UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

JEFFERSON LUIZ BOSA

# Sistema Embarcado para a Manutenção Inteligente de Atuadores Elétricos

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação

Prof. Dr. Marcelo Soares Lubaszewski
Orientador

Porto Alegre, dezembro de 2009.

**CIP – CATALOGAÇÃO NA PUBLICAÇÃO**

Bosa, Jefferson Luiz

Sistema Embarcado para a Manutenção Inteligente de Atuadores Elétricos / Jefferson Luiz Bosa – Porto Alegre: Programa de Pós-Graduação em Computação, 2009.

169 f.:il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação. Porto Alegre, BR – RS, 2009. Orientador: Marcelo Soares Lubaszewski.

1. Testes On-Line. 2. Tolerância a Falhas. 3. Mapas Auto-Organizáveis. 4. Sistema Embarcado. 5. FPGA. 6. Manutenção. I. Lubaszewski, Marcelo Soares. II. Título.

# AGRADECIMENTOS

Agradeço em primeiro lugar ao Senhor Jesus pelo dom da vida, salvação, pelo amor demonstrado e tudo o que aconteceu em minha vida. Também dou graças a Deus pelo tempo que morei em Porto Alegre, pois foram tempos que jamais me esquecerei.

Em segundo lugar quero honrar meus pais (José Bosa e Elieusa Chagas Bosa) pelo amor, pelas orações, testemunho de vida e por terem me ensinado a andar nos caminhos de Deus. Também quero honrar meus irmãos (Estefan Michel Bosa e Ruan Hedpo Bosa) pelo amor, companheirismo e pelas orações. Agradeço a Deus todos os dias pela vida de vocês. Agradeço a meus avós (Helena da Rosa e Ari da Rosa) pelo amor e em especial pelas orações e súplicas.

Agradeço aos irmãos de aliança em Cristo (Jackson, Daniel, Fernando e grupo) que conheci em Porto Alegre, por terem dedicado suas vidas em andarmos juntos e a formar a imagem de Cristo em nossas vidas. Também agradeço aos irmãos com quem dividi moradia (Mateus, Cristiano, Carlos Eduardo e Marko) por serem pessoas que me ensinaram a viver como família de Cristo. E aos demais irmãos (que são muitos) que congregam na Igreja em Porto Alegre, com quem mantive vínculo. Dou graças a Deus por ter conhecido vocês.

Agradeço aos amigos do laboratório 205 e do Laprot, pela amizade e momentos que passamos juntos.

Agradeço ao professor Marcelo Lubaszewski, pelo privilégio de trabalharmos juntos, exemplo de profissional, pela paciência e amizade.

Agradeço a UFRGS pela oportunidade de estudar nesta universidade e a CAPES pelo apoio financeiro.

*“Confia no Senhor de todo o teu coração, e não te estribes no teu próprio entendimento. Reconhece-o em todos os teus caminhos e Ele endireitará as tuas veredas” Provérbios 4:5-6*

# SUMÁRIO

# LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| AR | Autoregressive Model |
| ARMA | Autoregressive Moving Model |
| BMU | (*Best-Matching Unit*) Neurônio Vencedor |
| BRAM | Block RAM |
| CBM | Manutenção Baseado em Condição |
| CNCO | Centro Nacional de Controle Operacional da Petrobras |
| Creduto | Centro Nacional de Reparo de Dutos da Petrobras |
| CTDUT | Centro de Tecnologia de Dutos da Petrobras |
| FFT | *Fast Fourier Trnasform* |
| FPGA | *Field Programmable Gate Array* |
| IHM | Interface Homem-Máquina |
| IMS Center | *Center for Intelligent Maintenance Systems* |
| MMU | *Memory Management Unit* |
| OSA-CBM | *Open Systems Architecture for Condition-Based Maintenance* |
| PCA | *Principal Component Analysis* |
| SCADA | *Supervisory Control and Data Aquisition* |
| Sistema DDF | Sistemas de Detecção e Diagnóstico de Falhas |
| Sistema DDPF | Sistemas de Detecção, Diagnóstico e Predição de Falhas |
| SMI | Sistema de Manutenção Inteligente |
| SOE | Sistema Operacional Embarcado |
| SOM | (*Self-Organizing Maps*) Mapas Auto-Organizáveis |
| STM | (S*hort-term memory*) Memória de Curto Prazo |
| TKM | (*Temporal Kohonen Maps*) Mapa Temporal de Kohonen |
| UFRGS | Universidade Federal do Rio Grande do Sul |
| *U-Matrix* | (*Unified Distance Matrix*) Matriz de Distância Unificada |
| VHDL | *Very High Speed Integrated Circuit Hardware Description Language* |
| WA | *Watchdog Agent* |

# LISTA DE FIGURAS

# LISTA DE TABELAS

# RESUMO

O elevado custo de manutenção nos ambientes industriais motivou pesquisas de novas técnicas para melhorar as ações de reparos. Com a evolução tecnológica, principalmente da eletrônica, que proporcionou o uso de sistemas embarcados para melhorar as atividades de manutenção, estas agregaram inteligência e evoluíram para uma manutenção pró-ativa. Através de ferramentas de processamento de sinais, inteligência artificial e tolerância a falhas, surgiram novas abordagens para os sistemas de monitoramento a serviço da equipe de manutenção. Os ditos sistemas de manutenção inteligente, cuja tarefa é realizar testes em funcionamento (*on-line*) nos equipamentos industriais, promovem novos modelos de confiabilidade e disponibilidade. Tais sistemas são baseados nos conceitos de tolerância a falhas, e visam detectar, diagnosticar e predizer a ocorrência de falhas. Deste modo, fornece-se aos engenheiros de manutenção a informação antecipada do estado de comportamento do equipamento antes mesmo deste manifestar uma falha, reduzindo custos, aumentando a vida útil e tornando previsível o reparo. Para o desenvolvimento do sistema de manutenção inteligente objeto deste trabalho, foram estudadas técnicas de inteligência artificial (redes neurais artificiais), técnicas de projeto de sistemas embarcados e de prototipação em plataformas de hardware. No presente trabalho, a rede neural Mapas Auto-Organizáveis foi adotada como ferramenta base para detecção e diagnóstico de falhas. Esta foi prototipada numa plataforma de sistema embarcado baseada na tecnologia FPGA (*Field Programmable Gate Array*). Como estudo de caso, uma válvula elétrica utilizada em dutos para transporte de petróleo foi definida como aplicação alvo dos experimentos. Através de um modelo matemático, um conjunto de dados representativo do comportamento da válvula foi simulado e utilizado como entrada do sistema proposto. Estes dados visam o treinamento da rede neural e visam fornecer casos de teste para experimentação no sistema. Os experimentos executados em software validaram o uso da rede neural como técnica para detecção e diagnóstico de falhas em válvulas elétricas. Por fim, também realizou-se experimentos a fim de validar o projeto do sistema embarcado, comparando-se os resultado obtidos com este aos resultados obtidos a partir de testes em software. Os resultados revelam a escolha correta do uso da rede neural e o correto projeto do sistema embarcado para desempenhar as tarefas de detecção e diagnóstico de falhas em válvulas elétricas.

**Palavras-Chave:** testes on-line, tolerância a falhas, mapas auto-organizáveis, sistema embarcado, FPGA, manutenção.

**Embedded Systems for Intelligent Maintenance of Electrical Actuators**

# ABSTRACT

The high costs of maintenance in industrial environments have motivated research for new techniques to improve repair activities. The technological progress, especially in the electronics field, has provided for the use of embedded systems to improve repair, by adding intelligence to the system and turning the maintenance a proactive activity. Through tools like signal processing, artificial intelligence and fault-tolerance, new approaches to monitoring systems have emerged to serve the maintenance staff, leading to new models of reliability and availability. The main goal of these systems, also called intelligent maintenance systems, is to perform in-operation (on-line) test of industrial equipments. These systems are built based on fault-tolerance concepts, and used for the detection, the diagnosis and the prognosis of faults. They provide the maintenance engineers with information on the equipment behavior, prior to the occurrence of failures, reducing maintenance costs, increasing the system lifetime and making it possible to schedule repairing stops. To develop the intelligent maintenance system addressed in this dissertation, artificial intelligence (neural networks), embedded systems design and hardware prototyping techniques were studied. In this work, the neural network Self-Organizing Maps (SOM) was defined as the basic tool for the detection and the diagnosis of faults. The SOM was prototyped in an embedded system platform based on the FPGA technology (Field Programmable Gate Array). As a case study, the experiments were performed on an electric valve used in a pipe network for oil transportation. Through a mathematical model, a data set representative of the valve behavior was obtained and used as input to the proposed maintenance system. These data were used for neural network training and also provided test cases for system monitoring. The experiments were performed in software to validate the chosen neural network as the technique for the detection and diagnosis of faults in the electrical valve. Finally, experiments to validate the embedded system design were also performed, so as to compare the obtained results to those resulting from the software tests. The results show the correct choice of the neural network and the correct embedded systems design to perform the activities for the detection and diagnosis of faults in the electrical valve.

**Keywords:** on-line test, fault-tolerance, self-organizing maps, embedded systems, FPGA, maintenance.

# 1 INTRODUÇÃO

Os equipamentos presentes em processos industriais, à medida que são utilizados, tornam-se suscetíveis à ação da degradação: desgaste, corrosão, rachaduras, danos causados por operadores entre outras anomalias que podem contribuir para a degradação. Caso os reparos não sejam feitos, apresentarão falhas que afetam toda a cadeia produtiva, tornando os equipamentos incapazes de desempenharem sua função com eficiência (LEE, JAY, 2003).

A manutenção consiste em uma série de técnicas, ou ações, para restabelecer (ou reparar) os equipamentos. Tais medidas são tomadas principalmente com a intenção de corrigir os danos provocados pela degradação, de manter o equipamento em funcionamento e de reduzir a manifestação de falhas.

As estratégias de manutenção podem ser divididas em quatro grupos: manutenção corretiva (ações paliativas e corretivas), manutenção preventiva (ações periódicas e curativas), manutenção preditiva (ações planejadas e monitoradas) e a manutenção proativa (ações proativas) (ENDRENYI ET AL., 2001). Elas devem ser aplicadas de forma a orientar e planejar os trabalhos desempenhados pela equipe de manutenção, a fim de atingir os objetivos esperados pela empresa, pois estão diretamente associadas aos resultados produtivos que podem afetar a disponibilidade, qualidade dos produtos e os custos operacionais.

Devido a novas tecnologias de sensores e sistemas computacionais embarcados, tornou-se praticável avançar para além das estratégias de manutenção corretiva e preventiva na direção da manutenção proativa, também chamada de *manutenção inteligente*. Nessa nova estratégia, pode-se avaliar, com rapidez e precisão, os indicadores de desempenho, monitorando a condição de comportamento dos equipamentos e adicionando tolerância a falhas ao sistema, e portanto, possibilitando a detecção, o diagnóstico e a predição de falhas.

A partir desse ponto, em função da detecção, do diagnóstico e da predição de falhas nos equipamentos, torna-se possível, por exemplo, realizar os reparos de forma planejada em momentos de ociosidade ou de menor uso do equipamento. Se for impossível interromper o equipamento, ele pode reconfigurar-se para continuar operando, mesmo de maneira degradada, até que as partes defeituosas sejam reparadas.

## 1.1 Contexto e motivação

O presente trabalho está inserido no contexto de técnicas de teste e tolerância a falhas aplicadas a equipamentos industriais. É tratado com base em um estudo de caso

sobre uma necessidade da indústria de petróleo e gás natural (Transpetro[1], subsidiária de logística da Petrobras[2]) em relação a prestar suporte de manutenção na rede dutoviária. Essa rede é formada por diversos dutos para transporte dos derivados de petróleo, por válvulas de controle de fluxo, entre outros equipamentos.

O foco deste trabalho está nos atuadores elétricos, que são equipamentos motorizados acoplados às válvulas, a fim de automatizar sua operação. Esses equipamentos estão sujeitos à degradação. Quando ocorrem falhas nos atuadores ou válvulas, a empresa Transpetro tem dificuldades em gerenciar a manutenção. Parte dessa dificuldade está ligada ao tamanho continental de suas redes dutoviárias e da grande quantidade de atuadores espalhados pela rede, que nesse caso torna a realização das manutenções preventivas e corretivas muito caras.

Desse modo, esta pesquisa propõe a construção de um sistema embarcado que seja implantado no sistema eletrônico de controle dos atuadores. Este terá a finalidade de monitorar o equipamento em busca de falhas. Quando detectar alguma anormalidade, será enviado uma mensagem de alarme para a equipe de manutenção solicitando a realização de reparos, fornecendo também informações a respeito do diagnóstico e estimando o tempo restante até a falha.

Para desenvolver esse sistema embarcado, foi preciso estudar as ferramentas de inteligência artificial que possibilitam atender aos requisitos de tolerância a falhas para monitoramento dos equipamentos. Foi definido pesquisar a rede neural SOM (Mapas Auto-Organizáveis) (KOHONEN, 2001) como técnica computacional de detecção, diagnóstico e predição de falhas. Outra motivação foi a implementação dessa rede neural em um protótipo em *hardware*, a fim de ser um componente do sistema embarcado. Desta forma, foi também necessário revisar a fundamentação teórica, implementações em *softwares* e *hardware*, e estudar técnicas de projeto de sistemas embarcados.

