Dependable computing and fault tolerance : concepts and terminology, *em* '25th International Symposium on Fault-Tolerant Computing', Vol. 3, pp. 2–11.

Laprie, Jean-Claude, A. Avizienis e H. Kopetz (1992), *Dependability: Basic Concepts and Terminology*, Springer-Verlag New York, Inc., New York.

Laprie, Jean-Claude, T. Anderson, A. Avizienis, W. C. Carter, A. Costes, F. Cristian, Y. koga, H. Kopetz, J. H. Lala, J. F. Meyer, B. Randell e A. S. Robinson (1994), 'Dependability: Basic concepts and terminology', International Federation for Information Processing – IFIP WG 10.4 - Dependable Computing and Fault Tolerance – Draft Version.

Lucena, Pedro Berretta (2005), Análise de um controlador baseado no jacobiano estimado da planta através de uma rede neural, Dissertação de mestrado, Universidade Federal do Rio Grande do Norte.

Maxwell, J. C. (1964), On governors, *em* 'Proceedings of the Royal Society of London', Vol. 16, New York, pp. 270–283. Note: 1868, in *Selected Papers on Mathematical Trends in Control Theory*.

Mayr, Otto (1970), 'The origins of feedback control', *Scientific American* **223**, 110–118.

Mayr, Otto (1971), *Feedback mechanisms in the historical collections of the National Museum of History and Technology*, Washington: Smithsonian Institution Press. Includes bibliographical references [Acessado em 20/08/2010].
URL: *http://hdl.handle.net/10088/2411*

Mayr, Otto (1975), *The origins of feedback control*, MIT Press, Cambridge, MA, USA.

Mehra, R. K. e J. Peschon (1971), 'An innovations approach to fault detection and diagnosis in dynamic systems', *Automatica* **7**(5), 637–640.

Nørgaard, Magnus, Ole Ravn, Niels K. Poulsen e Lars K. Hansen (2000), *Neural Networks for Modelling and Control of Dynamic Systems*, Springer-Verlag.

Oliveira, Leonardo Dantas de (2008), 'Desenvolvimento de um gateway de comunicação para interação com plataformas de aquisição e escrita de dados', Trabalho de Conclusão de Curso – Universidade Federal do Rio Grande do Norte.

Patton, R. J. e J. Chen (1991), Robust fault detection using eigenstructure assignment: a tutorial consideration and some new results, *em* '30th IEEE Conference on Proceedings of the Decision and Control', Vol. 3, pp. 2242–2247.

Patton, R., P. Frank e P. Clark (2000), *Issues of fault diagnosis for dynamic systems*, Springer, New York.

Pereira Filho, Nélio Alves (2002), 'Linux, clusters e alta disponibilidade', Publicação online. [Acessado em 10/09/2010].
URL: *http://www.ime.usp.br/∼nelio/*

Rebouças, Diogo Leite (2009), 'Sistema de inferência neural e processamento estatístico multivariável aplicado a indústria do petróleo.', Trabalho de Conclusão de Curso – Universidade Federal do Rio Grande do Norte.

Ribeiro, Marco Antônio (1999), *Automação industrial*, 4ª edição, Tek Treinamento & Consultoria.

Routh, Edward John (1877), *A treatise on the stability of a given state of motion, particularly steady motion*, MacMillan, London.

Russel, E., L. Chiang e R. Baatz (2000), *Data-driven techniques for fault detection and diagnosis in chemical processes*, Springer, London.

Silva, Diego Rodrigo Cabral (2008), Sistema de detecção e isolamento de falhas em sistemas dinâmicos baseados em identificação paramétrica, Tese de doutorado, Universidade Federal do Rio Grande do Norte.

Simani, S., C. Fantuzzi e R. Patton (2003), *Model-based fault diagnosis in dynamic systems using identification techniques*, Springer, London.

Sreedhar, R., B. Fernandez e G. Y. Masada (1995), A neural network based adaptive fault detection scheme, *em* 'Proceedings of the American Control Conference', Vol. 5, pp. 3259–3263.

Talebi, H. A. e R. V. Patel (2005), A neural network-based fault detection scheme for satellite attitude control systems, pp. 1293–1298.

Tian, Jingwen, Meijuan Gao, Liting Cao e Kai Li (2007), Fault detection of oil pump based on fuzzy neural network, *em* 'Third International Conference on Natural Computation', Vol. 2, pp. 636–640.

Vachtsevanos, George, Frank Lewis, Michael Roemer, Andrew Hess e Biqing Wu (2006), *Intelligent fault diagnosis and prognosis for engineering systems*, John Wiley & Sons, New Jersey.

VDI (2003), 'VDI Guidelines 2206 – Design methodology for mechatronic systems'.

Vemuri, A. T., M. M. Polycarpou e S. A. Diakourtis (1998), 'Neural network based fault detection in robotic manipulators', *IEEE Transactions on Robotics and Automation* **14**(2), 342–348.

Venkatasubramanian, V., R. Rengaswamy, K. Yin e S. N. Kavuri (2003*a*), A review of process fault detection and diagnosis Part I: Quantitative model-based methods, *em* 'Computers & Chemical Engineering', Vol. 27, pp. 293–311.

Venkatasubramanian, V., R. Rengaswamy, K. Yin e S. N. Kavuri (2003*b*), A review of process fault detection and diagnosis Part II: Qualitative models and search strategies, *em* 'Computers & Chemical Engineering', Vol. 27, pp. 313–326.

Venkatasubramanian, V., R. Rengaswamy, K. Yin e S. N. Kavuri (2003*c*), A review of process fault detection and diagnosis Part III: Process history based methods, *em* 'Computers & Chemical Engineering', Vol. 27, pp. 327–346.

Weber, Taisy Silvar (2002), 'Um roteiro para exploração dos conceitos básicos de tolerância a falhas', Publicação online. [Acessado em 10/09/2010].
URL: *http://www.inf.ufrgs.br/ taisy/disciplinas/textos/*

Willsky, Alan S. (1976), 'A survey of design methods for failure detection in dynamic systems', *Automatica* **12**, 601–611.

Wu, N. E. (2004), 'Coverage in fault-tolerant control', *Automatica* **40**, 537–548.

Zhang, Y. e J. Jiang (2008), 'Bibliographical review on reconfigurable fault-tolerant control systems', *Annual Reviews in Control* **32**(2), 229–252.
URL: *http://dx.doi.org/10.1016/j.arcontrol.2008.03.008*

Ziegler, J. G. e N. B. Nichols (1942), 'Optimum settings for automatic controllers', *The American Society of Mechanical Engineers Transactions* **64**, 759–768.