A motivação principal deste trabalho é disponibilizar uma ferramenta que auxilie a equipe de manutenção em realizar as tarefas de manutenção. Para possibilitar o monitoramento de falhas dos equipamentos e fornecer informações precisas para a tomada de decisão no planejamento da manutenção. Este trabalho tem relevância para a Transpetro, pois será aplicado em todas as válvulas e atuadores instalados ao longo de sua rede dutoviária.

## 1.2 Objetivos

O objetivo dessa dissertação é projetar um protótipo de um sistema embarcado que implemente técnicas de manutenção inteligente, utilizando a rede neural SOM como ferramenta para monitoramento de falhas em atuadores elétricos nos dutos da Transpetro. No protótipo, será embarcada a rede neural implementada em *hardware* para fazer *detecção* e *diagnóstico* de falhas. A partir dessa meta inicial, o presente trabalho atende aos seguintes objetivos específicos:

a) Apresentar conceitos básicos sobre manutenção industrial e uma breve análise dos problemas da manutenção que justificam e contextualizam a

[1] Site da empresa Transpetro – www.transpetro.com.br

[2] Site da empresa Petrobras S.A. – www.petrobras.com.br

aplicação da manutenção inteligente como ferramenta de teste e tolerância a falhas em equipamentos industriais.

b) Apresentar conceitos, características e técnicas de tolerância a falhas para realizar detecção, diagnóstico e predição de falhas pertencentes ao contexto do presente trabalho.

c) Apresentar conceitos e fundamentação teórica sobre as redes neurais: SOM (Mapas Auto-Organizáveis) e TKM (Mapa Temporal de Kohonen), que serão utilizadas como ferramentas para análise de falhas.

d) Desenvolver experimentos em *software* do SOM e TKM, a fim de validar essas técnicas para os dados do estudo de caso em atuadores elétricos.

e) Desenvolver um protótipo em *hardware* de um sistema embarcado que implemente a manutenção inteligente, para monitorar em tempo real as falhas em atuadores elétricos. Esse protótipo deverá ser baseado em lógica programável (FPGA - *Field Programmable Gate Array*) e utilizará desenvolvimento misto entre *hardware* e *software*.

f) Exibir os resultados da análise de falhas em atuadores elétricos, considerando os experimentos em *software* e do protótipo em *hardware*, comparando os resultados para validar o projeto do protótipo.

Como objetivo final, do ponto de vista da manutenção, este trabalho propõe uma ferramenta que forneça subsídios para as equipes de manutenção a fim de melhorar o planejamento de suas tarefas de reparos nos equipamentos de modo mais eficiente. Dessa forma, pretende-se alcançar maior confiabilidade no processo industrial, reduzir custos, aumentar disponibilidade, programar com antecedência os reparos, reutilizar peças e evitar maiores danos nos equipamentos.

## 1.3 Contribuições

Esta dissertação traz os seguintes avanços e contribuições técnicas:

a) Revisão sobre estratégias de manutenção em plantas industriais e problemas econômicos envolvidos.

b) Revisão de técnicas e conceitos de tolerância a falhas para detecção, diagnóstico e predição de falhas do ponto de vista de sistemas computacionais.

c) Revisão do estado da arte de redes neurais não supervisionadas (SOM) e a não supervisionada temporal (TKM).

d) Implementação e revisão sobre projeto de sistemas embarcados, utilizando plataforma FPGA, sistemas operacionais embarcados e *software* embarcado.

e) Proposta de uma implementação da rede neural SOM em *hardware* em plataforma FPGA.

f) Proposta de uma plataforma em sistema embarcado para experimentos, com preparação futura do protótipo do sistema de manutenção inteligente a ser implantado em atuadores elétricos.

g) Simulação do protótipo utilizando dados de teste para um conjunto de falhas do atuador.

Finalmente, com a revisão bibliográfica foi constatado que existem diversos trabalhos acadêmicos que utilizam o SOM como ferramenta de detecção e diagnóstico de falhas em diversos processos, comprovando a eficácia da rede neural nessa função. No entanto, o presente trabalho traz a contribuição de aplicar essas técnicas em atuadores elétricos, por meio de prototipação do SOM em FPGA. Além disso, foi estudada também a utilização do TKM como ferramenta para predição de falhas.

## 1.4 Organização do trabalho

A continuidade deste trabalho está organizada da seguinte forma:

- **Capítulo 2: Manutenção Inteligente**

  Este capítulo descreve, de forma geral, os conceitos relacionados à manutenção industrial, apresentando análise econômica envolvida e as principais estratégias de manutenção adotadas pela indústria. A discussão é focada na manutenção inteligente. Por fim, é apresentado o contexto do estudo de caso adotado.

- **Capítulo 3: Detecção, Diagnóstico e Predição de Falhas**

  Este capítulo apresenta uma revisão dos principais fundamentos sobre tolerância a falhas e tem seu foco nas técnicas de detecção, diagnóstico e predição de falhas. É apresentado também a fundamentação teórica sobre a rede neural SOM e a TKM como ferramentas para se alcançar a tolerância a falhas. Finalmente, é feita uma revisão na literatura de trabalhos correlatos.

- **Capítulo 4: Proposta de Protótipo de um Sistema de Manutenção Inteligente para Atuadores Elétricos**

  Este capítulo detalha a proposta de um sistema de manutenção inteligente para atuadores elétricos. Inicialmente, é apresentada uma visão geral de cada um dos componentes que formam a arquitetura do sistema de manutenção inteligente, os algoritmos para tolerância a falhas e a implementação destes em *software*. Em seguida, é feito o projeto de implementação da rede neural SOM em *hardware*, com base em FPGA, mostrando toda arquitetura, máquinas de estado e resultados (área, desempenho e consumo de energia). Em última etapa, é visto o projeto do sistema embarcado, que faz uso de uma placa de desenvolvimento para prototipação do sistema de manutenção inteligente. São apresentados arquitetura, integração do *hardware* do SOM e a plataforma de *software* embarcado adotado.

- **Capítulo 5: Experimentos e Resultados**

  Este capítulo visa relatar os resultados de experimentos realizados utilizando o protótipo e *software* implementados neste trabalho. São mostrados a metodologia adotada para realização dos experimentos, o estudo de caso e a ferramenta para simulação do modelo matemático do conjunto atuador e válvula. A seguir, é abordado o modelo para injeção de falhas no atuador, que faz uso do simulador, e como realizar a preparação e o treinamento das redes neurais desenvolvidas no decorrer do trabalho. Por fim, são apresentados os resultados do *software* para avaliação do uso da rede neural, considerando as tarefas de detecção, diagnóstico e predição de falhas. Finalmente, o protótipo é implementado em *hardware*, em uma placa de desenvolvimento, e os resultados

obtidos em hardware são comparados com os resultados do *software* para fim de validação do protótipo.

- **Capítulo 6: Conclusões e Trabalhos Futuros**

  Este capítulo relata as conclusões gerais e considerações finais do presente trabalho, apresentando as limitações do sistema proposto e discutindo possíveis desdobramentos e direções de trabalhos a serem realizados no futuro.

# 2 MANUTENÇÃO INTELIGENTE

## 2.1 Introdução

No contexto industrial, muitas das tarefas consideradas simples podem ser realizadas por operadores humanos, por exemplo: a abertura e o fechamento de válvulas ou o acionamento e desligamento de equipamentos, etc. Com o avanço tecnológico, os operadores foram, com o tempo, sendo substituídos pela automatização. Esse progresso trouxe enormes benefícios em vários setores da indústria, tais como químico, petroquímico, siderurgia, energia e saneamento básico. Além disso, aumentou a complexidade nas plantas industriais.

Entretanto, uma tarefa de suma importância é a manutenção dos equipamentos industriais, uma atividade basicamente manual. A manutenção consiste em uma série de técnicas e medidas de prevenção (para manter os equipamentos em funcionamento), correção (com o intuito de restabelecer os equipamentos danificados) e predição (estimar um tempo até a manifestação da falha).

Nesse contexto, será definido o termo “manutenção” como as ações praticadas com a intenção de reparar (corrigir, restabelecer) os danos provocados pela ação da degradação, a fim de manter os equipamentos em pleno funcionamento e com segurança (LEE, JAY, 2003).

Um equipamento está em funcionamento “normal” quando desempenha as tarefas (ou serviços) dentro da especificação para o qual foi projetado. Um “defeito” ocorre quando as funções desempenhadas pelo equipamento se desviam do especificado e pode ou não afetar a capacidade de desempenhar as tarefas. A “degradação” é um conjunto de funções em situação de defeito, em que, mesmo degradado, o equipamento pode desempenhar um subconjunto reduzido de tarefas. Uma “falha” é o estado de funcionamento que impede o equipamento de desempenhar as tarefas especificadas, ou seja, não cumpre com a especificação final de projeto.

A tarefa de manutenção é sensível a eventos anormais que ocorrem nos equipamentos. Quando se detecta um evento anormal na hora correta, é possível diagnosticar a origem da causa e, então, tomar as decisões apropriadas para reparos (DJURDJANOVIC ET AL., 2003).

Dentro dos ambientes industriais, as tarefas de manutenção são operações rotineiras e essenciais para manter a planta em funcionamento. Por isso, deve ser tratada com importância estratégica dentro da empresa, pois está diretamente vinculada à capacidade produtiva da indústria, caso contrário pode resultar em custos extras na produção.

Segundo (ALMEIDA, 2007), os custos de manutenção correspondem à parte principal dos custos operacionais totais em plantas industriais. Dependendo do ramo da indústria, os custos com manutenção podem representar entre 15 e 30% do custo de produção. Em indústrias alimentícias, os custos médios para manutenção podem alcançar em torno de 15% do custo de produção, enquanto que nas indústrias siderúrgicas, de papel e celulose e outras indústrias pesadas, a manutenção chega alcançar até 30% desses custos.

De acordo com (DJURDJANOVIC ET AL., 2003), devido à alta competitividade no mercado nacional e internacional, as empresas devem fornecer produtos e serviços com a mais alta qualidade possível, com a intenção de absorver e manter uma posição favorável de fatia de mercado. Problemas no fornecimento dos produtos, como atrasos ou baixa qualidade são praticamente inadmissíveis pelo mercado. Por exemplo, 1 minuto de parada em uma linha de montagem automotiva pode gerar uma perda de aproximadamente US$ 20.000,00. Por isso, manter uma planta, em funcionamento, sem a ocorrência de paradas, a alta produtividade e qualidade dos produtos finais é fundamental para as empresas nesse mercado.

O custo decorrente da falta de manutenção adequada em indústrias petroquímicas nos Estados Unidos consome aproximadamente 20 bilhões de dólares com perdas anuais (VENKATASUBRAMANIAN; RENGASWAMY; YIN; ET AL., 2003). O custo é ainda maior quando são incluídas situações semelhantes as que ocorrem em outras áreas da indústria, como a farmacêutica, metal mecânica, mineração, geradoras de energia, etc.

Não só as perdas financeiras é o principal prejuízo da deficiência na manutenção, mas também muitos acidentes de trabalho podem ocorrer. Um levantamento estatístico feito por (DE SOUZA; DE FREITAS, 2003) revela que 70% dos acidentes de trabalho ocorrido em ambientes industriais são causados por erros humanos, e que estão relacionados com a falta de manutenção. Tais eventos podem ocasionar problemas significativos de importância econômica, segurança e de impacto ambiental.

Conforme (VENKATASUBRAMANIAN; RENGASWAMY; YIN; ET AL., 2003), a ocorrência de grandes catástrofes e desastres em ambientes industriais é rara, entretanto aqueles de pequeno porte são muito comuns, pois ocorrem, pelo menos, uma vez ao dia, resultando em pessoas feridas, o que gera um custo de bilhões de dólares para a sociedade todos os anos.

Então, é imprescindível que as empresas estejam atentas à manutenção dos processos industriais, por meio de melhorias operacionais como fornecer ferramentas para auxiliar a equipe de manutenção, implantar gestão de qualidade, aperfeiçoar as estratégias e utilizar sistemas automáticos para a detecção, o diagnóstico e a predição de falhas nos equipamentos (LEE, J. ET AL., 2004).

Neste capítulo, será apresentada uma nova metodologia para aperfeiçoar as atividades de manutenção em plantas industriais. Essa metodologia aplica técnicas de tolerâncias a falhas, a fim de aumentar a confiabilidade dos equipamentos, uma vez que por intermédio disso, pode-se elevar os níveis de produtividade e disponibilidade do processo produtivo, além de reduzir custos relacionados à manutenção.

## 2.2 Análise dos Custos Envolvidos na Manutenção

Segundo (KARDEC; NASCIF, 2001) e (MARCORIN; LIMA, 2003), os custos envolvidos com as tarefas de manutenção geralmente não são vistos como muito relevantes pela maioria das empresas. Em muitos casos, os seus custos são analisados isoladamente, o que acaba impedindo a empresa de considerá-la em sua estratégia, colocando-a em segundo plano ou, mesmo, tratando-a como “um mal necessário”.

Entretanto, a importância do programa de manutenção é percebida quando se compara seus custos com os originados pela falta da manutenção.

Um programa de manutenção, quando aplicado com eficiência, pode até reduzir os custos de produção. Devido não apenas a redução nos custos de reparo da máquina, mas também no tempo de parada da produção.

A fim de se manter e também ganhar novos mercados é fundamental que a empresa esteja em um nível elevado de qualidade e produtividade. Essa busca envolve muitas variáveis, como as estratégias de gestão da qualidade, a escolha do melhor sistema produtivo, a capacitação de recursos humanos, etc. A manutenção tem um papel indispensável para garantir tanto a qualidade quanto a produtividade da empresa.

Conforme (KARDEC; NASCIF, 2001), a tarefa de manutenção deve ser encarada como função estratégica para obter melhores resultados. Levando a empresa a um nível superior de competitividade do ponto de vista da qualidade e produtividade.

A queda de produção é percebida com facilidade, pois dependente basicamente do desempenho dos equipamentos. A redução da produtividade, em função de paradas, pode ser pela falta ou ineficiência de manutenção. Devido à redução no desempenho, mesmo não ocorrendo uma parada efetiva do processo, pode ocorrer o aumento no tempo de produção e também a queda na qualidade dos produtos.

A falta de estratégias de manutenção acarreta queda na capacidade produtiva, paradas efetivas e redução da disponibilidade do equipamento. Uma estratégia adequada de manutenção deve manter a capacidade e a produtividade da máquina em taxas normais de operação. Com isso, é possível evitar paradas e criar condições para intervenções de reparos de forma rápida e eficaz no mesmo instante da falha.

Segundo (MIRSHAWKA; OLMEDO, 1999), os custos gerados pela manutenção são apenas a ponta de um *iceberg*. Essa ponta visível corresponde aos custos com mão de obra, ferramentas e instrumentos, material aplicado nos reparos, custo com contratação e outros referentes à instalação ocupada pela equipe de manutenção. No entanto, a falta de uma estratégia de manutenção gera custos adicionais, pois logo abaixo da parte visível do *iceberg*, estão os maiores custos que são decorrentes da indisponibilidade do equipamento no processo produtivo.

O custo da indisponibilidade é responsável pela perda de produção, baixa qualidade dos produtos, reformulação da produção e penalidades comerciais com possíveis consequências sobre a imagem da empresa (MIRSHAWKA; OLMEDO, 1999).

Na Figura 2.1, é apresentada uma análise propostas por (MIRSHAWKA; OLMEDO, 1999) que ilustra a relação entre o custo de realização de manutenção e o custo das falhas. Entre os custos decorrentes estão: as peças, a mão de obra de reparo e, principalmente, o custo da indisponibilidade do equipamento.

Esse gráfico revela que investimentos crescentes em manutenção tendem a reduzir os custos decorrentes das falhas (paradas da produção) e, por consequência, a diminuir o custo total da manutenção (soma dos custos de manutenção com os custos das falhas). Entretanto, nota-se no gráfico que a partir do nível ótimo com investimentos em manutenção, uma quantidade maior de investimento produz poucos benefícios para reduzir os custos de paradas e acaba elevando o custo total.

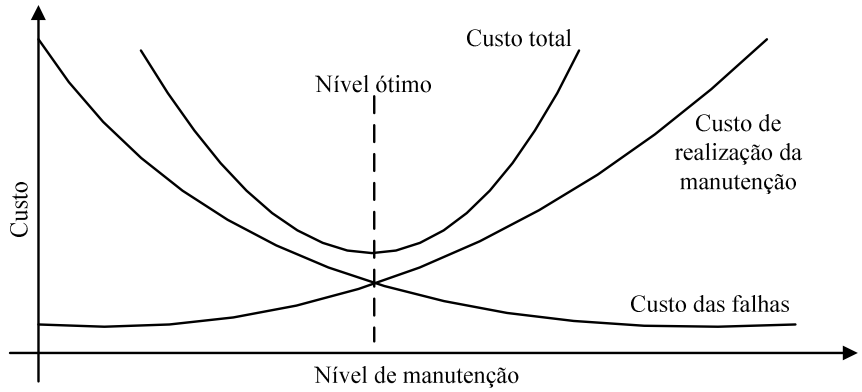



Figura 2.1: Gráfico do custo em relação ao nível de manutenção adotado (MIRSHAWKA; OLMEDO, 1999).

Na Figura 2.2, são apresentados os resultados para o cálculo do ponto ótimo de disponibilidade em relação ao lucro, estudado por (MURTY; NAIKAN, 1995). O gráfico mostra que a busca por um nível de 100% de disponibilidade (falhas em zero) requer gastos cada vez maiores com manutenção, causando a redução nos lucros.

Por isso, o grande desafio na gestão da manutenção é encontrar um ponto ótimo de disponibilidade, em que o custo da manutenção esteja em um nível capaz de gerar o máximo de lucros. Desse modo, a manutenção deve garantir a produtividade e o lucro com o menor custo possível.

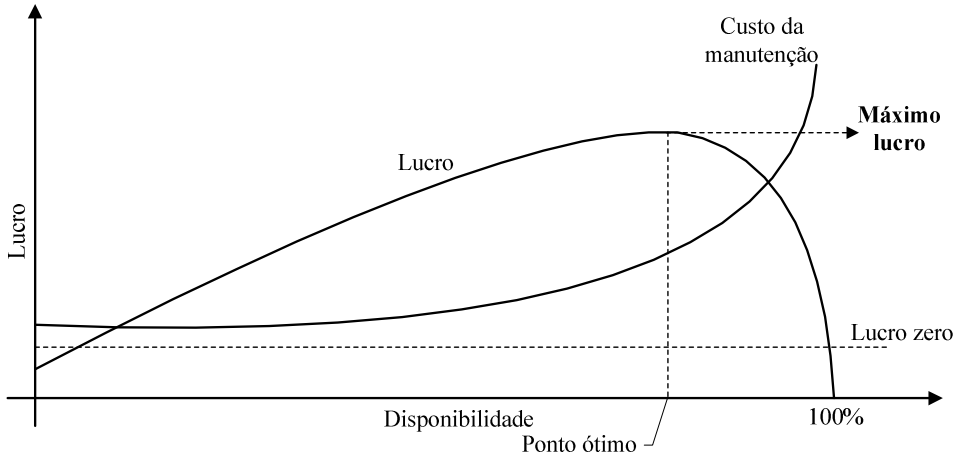



Figura 2.2: Gráfico da relação entre o lucro e disponibilidade (MURTY; NAIKAN, 1995).

Em (KARDEC; NASCIF, 2001) os custos da manutenção são classificados em três formas:

- Custos diretos: são utilizados para manutenção dos equipamentos em operação, por exemplo, inspeções regulares de lubrificação, custo de reparos ou revisões, custos de parada da produção para reparos, serviços de reforma ou modernização. Muitos desses custos estão diretamente ligados à estratégia de manutenção adotada.
- Custos de perda de produção: são oriundos da perda de produção, causados pela falha do equipamento principal e por não ter um de reserva para manter a unidade produzindo, ou pela falha do equipamento causada por uma ação imprópria da manutenção.
- Custos indiretos: são relacionados com a estrutura gerencial e de apoio administrativo, por exemplo, custos com análises e estudos de melhorias, engenharia de manutenção e supervisão, aquisição de novos equipamentos, instrumentação para manutenção, custos de amortização, depreciação, iluminação, energia elétrica e outras utilidades.

## 2.3 Estratégias de Manutenção

As estratégias de manutenção são metodologias ou políticas para gerenciamento das tarefas de reparos, da equipe técnica, das peças ou dos equipamentos em estoque e dos eventos relacionados a manutenção dos equipamentos de uma planta industrial.

Segundo (ENDRENYI ET AL., 2001) , o propósito da manutenção é estender o tempo de vida em funcionamento dos equipamentos. É esperado que uma política eficiente de manutenção possa reduzir a frequência de paradas e reduzir suas consequências. A manutenção é uma das ferramentas que assegura um nível satisfatório de confiabilidade ao equipamento (MARÇAL, 2000).

Na Figura 2.3, são apresentadas as estratégias de manutenção adotadas neste trabalho, que foram fundamentadas nas pesquisas de (ENDRENYI ET AL., 2001) e (KOTHAMASU ET AL., 2006). Elas são resumidas em quatro classes:

- Corretiva: os reparos são realizados somente depois da manifestação da falha.
- Preventiva: os reparos são realizados em intervalos de tempo predeterminados para reduzir a probabilidade da ocorrência de falhas.
- Preditiva: é realizada com base no monitoramento da “saúde” do equipamento, a fim de aplicar os reparos quando os parâmetros estiverem anormais, tornando a manutenção planejada.

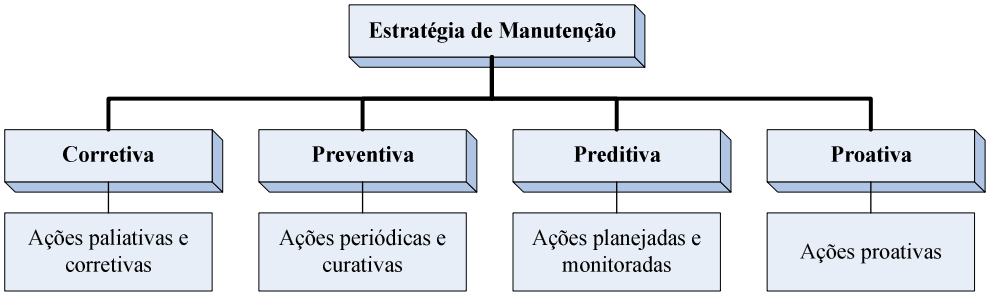



Figura 2.3: Estratégias de manutenção.

- Proativa: Além de monitorar e planejar os reparos tem o diferencial de reconfigurar o sistema e tomar decisões pela utilização de sistemas embarcados implantados nos equipamentos.

### 2.3.1 Manutenção Corretiva

Segundo (ENDRENYI ET AL., 2001) e (MARÇAL, 2000), a estratégia de manutenção corretiva é simples e direta, ou seja, quando uma máquina "quebrar", conserte-a. Esse método ("se não está quebrada, não conserte") tem sido aplicado em grande parte nas tarefas de manutenção nos equipamentos, desde a revolução industrial. É uma técnica puramente reativa que espera pela manifestação da falha e somente depois é tomada qualquer ação para reparo.

Os reparos são trabalhos realizados pela equipe de manutenção que visam restabelecer a função para qual a máquina foi especificada, eliminando o estado de falha (NUNES, 2001). Esses reparos podem ser intervenções para corrigir, provisoriamente, os danos e depois, em um tempo futuro, fazer um reparo definitivo (*paliativas*) ou intervenções típicas de reparos definitivos (*curativas*).

Atualmente, poucas indústrias usam uma filosofia puramente voltada para a gerência de manutenção corretiva. Em muitos casos, elas realizam tarefas com um misto de técnicas básicas de prevenção, como lubrificação e ajustes nas máquinas, mesmo dentro de um ambiente de manutenção corretiva.

Entretanto, nesse tipo de estratégia, as máquinas e outros equipamentos não são revisados nem são feitos grandes reparos até que o equipamento falhe por completo. Há empresas que inclusive aproveitam essas paradas para fazer uma revisão geral na máquina, que acaba estendendo o tempo de não produtividade.

Segundo (ALMEIDA, 2007), as empresas até encontram algumas vantagens utilizando essa estratégia como gerência de manutenção, que é o baixo gasto financeiro enquanto tudo está funcionando normalmente. Outra vantagem está no fato de as máquinas não precisarem ser desligadas para monitoramento ou inspeções periódicas.

A manutenção corretiva é o método mais caro entre as estratégias de gestão de manutenção. Os maiores custos e desvantagens associadas a essa estratégia são: altos custos de estoque de peças sobressalentes, trabalhos extras, tempo de máquina parada elevado e baixa disponibilidade de produção.

Nessa estratégia de gerência de manutenção, o planejamento de custos para os reparos é difícil de coordenar, pois as falhas são imprevisíveis. Assim, uma indústria que aplica somente essa estratégia deve ser capaz de reagir com rapidez a todas as possíveis falhas que ocorram. Além disso, por ser um método reativo, torna-se necessário ter um grande estoque de peças sobressalentes que incluem máquinas reservas ou, pelo menos, todos os principais componentes dos equipamentos críticos da planta industrial.

A manutenção corretiva pode ser uma solução interessante quando os custos da indisponibilidade do processo produtivo são menores que os custos necessários para evitar a falha. Essa condição é encontrada em equipamentos sem influência direta no processo produtivo (ALMEIDA, 2007).

Por fim, a manutenção corretiva é geralmente aplicada como um complemento às demais estratégias, que serão vistas a seguir.

### 2.3.2 Manutenção Preventiva

A manutenção preventiva consta de ações de reparo praticadas com a intenção de reduzir a probabilidade de ocorrência de falhas, mesmo em uma situação em que não se caracterizou como estado de falha.

Segundo (SHIKARI ET AL., 2004) e (KOTHAMASU ET AL., 2006), na estratégia de manutenção preventiva as ações de manutenção são planejadas. Por meio de inspeções realizadas periodicamente, tende-se a minimizar as paradas e reduzir a taxa de degradação, restaurando o sistema para o estado para o qual foi especificado.

Essa estratégia é acionada por tempo, ou seja, as tarefas de manutenção são baseadas no tempo gasto ou horas operacionais dos equipamentos. Um exemplo simples são as revisões de automóveis que precisam se feitas depois de uma determinada quilometragem.

Por exemplo, uma máquina nova e recém-instalada, tem uma alta probabilidade de falha devido à problemas na instalação que podem ocorrer durante as primeiras semanas de vida. Após esse período inicial, a probabilidade de falha é relativamente baixa por um prolongado período de tempo. Após esse período normal de vida da máquina, a probabilidade de falha tende a aumentar (ALMEIDA, 2007).

Na estratégia de manutenção preventiva, os reparos do equipamento são realizados de forma programada, com base em históricos e cálculos estatísticos sobre os dados referentes às horas operacionais.

A aplicação da manutenção preventiva corresponde a um conjunto de práticas, como planejamento de reparos, lubrificação, limpeza, ajustes, substituição de peças, etc. Essas ações práticas devem ser aplicadas em todas as partes consideradas como críticas na planta industrial.

O principal foco dessa estratégia é o planejamento da manutenção em relação ao tempo. Além disso, é assumido que as máquinas se degradarão em certo período de tempo, sendo este tempo uma informação particular de cada máquina.

Segundo (ALMEIDA, 2007), o problema dessa abordagem é que o modo de operação e as variáveis específicas da planta industrial afetam diretamente a vida operacional normal da máquina. O tempo médio entre as falhas pode-ser diferente para um mesmo tipo de equipamento, duas máquinas iguais fabricando produtos diferentes, dificilmente apresentarão o mesmo comportamento antes da ocorrência da falha.

Com isso muitas vezes pode ocorrer que se programe uma manutenção realizar um reparo desnecessário ou ocorrer uma falha catastrófica imprevista. Portanto, no primeiro caso a mão de obra e o material usado no reparo foram desperdiçados, pois as peças substituídas ainda teriam muito tempo de vida útil. No segundo caso, terá que aplicar os reparos usando técnicas corretivas, tornando-se muito caro. Assim, essa política, em muitos casos, pode levar a desperdícios, pois não considera a condição real do equipamento e eleva os custos operacionais (ALMEIDA, 2007).

Em resumo, essa estratégia é realmente eficiente nos casos de sistemas que sofrem algum tipo de degradação em um ritmo uniforme e previsível e para as quais os custos de uma falha sejam altos quando comparados aos custos da manutenção (MARCORIN; LIMA, 2003).

Um dos pontos críticos dessa estratégia de manutenção é que em muitos casos a quebra das máquinas ocorrerá quando as demandas de produção estiverem em níveis

elevados (no pior caso), pois os equipamentos são mais exigidos. Nessa situação, a equipe de manutenção precisa reagir com rapidez à falha inesperada, aplicando a manutenção corretiva.

A diferença mais importante entre manutenção corretiva e preventiva está na capacidade de planejamento dos reparos, para reduzir o impacto sobre a produção.

### 2.3.3 Manutenção Preditiva

A estratégia de manutenção preditiva visa aplicar o monitoramento contínuo dos parâmetros de controle, das condições reais de funcionamento, do rendimento operacional, dentre outros indicadores da planta industrial (ENDRENYI ET AL., 2001).

Segundo (KOTHAMASU ET AL., 2006) e (ALMEIDA, 2007), essa estratégia visa adquirir informações de variáveis do ambiente da máquina, a fim de estimar a probabilidade de comportamento correto e das leis de degradação, assegurando um intervalo máximo entre os reparos até uma eventual falha. Com isso, a equipe de manutenção pode programar intervenções e aquisições de peças, reduzindo gastos com estoque e evitando paradas desnecessárias da linha de produção.

Nessa estratégia, o objetivo é elevar a disponibilidade da produção à medida que não ocorram intervenções de reparos nos equipamentos. As intervenções são efetuadas somente quando o grau de degradação se aproximar ou atingir um limite previamente estabelecido (QIU, HAI ET AL., 2003). Isso é necessário somente quando os parâmetros monitorados indicam tal necessidade e aplica-se então a técnica de manutenção corretiva (agora planejada). Permite, portanto, uma preparação prévia do serviço de manutenção, além da possibilidade de se tomar outras decisões alternativas relacionadas com a produção do processo industrial.

De acordo com (ALMEIDA, 2007), uma das vantagens da manutenção preditiva é minimizar o número e os custos com paradas não programadas provocadas por falhas. Além de melhorar a produtividade, a qualidade do produto, o lucro e a eficiência da planta industrial, a manutenção preditiva possibilita à equipe planejar as atividades de manutenção com muito mais eficiência, levando em consideração as variáveis que influenciam no processo de produção.

Conforme (OTANI; MACHADO, 2008)**,** essa estratégia visa avaliar o estado do equipamento por meio de medições e monitoração contínua, está exige um custo adicional com mão de obra qualificada, aparelhos e instrumentos de medição. Poder ser realizada da seguinte forma:

- Monitoramento subjetivo: é exercido pelo pessoal de manutenção utilizando os cinco sentidos (tato, olfato, audição e visão). Por exemplo, quando um mecânico tocar uma parte da máquina, ele pode sentir a temperatura, a vibração, etc. É evidente que esse procedimento é diretamente ligado à experiência dos profissionais. Essa forma de monitoramento tem grande margem de erro.
- Monitoramento objetivo: é feito pela utilização de equipamentos ou instrumentos, para coletar informações na planta industrial. A técnica visa fornecer um valor de medição do parâmetro que está sendo acompanhado e esse valor medido é independente do operador do instrumento. Exige-se um treinamento da equipe de manutenção para manuseio adequado dos instrumentos.

- Monitoramento contínuo: é realizado por um acompanhamento *on-line* dos parâmetros de desempenho dos equipamentos pelo uso de sensores inteligentes para coleta de dados. São utilizados, muitas vezes, sistemas computacionais supervisórios para aquisições e visualização dos estados de funcionamento da planta industrial.

Dependendo da forma como essa estratégia de manutenção é aplicada, pode inicialmente parecer que há um custo elevado devido à implantação, mas esse custo é em longo prazo minimizado pelos resultados obtidos.

Pode-se dizer que a estratégia de manutenção preditiva é uma manutenção preventiva acionada por condições do equipamento. Em vez de ser puramente estatística, baseando-se na vida média da planta industrial para planejar os reparos, essa técnica usa o monitoramento das condições físicas do equipamento (p. ex., rendimento do sistema, desgastes mecânicos, temperatura, etc.).

As máquinas que compõem uma planta industrial são formadas por partes mecânicas que degradam-se a uma velocidade diretamente proporcional à gravidade do defeito, até que causem uma falha. Então, é fundamental que a degradação seja detectada com antecedência para que sejam realizados os devidos reparos e a manifestação da falha seja evitada.

A seguir, segundo (OTANI; MACHADO, 2008), são citadas algumas técnicas de monitoramento para coletar dados com o equipamento em pleno funcionamento ou com o mínimo de interferência possível no processo de produção:

- Monitoramento de vibração.
- Monitoramento de parâmetros.
- Monitoramento de lubrificantes.
- Monitoramente de ruídos.
- Monitoramento de sinais elétricos (corrente, tensão, torque).
- Monitoramento térmico.
- Monitoramento e inspeção visual.

Cada uma dessas técnicas fornece um conjunto de informações importantes para a equipe de manutenção. Entretanto, combinando essas técnicas de diversas formas pode-se elevar o poder de monitoramento e melhorar os resultados. A adoção das técnicas depende do tipo de equipamento, seu impacto sobre a produção, parâmetros operacionais da planta industrial e dos objetivos que se deseja alcançar com a estratégia de manutenção preditiva (OTANI; MACHADO, 2008).

A manutenção preditiva tem se apresentado como uma solução economicamente mais viável para o caso de falhas em máquinas e equipamentos industriais em relação às outras estratégias de manutenção (ENDRENYI ET AL., 2001).

Além disso, a utilização dessa estratégia de manutenção apresenta benefícios, como prolongar a operação do processo produtivo, prevenir a ocorrência de falhas e paradas, redução de estoques para reparos, redução de acidentes de trabalho e otimização da equipe de manutenção. Como desvantagens dessa estratégia citam-se: disponibilidade de profissionais qualificados, equipe de manutenção em constante atualização tecnológica e inspeções periódicas nos equipamentos (OTANI; MACHADO, 2008).

Existe uma variação da manutenção preditiva conhecida como *Manutenção Baseada na Condição* (CBM – *Condition Based Maintenance*) (LEE, JAY ET AL., 2006), em que as ações de reparos pela equipe de manutenção são tomadas com base nas condições monitoradas na planta industrial.

A principal vantagem dessa estratégia é que a planta industrial é avaliada enquanto seus equipamentos e máquinas estão em funcionamento normal, por meio da utilização de sensores e testes não invasivos. Após as análises das informações coletadas, é possível saber se determinado equipamento ou máquina precisa ou não de reparos. Essa estratégia de manutenção pode ser implementada de forma automática por um sistema computacional embarcado, que será apresentado nas próximas seções.

### 2.3.4 Manutenção Proativa

Com o aumento da demanda de clientes para produção, concorrência externa e busca por inovações tecnológicas, as empresas têm buscado melhorar os índices de produtividade. Isso foi possível em grande parte, devido à automação industrial que proporcionou a elevação da eficiência e a realização de mais produtos com menos recursos, e à evolução tecnológica da eletrônica e computação, que trouxeram a disseminação e redução de custos para projetar equipamentos industriais totalmente automatizados e com elevados índices de produtividade (VENKATASUBRAMANIAN; RENGASWAMY; YIN; ET AL., 2003).

No entanto, devido a essa exigência do mercado, esses equipamentos industriais passaram a ser mais complexos, sofisticados e totalmente interligados entre si, portanto mais suscetíveis a falhas. Devido a isso, estratégias de gerenciamento de manutenção passaram a ser mais exigidas. Utilizando apenas as estratégias de manutenção tradicionais como a corretiva ou preventiva, as equipes de manutenção não têm conseguido atingir resultados satisfatórios para melhorar os índices de produtividade e disponibilidade do processo produtivo.

(LEE, JAY ET AL., 2006) diz que esses equipamentos industriais complexos apresentam degradação nas partes que geralmente, são invisíveis aos olhos humanos da equipe de manutenção que utiliza a estratégia preditiva. Contudo, monitorar a evolução dessa degradação por sensores sofisticados e sistemas computacionais capazes de adquirir e processar essas informações, analisando o estado de operação e desempenho do equipamento, tornou-se possível devido ao desenvolvimento tecnológico.

Segundo (KOTHAMASU ET AL., 2006) e (MULLER ET AL., 2008), os problemas na aplicação das estratégias de manutenção corretiva ou preventiva nesses novos equipamentos é que ainda podem ocorrer paradas não programadas no processo de produção, provocando falhas. Desse modo, a idéia é quantificar e monitorar os desgastes da máquina para realizar a manutenção e reparos antes da manifestação da falha, e não depois.

Isso pode ser feito por meio de medições dos desgastes (ou degradação) da máquina e predição de falhas futuras, tomando como premissa a condição atual de operação e histórico de informações, como variáveis de desgastes, falhas passadas e tarefas de manutenção (MULLER ET AL., 2008). Como resultado da predição, alcança-se a minimização dos custos com manutenção, associados a problemas operacionais. Simultaneamente também minimiza-se o risco de paradas não programadas do processo produtivo.

Devido a essas transformações tecnológicas nas plantas industriais, passou-se a exigir novos requisitos para garantir um nível elevado de eficiência no processo produtivo. Surge uma nova estratégia para realização da manutenção, a estratégia de *manutenção proativa*, que também é conhecida como *manutenção inteligente* (LEE, JAY, 2003) (HUANG, R. ET AL., 2007) (DJURDJANOVIC ET AL., 2003) (MULLER ET AL., 2008) (JARDINE ET AL., 2006).

A manutenção proativa é um avanço em relação às estratégias de manutenção tradicionais, pois além de serem aplicados todos os conceitos referentes à manutenção preditiva e preventiva, como o monitoramento do equipamento em função do desgaste ocorrido, pode-se diagnosticar, quantificar a perda de desempenho e também atuar (reconfigurar) sobre o sistema que é analisado (MULLER ET AL., 2008).

Trata-se de uma abordagem de manutenção que continuamente trabalha em busca de indícios de degradação e infere indicadores de comportamentos futuros sobre a condição do equipamento. A predição de falhas deve ser realizada dentro de um tempo aceitável, antes da manifestação da falha; além disso, deve-se detalhar exatamente quais componentes (partes da máquina) provavelmente falharão (LEE, JAY ET AL., 2006).

Devido a essas razões, em (LEE, JAY, 2003), é proposta uma metodologia que avança para além de uma abordagem de manutenção tradicional (detecção e quantificação de falhas) para uma nova abordagem agora centrada em avaliação e predição do nível de degradação de um processo, máquina, etc. Nessa metodologia, o nível de degradação é tratado como um indicador para uma possível falha e é utilizado para prever um nível inaceitável de desempenho no processo antes da falha ocorrer. Com base nessa ideia, foi possível, a partir de uma estratégia de manutenção “falha e conserta”, evoluir para uma nova, “prever e prevenir”.

Segundo (LEE, JAY ET AL., 2006), a manutenção proativa é viabilizada pelo uso de sistemas embarcados, que monitoram os equipamentos da planta industrial remotamente, fazendo a aquisição de sinais indicativos da condição de funcionamento, analisando e processando as informações que possibilitam evoluir da estratégia de manutenção preditiva para a proativa.

Assim, a manutenção proativa é aquela que envolve tarefas de monitoração de desgastes, diagnóstico de falhas, quantificação de desempenho e reconfiguração do equipamento. Os objetivos dessa estratégia de manutenção é prover informações importantes e úteis para melhorar o planejamento da equipe de manutenção em realizar os devidos reparos. Caso não haja tempo suficiente para reparos em uma determinada máquina, a planta pode ser reconfigurada automaticamente (quando detectar uma provável falha) até a realização dos reparos para reduzir os prejuízos, sem paralisar a produção, mesmo operando de maneira degradada.

Pelo uso da estratégia de manutenção proativa, é possível alcançar níveis de eficiência elevados no processo produtivo. Seguem algumas das vantagens da utilização dessa estratégia (OTANI; MACHADO, 2008) (HUANG, R. ET AL., 2007) (LEE, JAY ET AL., 2006):

- Redução de custos com manutenção e eliminação de reparos desperdiçados.
- Redução de peças e equipamentos em estoque.
- Diminuição das paradas do processo produtivo, aumentando a disponibilidade.

- Previsibilidade para vida útil do equipamento e da ocorrência das falhas.
- Disponibilização de informações precisas e em tempo real sobre a condição da planta industrial.
- Planejamento da manutenção.
- Plantas industriais ficam mais seguras e preservadas.

As possíveis desvantagens dessa estratégia de manutenção:

- Requer uma equipe de manutenção mais capacitada e treinada com as novas tecnologias.
- Cada aplicação dessa estratégia de manutenção é desenvolvida unicamente para um determinado equipamento.
- Custo inicial alto para implantação.
- A implantação exige um maior investimento de recursos e de tecnologia, quando comparado com as outras estratégias.
- Falhas no próprio sistema de monitoramento da planta podem causar prejuízos incalculáveis.

A implantação de uma estratégia proativa tem inicialmente um espaço de projeto grande, como a escolha de sensores, ferramentas de processamento de sinais, definição de plataforma de sistema embarcado, definição de ferramentas de monitoramento, viabilidade técnica, viabilidade econômica e treinamento da equipe de manutenção (DJURDJANOVIC ET AL., 2003).

A manutenção proativa tem sido apresentada como uma solução inovadora para melhorar a gestão de manutenção em plantas industriais com sistemas produtivos complexos e críticos. Pela utilização de sistemas embarcados pode-se monitorar os desgastes nos equipamentos, fazer detecção, diagnóstico e predição das falhas.

O sistema embarcado que implementa a estratégia de manutenção proativa, a partir de agora, será chamado de *Sistema de Manutenção Inteligente* (SMI). Esse sistema é formado por um sistema computacional embarcado que realiza todos os processamentos necessários para aquisição de sinais, detecção, diagnóstico de falhas, predição da vida útil e reconfiguração do equipamento monitorado. O SMI também é construído com base na estratégia de Manutenção Baseada na Condição (CBM) e será apresentado na próxima seção.

## 2.4 Sistema de Manutenção Inteligente

Um Sistema de Manutenção Inteligente (SMI) é baseado em técnicas da estratégia de manutenção *proativa*. É uma ferramenta computacional que implementa uma metodologia para melhorar o gerenciamento das operações industriais no quesito de manutenção da planta industrial (LEE, JAY, 2003).

Nas modernas plantas industriais do século XXI, estão inclusos diversos sensores, equipamentos de controle e automação muito sofisticados. Devido a isso, o desempenho computacional de tais equipamentos eletrônicos em realizar suas tarefas, também tem sido mais exigido.

Essa tecnologia possibilitou reaproveitar uma parte do processamento para medir o desempenho da máquina por indicadores e avaliar essas informações. Tomando isso como princípio, um SMI é baseado no monitoramento e na avaliação da condição da planta industrial ou do equipamento, sendo uma ferramenta apropriada e eficiente para aumentar a disponibilidade do processo produtivo. Isso é feito pela redução das paradas do processo para próximo de zero, diminuindo as ocorrências de falhas (MULLER ET AL., 2008).

Segundo (DJURDJANOVIC ET AL., 2003)**,** um SMI implantado em um equipamento pode economizar na ordem de 20% ou mais de custos com paradas do processo produtivo, elevando o nível de qualidade e disponibilidade, redução de estoques de peças, etc.

Em um SMI, estão inclusas habilidades para monitorar as plantas industriais, a fim de detectar e diagnosticar falhas, e prever a perda de desempenho. Como saídas do sistema, estão as informações que complementam as tomadas de decisões pela equipe de manutenção, que visa planejar as tarefas de manutenção e produção para sincronizar com os melhores momentos para reparos, sem prejudicar o sistema produtivo com as interrupções.

As principais funções e objetivos alcançados pelo SMI são:

- Prover transparência, consistência e troca de informações de modo automático para equipe de manutenção e setores gerenciais.
- Aumentar a utilização dos recursos da planta industrial por meio da estratégia de manutenção proativa e preditiva.
- Otimização da planta industrial, aumentando a disponibilidade produtiva.

Segundo (JARDINE ET AL., 2006)**,** a confiabilidade é um aspecto importante para avaliação de processos e plantas industriais. Não importa quão bom seja o projeto de um produto que será fabricado, se o produto se desviar do especificado ao longo do tempo devido à deterioração no processo produtivo. Portanto, para assegurar um nível satisfatório de confiabilidade nos equipamentos industriais, têm sido investido mais recurso na tarefa de manutenção como um meio estratégico.

No presente trabalho, o SMI tem sua base sobre a metodologia da Manutenção Baseada na Condição (CBM – *Condition-Based Maintenance*), que apresenta uma arquitetura padrão de construção e uma série de etapas necessárias para o projeto desse sistema.

### 2.4.1 Metodologia de Manutenção Baseada na Condição (CBM)

A CBM (*Condition-Based Maintenance*) (THURSTON, M.G., 2001) (LEBOLD; THURSTON, M., 2001) (BENGTSSON, 2004) é uma estratégia de manutenção que trabalha de forma a recomendar tarefas de reparos, que são baseados nas informações coletadas pelo monitoramento da condição dos equipamentos. Dessa forma, objetiva evitar reparos desnecessários por meio da solicitação somente quando existir uma evidência de comportamento anormal nos equipamentos. Como resultado, pode significativamente reduzir o custo de manutenção pela diminuição do número de manutenções ou reparos desnecessários que são realizados nos equipamentos.

Segundo (JARDINE ET AL., 2006) (HUANG, R. ET AL., 2007), uma abordagem CBM consiste de três etapas-chave:

- Aquisição de dados: coletar e obter dados relevantes para análise da saúde do sistema em monitoramento.
- Processamento dos dados: filtrar e analisar os dados ou sinais coletados para um melhor entendimento e interpretação.
- Tomada de decisão de manutenção: computar as informações e recomendar tarefas de manutenção de forma eficiente para reparos no equipamento monitorado.

Basicamente, a etapa “Tomada de decisão” visa aplicar técnicas de tolerância a falhas, em que são utilizadas as seguintes técnicas: detecção de anormalidades, diagnóstico de falhas, predição de falhas e monitoramento da condição. Cada uma dessas técnicas será detalhada no capítulo a seguir.

#### *2.4.1.1 Aquisição de dados*

A aquisição de dados é o processo de coleta e armazenamento da informação útil do equipamento monitorado. Essa é uma tarefa essencial dentro do sistema CBM.

As informações coletadas incluem eventos como: “o que aconteceu?”, “o que foi feito nos equipamentos?”, medidas de variáveis de condição ou estado de saúde dos equipamentos que são coletados por sensores inteligentes.

Essas informações são compostas por dados que representam a condição do equipamento, por exemplo: vibração, acústica, análise de lubrificantes, temperatura, pressão, umidade, clima, corrente elétrica e outras variáveis.

Além disso, podem ser utilizadas outras informações que estejam disponíveis sobre a manutenção (p. ex., sistemas de informações gerenciais de manutenção), como o banco de dados com históricos dos sensores e a condição de todo o equipamento ou planta industrial.

#### *2.4.1.2 Processamento de dados*

Nesta etapa do processamento de dados (ou sinais), é feita a redução de ruídos por meio da filtragem. Isso é importante, pois os dados coletados geralmente apresentam sinais de ruídos e até erros que prejudicam a boa interpretação dos dados. Como no sistema CBM, os dados são utilizados para monitoramento da condição e podem apresentar erros derivados de falhas presentes nos sensores.

Na próxima etapa desse processamento, é feita a análise dos sinais. Uma grande variedade de modelos, algoritmos e ferramentas estão disponíveis na literatura para essa tarefa, que visa melhorar o entendimento e compreensão dos dados coletados. A escolha da ferramenta ideal depende principalmente do tipo de sinal e o que pretende-se extrair dele.

De acordo com (JARDINE ET AL., 2006), os sinais coletados para monitoramento da condição são classificados em três tipos:

- Valor: os dados coletados de uma variável para um tempo específico são simplesmente valores reais. Por exemplo, temperatura, pressão, corrente elétrica, etc.

- Forma de onda: os dados coletados de uma variável para um tempo específico são séries temporais, que formam uma onda completa em um período de tempo. Por exemplo, dados de vibração, acústicos, torque, etc.
- Multidimensional: os dados coletados de uma variável para um tempo específico são multidimensionais. Por exemplo, imagens térmicas, raio X , fotos, etc.

O processamento para os dados na categoria *forma de onda* e *multidimensional* também é chamado de *processamento de sinais*. Várias técnicas para o processamento de sinais foram desenvolvidas, a fim de analisar tais sinais e extrair características que auxiliem no processo de detecção, diagnóstico e predição de falhas. A seguir, algumas das técnicas para o processamento de sinais são apresentadas conforme (JARDINE ET AL., 2006):

- Análise no domínio de tempo: baseada no tempo da onda do sinal. Calcula características do tempo do sinal, utilizando estatísticas, tais como: médias, pico, intervalo pico a pico, desvio padrão, etc. Por exemplo: modelo AR (*autoregressive*), ARMA (*autoregressive moving average*), PCA (*principal component analysis*), etc.
- Análise no domínio de frequência: baseada na transformação do sinal para o domínio de frequência. A vantagem é a habilidade de facilmente identificar e isolar certos componentes de frequências que forem de interesse da análise. Por exemplo, FFT (*fast fourier transform*).
- Análise no domínio de tempo-frequência: uma limitação da análise domínio de frequência é a inabilidade de lidar com sinais de ondas não estacionárias, que são muito comuns em plantas industriais, especialmente em análise de falhas. Assim, a análise de tempo-frequência investiga as ondas do sinal no domínio do tempo e da frequência. Por exemplo, transformada *wavelets* e transformada *wavelet packets*.

#### *2.4.1.3 Tomada de decisão de manutenção*

Esta etapa fornece suporte para a tomada de decisão e auxilia a equipe de manutenção na realização de tarefas de reparos dos equipamentos. Em um sistema CBM, as técnicas para suporte à tomada de decisão são: detecção, diagnóstico e predição.

A detecção revela o momento quando uma condição anormal ocorre. O diagnóstico tem a função de isolar e identificar a falha ocorrida. A predição antecipa as falhas antes de elas ocorrerem.

Obviamente, a predição aparenta ser superior à detecção e diagnóstico, já que pode prevenir as falhas e preparar a equipe de manutenção para reparos planejados. Com isso, é possível solucionar muitos dos problemas que podem ocorrer nos equipamentos, devido a paradas não planejadas, economizando custos extras.

Todavia, a predição não pode substituir o diagnóstico por completo, pois, na prática, nem todas as falhas são previstas. A predição não é uma técnica com 100% de certeza em acertos. O diagnóstico é uma ferramenta complementar importante, que tem a finalidade de fornecer suporte à tomada de decisão de manutenção quando a predição for errada, assim o diagnóstico consegue identificar a nova falha ocorrida

(VENKATASUBRAMANIAN; RENGASWAMY; YIN; ET AL., 2003) (MULLER ET AL., 2008).

Essa etapa que envolve as técnicas de tolerância a falhas será detalhada no próximo capítulo deste trabalho.

### 2.4.2 Arquitetura OSA-CBM

O OSA-CBM (*Open Systems Architecture for Condition-Based Maintenance*) (LEBOLD; THURSTON, M., 2001), é uma proposta de padronização para construção de uma arquitetura para o desenvolvimento de sistemas de manutenção inteligente e é baseada na ideia de camadas que se relacionam entre si. Este propõe facilitar a integração e interoperabilidade de componentes entre diferentes fabricantes de equipamentos. Na Figura 2.4, é apresentado o modelo da arquitetura em camadas do padrão OSA-CBM.

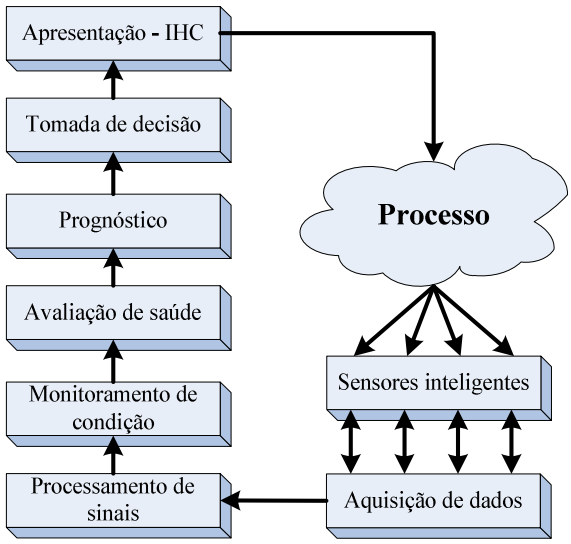



Figura 2.4: Arquitetura em camadas do padrão OSA-CBM.

Essa arquitetura foi projetada para reutilizar os padrões de comunicação distribuída entre computadores e, para isso, foi adotado o padrão de comunicação em redes (p. ex., Internet). O modelo de computação adotado abre um grande leque de tecnologias atuais que podem ser utilizadas para troca de informações entre os componentes.

Cada uma das camadas da arquitetura OSA-CBM (Figura 2.4) tem uma tarefa na manutenção (LEBOLD; THURSTON, M., 2001):

- Sensores inteligentes: formada por diversos sensores sofisticados e estrategicamente posicionados em partes críticas do processo (equipamentos ou máquinas), a fim de coletar sinais precisos para o monitoramento da condição.
- Aquisição de dados: módulo responsável por ler os sinais dos sensores e transformá-los a partir de grandeza física em uma grandeza elétrica, disponibilizando uma informação útil para o restante do sistema. Também pode ser responsável pelo armazenamento e formação de banco de dados de históricos.

- Processamento de sinais: aplica os primeiros cálculos sobre os dados adquiridos, utilizando técnicas de processamento de sinais para filtragens e transformações.
- Monitoramento de condição: etapa responsável pela detecção de anormalidades na condição de funcionamento do processo. São comparadas as características extraídas (sensores) com as previamente treinadas (modelo), que visam avaliar a situação do processo. Por exemplo, valores esperados, limites de operação ou indicadores de condição (nível baixo, nível normal ou nível alto).
- Avaliação de saúde: fase que realiza o diagnóstico das anormalidades detectadas pelo monitoramento da condição. Essa fase tem como propósito identificar as partes da máquina ou equipamentos da planta industrial que apresentam falhas. Além disso, faz uso de informações sobre tendências, históricos e regras de conhecimentos de especialistas para diagnosticar e pode armazenar os resultados formando um banco de dados com históricos da saúde do equipamento.
- Prognóstico: tem a função de realizar uma previsão da saúde futura do equipamento monitorado, estimar a vida útil restante ou calcular uma taxa de probabilidade para um determinado tempo antes da falha se manifestar. Ademais, faz uso de técnicas estatísticas, redes neurais e conhecimentos de especialistas para criar um modelo de tendência para acompanhar a condição de operação do processo.
- Tomada de decisão: fornece suporte à tomada de decisão para a equipe de manutenção com base nos resultados obtidos nas camadas anteriores. Também fornece ações mais recomendadas para solucionar as falhas ocorridas, que podem ser, por exemplo, agendar as manutenções, reconfigurar o equipamento ou a planta, emitir um alarme de emergência, tolerar a falha até se tornar mais crítica, informar relatório sobre as falhas, etc.
- Apresentação – IHC: é a interface homem-máquina, para fornecer as informações de saída desse sistema, que pode ser de forma única por meio de uma tela de computador para um especialista em manutenção, ou de forma distribuída pela Internet. Essa etapa objetiva produzir as saídas para realimentar o processo de modo a solucionar as falhas ocorridas pela equipe de manutenção.

No presente trabalho, foi tomado o padrão OSA-CBM como base para o desenvolvimento do sistema embarcado para manutenção inteligente, que será apresentado nos próximos capítulos.

### 2.4.3 Watchdog Agent ™

O *Watchdog Agent*™ (*WA*) (DJURDJANOVIC ET AL., 2003) (LEE, J. ET AL., 2004) é uma proposta acadêmica para um Sistema de Manutenção Inteligente do grupo *IMS Center*[3] (*Center for Intelligent Maintenance Systems*), localizado nos EUA e

[3] Site oficial do grupo *IMS Center* – www.imscenter.net

vinculado com as universidades de Cincinnatti, Missouri-Rolla e Michigan e empresas colaboradoras, como a *Boeing*, *Toyota*, *Siemens*, *Carterpillar*, *AMD*, entre outras.

O *WA* é um conjunto de ferramentas que possibilita o monitoramento em tempo real de um equipamento em uma planta industrial. É desenvolvido sobre uma plataforma composta por um computador industrial, em que foram implementadas as ferramentas com a intenção de avaliar e predizer o desempenho dos equipamentos. As ferramentas são desenvolvidas em MATLAB.

O processamento de informações pelo *WA* é composto por quatro camadas: *processamento de sinais*, *extração de características*, *avaliação de desempenho* e *fusão de sensores*. Ele foi desenvolvido para manter compatibilidade com a arquitetura do padrão OSA-CBM, a fim de possibilitar a integração de novas ferramentas com outros dispositivos compatíveis com esse modelo. Além disso, tem a capacidade de fazer a aquisição de dados por meio de leituras nos sensores instalados na planta industrial.

De acordo com Jay Lee em (DJURDJANOVIC ET AL., 2003), o idealizador do *WA*, este foi projetado com objetivo de realizar a tarefa de avaliação da degradação no desempenho, tomando como base a leitura de sensores que medem as propriedades críticas dos equipamentos. Assumiu-se que os sensores eram sensíveis o suficiente para detectar alterações da variável de degradação, proporcionando, desse modo, a avaliação e quantificação da degradação.

Para possibilitar o monitoramento do desempenho, o *WA* precisa adquirir algum conhecimento, *a priori*, sobre o processo de degradação do equipamento. Nessa ferramenta, é possível utilizar para análises de comportamento um modelo matemático ou conhecimentos de especialistas, ou registros de dados históricos.

Como o *WA* ainda está em desenvolvimento, as tarefas de tolerância a falhas ainda não estão completas. Todavia, no projeto atual já está prevista a inserção de elementos inteligentes capazes de realizar as três tarefas básicas da tolerância a falhas em equipamentos industriais (DJURDJANOVIC ET AL., 2003):

- Avaliação quantitativa do desempenho da degradação (*detecção de anormalidades*).
- Diagnóstico do desempenho da degradação atual ou prevista (*diagnóstico de falhas*).
- Predição do desempenho da degradação (*predição de falhas*).

A tarefa de *detecção* já está implementada pela avaliação de comportamentos atuais comparado a comportamentos normais.

A tarefa de *predição* será realizada pela análise de tendência de comportamento ou um modelo matemático da dinâmica de comportamento observada no equipamento, o que permite fazer a predição de comportamentos futuros para o equipamento.

A tarefa de *diagnóstico* será realizada por meio de aprendizado de padrões do comportamento, a fim de reconhecer situações que foram observadas no passado ou estar ciente de situações que nunca foram observadas anteriormente.

Como saída, quando o *WA* estiver implantado em um processo industrial, fornecerá informações para auxiliar na tomada de decisão da equipe de manutenção.

Vale destacar que as etapas de, a aquisição de dados, de processamento de sinais, de extração de características e avaliação de desempenho já estão implementadas e validadas no *WA*. Nos experimentos deste trabalho, serão reutilizadas as etapas de processamento de sinais e a extração de características.

## 2.5 Estudo de Caso: Transporte de Combustíveis para Petrobras

A Petrobras Transporte S.A, conhecida por Transpetro[4], subsidiária integral da Petrobras, segundo informações obtidas no site oficial é a principal empresa de logística e transporte de combustíveis do Brasil. Atende às atividades de *transporte* e *armazenamento* de petróleo e derivados, álcool, biocombustíveis e gás natural.

A Transpetro é responsável pelo gerenciamento de uma rede dutoviária formada por mais de 11 mil km de dutos (oleodutos e gasodutos) que interligam todas as regiões brasileiras e abastecem os mais remotos pontos do país.

As redes de dutos são interligadas a terminais terrestres e marítimos, unindo as áreas de produção, refino e distribuição da Petrobras e atuando na importação e exportação de petróleo e derivados, de biocombustíveis e de gás natural. Nessa estrutura dutoviária, passam anualmente bilhões de litros de combustíveis. É formada por 7 mil km de oleodutos, 4 mil km de gasodutos, 20 terminais terrestres, 26 terminais aquaviários e uma frota de 54 navios-petroleiros. Na Figura 2.5, é apresentada a logística que a Transpetro utiliza no Brasil, extraída do site oficial da empresa.

Para gerenciar toda essa infraestrutura, a empresa criou o Centro Nacional de Controle Operacional (CNCO), localizado no Rio de Janeiro. Esse centro visa monitorar e controlar todas as operações de transporte dutoviário para centralizar as operações. proporcionando aumento da eficiência e da segurança operacional, assim como a redução de custos.

Todas as instalações dutoviárias espalhadas pelo país são interligadas a uma rede de comunicação que interage com um sistema supervisório (SCADA – Sistemas de Supervisão e Aquisição de Dados). Isso possibilita monitorar as informações operacionais. Ademais, permite que os técnicos de operação tenham informações precisas, mensagens de alarmes e comandem as válvulas e os equipamentos (p. ex., ligar e desligar bombas, abrir e fechar válvulas, alterar fluxo nas malhas, além de detectar vazamentos e realizar as simulações operacionais).

Dentro de toda a logística de abastecimento do Sistema Petrobras, é dada uma atenção especial às operações de terminais e oleodutos, pois é onde flui a produção da empresa até chegar aos consumidores. O transporte dutoviário possibilita agilidade, segurança e capacidade de fluxo na movimentação dos combustíveis entre essas partes.

Dos campos de produção, o petróleo é transportado, por oleodutos e/ou por navios, para os terminais e de lá até as refinarias. Após o refino, os derivados são novamente escoados por dutos aos terminais aquaviários e terrestres para ser entregues, por dutos e navios, às companhias distribuidoras, chegando aos clientes nos mercados nacionais e internacionais.

---

[4] Site Transpetro – www.transpetro.com.br

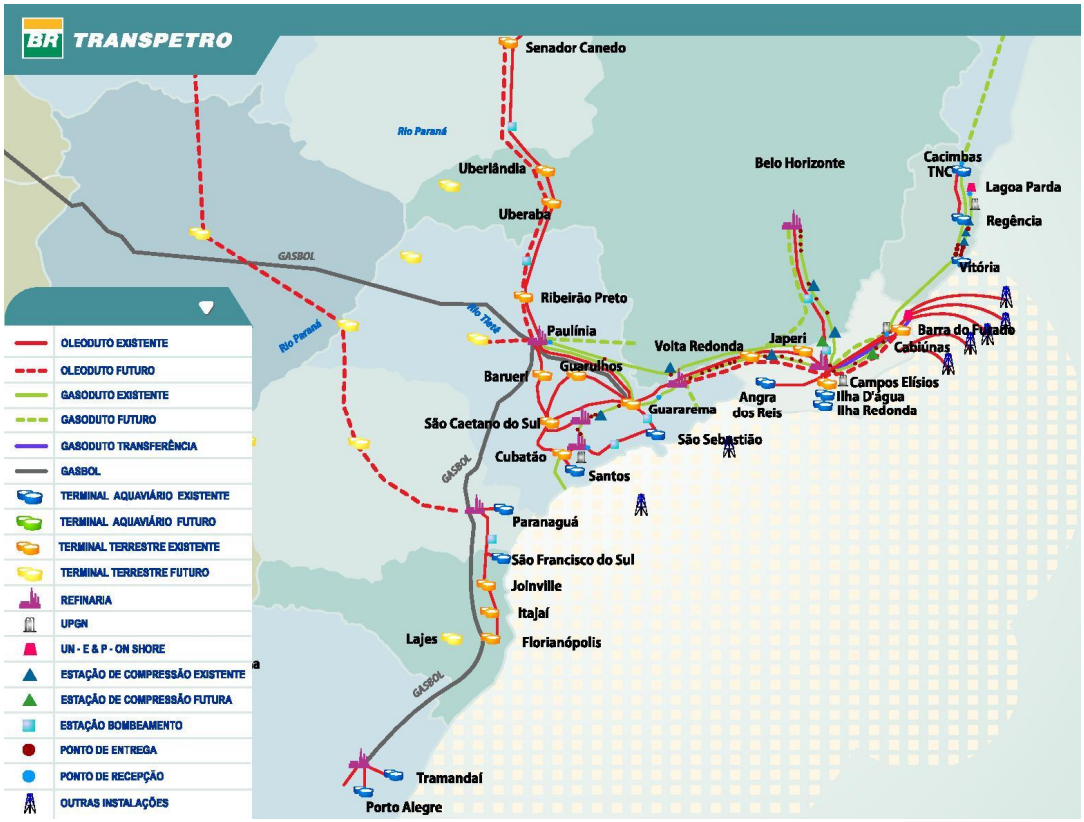



Figura 2.5: Mapa de dutos e terminais da Transpetro no Brasil.

Nessa grande logística de fornecimento em todo o Brasil, os dutos são o meio de transporte preferencial tanto para atender o abastecimento das refinarias como para suprir a necessidade dos grandes centros consumidores de combustíveis.

Em função da grande importância estratégica que se encontra o sistema de transporte dutoviário para Petrobras, a Transpetro investiu na criação de dois centros de pesquisas especializadas em manter esse sistema:

- Centro de Tecnologia de Dutos (CTDUT): visa à promoção do desenvolvimento de tecnologias de inspeção, manutenção e reparo de oleodutos, além de formar mão de obra especializada. Possui um laboratório de pesquisa em escala real para o desenvolvimento de novas tecnologias em dutos, testes de produtos, equipamentos e sistemas de proteção ambiental.
- Centro Nacional de Reparo de Dutos da Petrobras (Creduto): surgiu da necessidade estratégica de gerar capacitação e recursos próprios para reparos e outros tipos de intervenção em dutos com qualidade, segurança e custos adequados à atividade dutoviária. Seu objetivo é capacitar e manter disponíveis recursos humanos e materiais para a execução de reparo em dutos terrestres.

Vista a dimensão continental desse sistema de logística percebe-se a dificuldade que a equipe de manutenção deve ter para atender a todas as falhas ocorridas e realizar reparos nas partes desse sistema.

As manutenções realizadas na rede dutoviária são realizadas pela Transpetro por meio de recursos humanos próprios formados pelos seus dois centros (CTDUT e

Creduto). A equipe de manutenção aplica na prática a estratégia preventiva, corretiva e, às vezes preditiva, que depende do equipamento alvo da manutenção.

Em todos os milhares de quilômetros que formam a rede dutoviária, durante seu percurso, podem ser encontradas centenas ou até milhares de válvulas. Então, o presente trabalho focou na ideia de implantar, a princípio, um Sistema de Manutenção Inteligente nas válvulas, que são operadas por um atuador elétrico (motor).

A válvula foi escolhida como estudo de caso por ser um equipamento importante dentro do processo de transporte de derivados de petróleo. No futuro, a ideia é expandir o Sistema de Manutenção Inteligente para outras partes do processo.

Essas válvulas são equipamentos industriais que tem a finalidade de controlar o fluxo de fluídos nos dutos. Seu trabalho é simples: abrir ou fechar o obturador de fluxo nos dutos.

Em uma válvula, podem ocorrer as seguintes falhas do ponto de vista de passagem de fluxo nos dutos:

- Bloquear o fluxo: ocorre quando a válvula está aberta e devido a um motivo desconhecido o fluxo nos dutos é interrompido pelo acionamento do obturador.
- Abertura para fluxo: acontece quando a válvula está fechada (obturador bloqueia o fluxo) e devido a um motivo desconhecido o obturador é aberto e o fluxo é liberado nos dutos.
- Fluxo parcial: ocorre quando a válvula não está nem fechada, nem aberta em sua totalidade, nesse caso, o fluxo é liberado parcialmente nos dutos.

Essas falhas têm as mais diversas causas que variam de defeitos internos na válvula (desgastes nas engrenagens, umidade, corrosão, etc.), defeitos no atuador elétrico (rompimento de bobina do motor, rolamentos, curtos circuitos, umidade, sobre torque, etc.) até defeitos internos no obturador (obstrução do obturador, sedimentos na base, vazamentos, etc.). O domínio de estudos para as falhas é diverso e serão apresentados nos próximos capítulos mais detalhes.

Quando as válvulas falham de forma indesejada, podem causar grandes prejuízos no processo dutoviário. Por exemplo, a interrupção de um gasoduto pode cessar o processo de refino e gerar prejuízos financeiros altíssimos, em função do tempo de parada na produção. Esse tipo de falha pode causar perdas financeiras e acidentes inimagináveis. Para ilustração, são apresentados, na Figura 2.6, alguns casos reais de dutos utilizados para transporte dos derivados de petróleo nas refinarias da Petrobras. Na Figura 2.7, são apresentados casos reais da utilização das válvulas acionadas por atuadores elétricos no ambiente de uma refinaria.

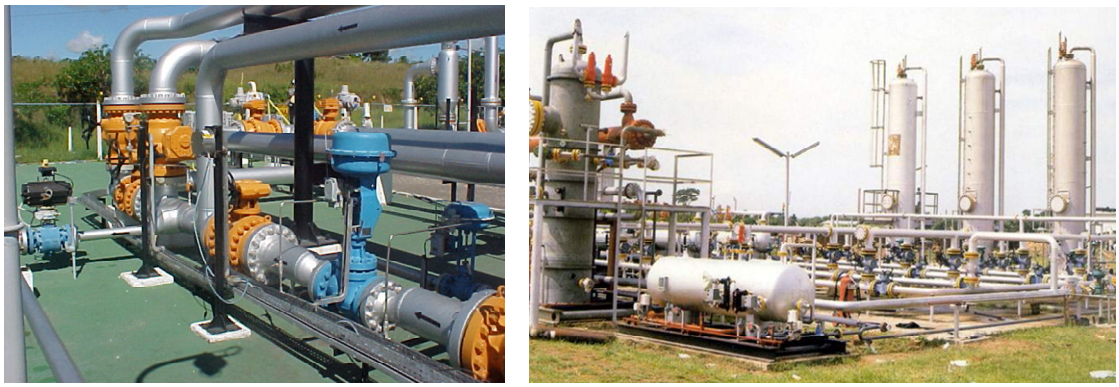
Desse modo, é de interesse da Transpetro implantar um Sistema de Manutenção Inteligente na sua rede dutoviária para que, além de monitorar as informações operacionais e controlar o processo, possa detectar, diagnosticar e prever falhas na operação dos equipamentos.

Assim, a determinação dos motivos que levaram à ocorrência de uma falha não prevista (diagnóstico) e a previsão de falhas nos equipamento ao longo da rede dutoviária tornam-se importantes para a Transpetro. Essas novas tecnologias seriam

integradas ao CNCO e a manutenção proativa adotada como estratégia de manutenção adicionaria mais robustez ao sistema de transporte de derivados de petróleo.


(a) (b)

Figura 2.6: Exemplos dos dutos dentro de uma refinaria. (a) Transporte interprocessos de refino e (b) abastecimento de uma refinaria.



(a) (b)

Figura 2.7: Exemplo de válvulas e atuadores utilizados em refinarias e terminais. (a) válvulas para controle de fluxo em dutos e (b) a grande quantidade de válvulas utilizadas apenas em uma parte da refinaria.

## 2.6 Resumo do Capítulo

Ao longo deste capítulo foi abordada a importância das ações de manutenção dentro dos processos produtivos em uma planta industrial, a fim de reduzir custos e aumentar a eficiência produtiva.

Foi mostrado que a manutenção, quando é bem vista e aplicada pela empresa, pode vir a ser um diferencial em relação aos concorrentes, quanto à redução de custos operacionais de produção. Desse modo, a manutenção é vista como estratégica, pois, se mal aplicada, pode prejudicar diretamente a qualidade dos produtos e a disponibilidade do processo produtivo, afetando os resultados de lucros da empresa.

Neste capítulo, foram apresentadas as definições e características das quatro estratégias mais comuns de manutenção, entre elas a corretiva, preventiva, preditiva e proativa, sendo a última um dos focos do presente trabalho.

As estratégias de manutenção preditiva e proativa passaram a ser implementadas com o avanço tecnológico, uma vez que o sistema produtivo tornou-se mais complexo.

Essas duas estratégias utilizam técnicas de monitoramento da condição dos equipamentos por meio de sistemas computacionais. Em especial, a proativa apresenta-se como a mais interessante, já que possibilita analisar as condições de operação dos equipamentos para determinar se estão ou não funcionando em condições normais e, caso seja detectada alguma anormalidade, pode reconfigurar o equipamento para reduzir os danos. Desse modo, a manutenção preditiva e proativa proporcionam o planejamento da manutenção, atingindo a redução de custos operacionais, como a eliminação de reparos desperdiçados, estoques menores, menos paradas no processo produtivo, previsibilidade da vida útil e planejamento dos trabalhos de manutenção.

Como as modernas plantas industriais são compostas por sensores e sistema de controle e automação sofisticados, baseados em computação, proporcionam reutilizar esses recursos para implantar um sistema de manutenção inteligente nesses equipamentos industriais.

O foco principal desta pesquisa é o desenvolvimento de um Sistema de Manutenção Inteligente (SMI), que é uma técnica computacional para monitoramento e avaliação da condição da planta industrial, sendo uma ferramenta apropriada e eficiente para aumentar a disponibilidade do processo produtivo. Esse sistema visa fazer a detecção de anormalidades, diagnóstico de falhas e predição de comportamentos futuros. Foram apresentadas as definições, características e metodologia para desenvolvimento de um SMI.

Este trabalho está baseado na metodologia adotada pelo padrão OSA-CBM, que é uma arquitetura utilizada para projeto de sistema de manutenção inteligente. Para um melhor entendimento, foram apresentados a arquitetura e os conceitos básicos desse padrão. Foi apresentado um exemplo de SMI, conhecido por *Watchdog Agent*$^{TM}$, também baseado na arquitetura do padrão OSA-CBM.

Por fim, foi apresentado o estudo de caso para o qual o presente trabalho está sendo desenvolvido. Este trabalho pretende implantar SMI como uma ferramenta de tolerância a falhas no sistema de transporte dutoviário da Transpetro, em que, por meio desse sistema, pode-se monitorar a condição de operação dos componentes ao longo dos dutos.

# 3 DETECÇÃO, DIAGNÓSTICO E PREDIÇÃO DE FALHAS

## 3.1 Introdução

No contexto industrial, as máquinas ou equipamentos utilizados nas linhas produtivas também estão suscetíveis as falhas. Estas falhas provocam interrupção da produção, reduzem a qualidade dos produtos e reduz os lucros da empresas. Uma solução adotada pelas empresas é manter uma equipe de manutenção em prontidão para realizarem os reparos para restabelecimento do processo produtivo.

A manutenção é uma tarefa de fundamental importância nos processos industriais e é uma atividade realizada por técnicos humanos. Quando se detecta um evento anormal no processo, é possível diagnosticar a origem da causa e, então, tomar as decisões apropriadas para restauração do processo.

Contudo, a completa confiança em operadores humanos para lidar com eventos anormais e em situações de emergências tem tornado-se gradativamente difícil, devido a vários fatores. Esta dificuldade é em função do amplo escopo das atividades de teste e tolerância a falhas, que apresentam um grande número de aplicações em problemas, como, defeitos nas unidades dos processos, degradação dos processos, parâmetros errados, eventos naturais, etc. Isso, com o passar do tempo, pode se complicar em virtude do tamanho e da complexidade dos processos industriais.

Devido ao rápido desenvolvimento tecnológico e crescimento da complexidade dos processos industriais, passou-se a estudar a aplicação de técnicas de teste e tolerância a falhas, a fim de tornar os equipamentos industriais mais robustos (VENKATASUBRAMANIAN; RENGASWAMY; YIN; ET AL., 2003).

Como pode ser visto, aumentar a confiabilidade dos processos industriais é um grande desafio para os engenheiros. No passado, a comunidade de pesquisa discutia como o controle poderia ser automatizado usando sistemas computacionais para remover os operadores humanos da interação direta com o processo industrial. Isso conduziu para um grande progresso na qualidade e consistência na produção, segurança e eficiência dos processos. O desafio atual está na automação da manutenção de ocorrência dos eventos anormais, que faz uso de sistemas computacionais inteligentes, fornecendo aos operadores humanos a assistência a urgências em muitas áreas.

Neste capítulo, será abordado sobre as técnicas de teste e tolerância a falhas que podem ser adotadas para melhorar os processos industriais e auxiliar nas tarefas de

manutenção. Através do uso de Sistema de Manutenção Inteligente pode-se implementar soluções para detecção, diagnóstico e predição de falhas nos processos industriais.

Definições, nomenclaturas e conceitos básicos sobre testes e tolerância a falhas em geral serão abordados, além é claro, de algumas das técnicas adotadas no presente trabalho para monitorar o comportamento em processos industriais.

O principal objetivo deste capítulo está no estudo da adoção de técnicas de rede neurais para alcançar a tolerância a falhas no processo industrial. Foi definido o algoritmo de Mapas Auto-Organizáveis (SOM) como ferramenta para aquisição de conhecimento sobre o comportamento do sistema e será responsável pelas tarefas de detecção de anormalidades, diagnóstico de falhas e predição de comportamento futuros.

## 3.2 Definições e Conceitos de Testes e Tolerância a Falhas

Neste capítulo, serão apresentadas as definições básicas de testes e tolerância a falhas baseadas nas pesquisas de (AVIZIENIS ET AL., 2004) e demais trabalhos (ISERMANN, 1997), (WEBER, 2003) e (WEBER, 2002). As definições abrangem uma grande área de estudos, desde sistemas baseados em computadores, *hardware*, processos produtivos em geral, operadores humanos e usuários dos sistemas. Contudo, neste capítulo, serão focados os termos do ponto de vista de **sistemas baseados em computador**.

### 3.2.1 Função do sistema, comportamento, estrutura e serviço

Um **sistema,** neste trabalho, é uma entidade que interage com outra entidade, outros sistemas, incluindo *hardwares*, *softwares*, equipamentos, máquinas, processos industriais, homens e o mundo físico. Os outros sistemas são o **ambiente** de um dado sistema. A **fronteira do sistema** é o limite comum entre o sistema e o ambiente.

A **função** de tal sistema está em o que o sistema pretende fazer e é descrito pela **especificação funcional** em termos de funcionalidade e desempenho. O **comportamento** do sistema é o que o sistema faz para executar a função e é descrito por uma sequência de estados. O **estado total** de um dado sistema é o conjunto de estados: computação, comunicação, informações armazenadas, intercomunicação, ação e condição física.

A **estrutura** de um sistema é o que permite gerar o comportamento. Um sistema é composto por um conjunto de componentes interligados e estes interagem entre si, onde cada **componente** é outro sistema.

O **serviço** entregue pelo sistema é como o seu comportamento é percebido pelos demais usuários. Um **usuário** é outro sistema que recebe os serviços.

### 3.2.2 Defeito, falha e erro

Um **serviço correto** é entregue quando o serviço implementa a função do sistema. O **defeito** (*failure*) é um evento que ocorre quando o serviço entregue se desvia do serviço correto. Um serviço falha também por não cumprir com a especificação final ou pela especificação não descrever adequadamente a função do sistema. Uma falha de serviço é a **transição** de um serviço correto para um incorreto. A transição do serviço incorreto para o correto é uma **restauração do serviço**.

Desde que um serviço seja uma sequência de estados externos do sistema, a sua falha diz respeito a, pelo menos, um (ou mais) estado(s) externo(s) do sistema que desviam-se do estado de serviço correto. O desvio é chamado de **erro** (*error*). A causa desse erro é chamada de **falha** (*fault*). As falhas do sistema podem ser internas ou externas.

A definição de **erro** é uma parte de um estado total de falha do sistema que pode conduzir para uma falha do serviço. A falha está **ativa** quando causa um erro, caso contrário está **inativa**.

Quando a especificação funcional de um sistema inclui um conjunto de funções em situação de defeito (*failure*), isso pode deixar o sistema em **modo de degradação**. Nestas condições, mesmo degradado, o sistema pode oferecer um subconjunto reduzido de serviços para o usuário. A especificação pode identificar vários modos, por exemplo, serviços lentos, serviços limitados, serviço de emergência, serviço perigoso, etc. Dize-se, então, que neste caso o sistema sofre uma **falha parcial**.

### 3.2.3 Atributos de dependabilidade

O objetivo de técnicas de tolerância a falhas é alcançar **dependabilidade**. Esse termo é uma tradução literal do termo inglês *dependability*, que indica a qualidade do serviço fornecido por um dado sistema e a confiança depositada no serviço fornecido (WEBER, 2002). Uma definição alternativa é a habilidade de evitar falhas de serviços que são mais frequentes e mais graves do que aceitável.

As principais medidas de dependabilidade são:

- **Disponibilidade** (*availability)*: disponibilizar serviços corretos. Probabilidade de o sistema estar operacional em um determinado instante de tempo. Alternância de períodos de funcionamento e reparo.
- **Confiabilidade** (*reliability)*: continuidade de serviços corretos. Capacidade de atender à especificação, dentro de condições definidas, durante certo período de funcionamento e condicionado a estar operacional no início do período.
- **Segurança** (*safety)*: ausência de consequências catastróficas para o usuário ou ambiente. Probabilidade de o sistema estar operacional e executar sua função corretamente ou descontinuá-la de forma que não provoque dano a outros sistemas ou pessoas que dele dependam.
- **Mantenabilidade** (*maintainability)*: habilidade para aplicar modificações e reparos. Facilidade de realizar a manutenção do sistema, ou seja, a probabilidade que um sistema com defeitos seja restaurado a um estado operacional dentro de um período determinado. A restauração envolve a localização do problema, o reparo físico e a colocação em operação.

### 3.2.4 Técnicas para alcançar dependabilidade

No desenvolvimento de um sistema que precisa ter atributos de dependabilidade, um conjunto de métodos e técnicas deve ser empregado durante o projeto do sistema. Esses métodos e técnicas são divididos em cinco grupos, conforme (AVIZIENIS ET AL., 2004) e (WEBER, 2003):

- **Prevenção de falhas:** recurso para prevenir a ocorrência ou introdução de falhas. Envolve a seleção de metodologias de projeto e de tecnologias adequadas para o processo de engenharia e projeto dos sistemas. Geralmente, é realizada durante o desenvolvimento do sistema, aplicando-se metodologias que visam evitar as falhas. As melhorias no processo de desenvolvimento objetivam reduzir a incidência de falhas introduzidas no sistema, devido a erros de projeto.
- **Tolerância a falhas**: recurso que evita falha de serviços durante a presença de uma falha no sistema. Ele visa evitar a ocorrência das falhas pela *detecção* de erros e *recuperação* do sistema. Com frequência, o tratamento de falhas é seguido por uma operação de *manutenção*, cujo foco é a remoção das falhas em tratamento, isto é, o fator que distingue tolerância a falhas da manutenção é que esta requer a participação de um agente externo.
- **Remoção de falhas**: recurso para reduzir o número e a gravidade das falhas. Pode ser aplicado durante o desenvolvimento do sistema, visando encontrar suas falhas antes de sua inserção no mercado. Além disso, pode ser empregado durante o uso do sistema, em que é realizado por meio da manutenção corretiva ou preventiva. A *manutenção corretiva* tem por objetivo remover as falhas que produziram um ou mais erros reportados, enquanto a *manutenção preventiva* propõe descobrir e remover falhas antes delas causarem erros durante o funcionamento normal do sistema. Essas formas de manutenção aplicadas a sistemas não-tolerantes a falhas os tornam como um sistema tolerante a falhas, em que sua manutenção pode ser realizada *on-line* (sem interrupção do fornecimento dos serviços) ou *off-line* (durante a parada dos serviços).
- **Diagnóstico de falhas**: recurso para classificar ou identificar o tipo, tamanho, a localização e o tempo da ocorrência da falha. O diagnóstico de falhas visa principalmente realizar uma classificação da falha detectada, servindo como auxílio para outras técnicas. Quando uma falha é detectada, o diagnóstico é usado para avaliá-la e determinar a sua causa.
- **Predição de falhas**: recurso para estimar o número, a incidência futura e provável consequência das falhas supostamente presentes. É conduzida pela realização de uma *avaliação do comportamento* do sistema com respeito à ocorrência de falhas. A avaliação pode ser *qualitativa* e seu propósito é identificar, classificar e ordenar os modos de defeitos ou combinação de eventos (defeito em componentes ou condições do ambiente) que deveriam levar o sistema a falhas. Também pode ser *quantitativa*, cuja finalidade é a avaliação do nível de probabilidade que alguns atributos são satisfeitos, em que esses atributos são vistos como medidas.

Este trabalho focaliza a aplicação dessas técnicas para alcançar a dependabilidade nos sistemas aplicados em processos industriais, não sendo aqui abordadas, no entanto, técnicas para prevenção de falhas.

### 3.2.5 Ameaças à dependabilidade

De acordo com definições de (AVIZIENIS ET AL., 2004), será abordado o ciclo de vida e algumas ameaças prejudiciais que afetam o nível de dependabilidade do sistema. Também serão aprofundados conceitos sobre defeito, falha e erro

*3.2.5.1 Ciclo de vida de um sistema*

O ciclo de vida de um sistema é separado em duas fases:

- *Fase de desenvolvimento* inclui atividades, como a concepção do sistema e avaliação do usuário até que o sistema seja aprovado em todos os casos de teste e esteja pronto para disponibilizar os serviços. Durante o desenvolvimento, o sistema interage com o ambiente de desenvolvimento e falhas podem ser introduzidas desde as primeiras atividades.
- *Fase de uso* de um sistema começa quando ele é aceito pelo usuário e inicia-se a disponibilização dos serviços. O uso consiste de períodos alternados de **entrega de serviços corretos**, **parada de serviço** e **suspensão (desligamento) de serviço**. A parada de serviço é causada por um defeito. Esse é o período em que um serviço incorreto é entregue como saída. A suspensão de serviço é uma parada intencional do serviço por uma entidade autorizada. A **manutenção** são ações aplicadas durante os três períodos da fase de uso.

Durante a fase de uso, o sistema interage com o *ambiente de uso* e pode ser afetado por falhas originadas nesse ambiente. O **ambiente de uso** consiste nos seguintes elementos:

1. *Mundo físico*: os fenômenos naturais.
2. *Administradores*: entidades (humanos ou outros sistemas) que têm autoridade para gerenciar, modificar, reparar, restaurar e utilizar o sistema; alguns podem ter atitudes para manter o sistema (manutenção) ou atitudes maliciosas.
3. *Usuários*: entidades (humanos ou outros sistemas) que recebem os serviços fornecidos pelo sistema.
4. *Provedor*: entidades (humanos ou outros sistemas) que entregam serviços para o sistema.
5. *Infraestrutura*: entidades que disponibilizam serviços especializados para o sistema, como comunicação, localização, fonte de alimentação, refrigeração, etc.
6. *Intrusos*: entidades maliciosas (humanos ou outros sistemas) que desrespeitam um agente autorizado para ter acesso, alterar os serviços ou paralisá-los, modificar os serviços ou a funcionalidade, ou o desempenho.

O termo **manutenção** inclui os reparos ou a restauração e todas as modificações que o sistema sofre durante a fase de uso no ciclo de vida. Na Figura 3.1, são apresentadas as formas de manutenção de acordo com o ponto de vista de tolerância a falhas proposto por (AVIZIENIS ET AL., 2004).

A manutenção é divida em duas fases: a de *reparos* que é referente à restauração da condição de falha dos serviços do sistema para uma situação que forneça os serviços corretos, que basicamente visa remover as falhas reportadas ou descobertas no sistema. A outra é a de *modificações* que envolve a etapa de projeto do sistema ou a inserção de novas funcionalidades (atualização) no fornecimento dos serviços, por exemplo, devido a mudanças repentinas no ambiente de uso ou por falta de serviços especificados no projeto.

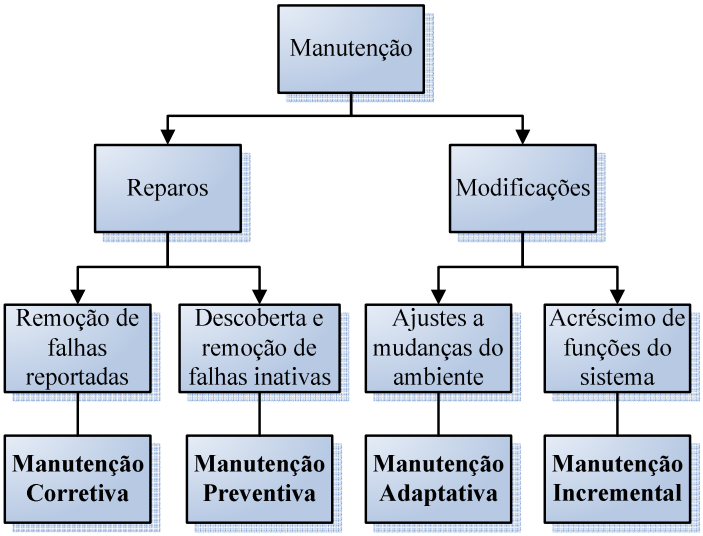



Figura 3.1: Formas de manutenção segundo (AVIZIENIS ET AL., 2004).

O reparo é um conceito aplicado à tolerância a falhas, que envolve a participação de um agente externo, ou seja, um técnico de uma equipe de manutenção, equipamento para testes *on-line* ou a avaliação por *softwares*. Aplica-se técnicas de *remoção de falhas* (durante a fase de uso) ou *predição de falhas* para apontar situações em que é necessário realizar o reparo.

De fato, o reparo pode ser considerado como uma atividade da tolerância a falhas dentro de um sistema mais amplo, que inclui o sistema em reparo, as pessoas envolvidas e os outros sistemas que interagem entre si, que visam realizar os consertos necessários.

#### *3.2.5.2 Defeitos*

O **defeito de serviço** é definido como um evento que ocorre quando a entrega de um serviço é desviado de um serviço correto. Os defeitos nos serviços se caracterizam de acordo com quatro pontos de vista:

- *Domínio*: os defeitos podem ser de *conteúdo*, em que o conteúdo das informações fornecidas pelo sistema desvia-se das funções implementadas, e de *temporização*, em que o tempo de chegada ou duração da informação entregue pelo sistema desvia-se das funções implementadas.
- *Detectabilidade*: visa sinalizar ao usuário os defeitos ocorridos no sistema. A sinalização faz uso de mecanismos de detecção que verificam o funcionamento correto dos serviços entregues pelo sistema. Quando perdas são detectadas e sinais de alerta são emitidos, ocorre a sinalização do defeito, caso contrário não é sinalizado. Quando a ocorrência do defeito provocar a redução dos serviços, o sistema está em modo *degradado* para o usuário. Esse modo pode reduzir o funcionamento, acionar uma emergência ou paralisar o sistema.
- *Consistência*: os defeitos *consistentes* são os serviços incorretos que são percebidos identicamente por todos os usuários do sistema, já os defeitos *inconsistentes* são percebidos de forma diferenciada por alguns ou todos os usuários do sistema.

- *Consequência*: os defeitos são definidos de acordo com uma relação entre benefício durante a ausência de falhas e a consequência dos serviços entregues pelo sistema. Nos *defeitos sem importância,* as consequências dos danos são similares aos custos e benefícios fornecidos pelo serviço correto e nos *defeitos catastróficos,* as conseqüências dos danos são ordens de magnitude ou imensuravelmente superiores aos benefícios fornecidos pelo serviço correto.

#### *3.2.5.3 Falhas*

As falhas que afetam o sistema durante seu ciclo de vida pertencem a quatro grupos básicos de acordo com a origem da causa:

- *Falhas de desenvolvimento*: incluem todas as classes de falhas que ocorrem durante o desenvolvimento do sistema, como erro na especificação, metodologia de projeto inadequada, etc.
- *Falhas de interação*: incluem todas as falhas externas ao sistema. São provocadas por elementos do ambiente que interagem com o sistema durante o seu uso.
- *Falhas naturais*: são falhas físicas que afetam a parte estrutural do sistema (ou *hardware*) e são causadas por fenômenos naturais sem a participação humana. Durante a operação do sistema, as falhas naturais podem ser *internas*, devido a processos naturais que causam deteriorações físicas, ou *externas*, em razão do processo natural ser afetado por sistemas vizinhos.
- *Falhas provocadas por humanos*: são falhas que incluem a ausência de atos que deveriam ser executados, ou seja, omissão. Também podem ser classificados de acordo com o objetivo dos humanos que interagem com o sistema:
    - *Falhas maliciosas*: são introduzidas no sistema com o objetivo de causar dano ao sistema, alterando seu funcionamento. Podem também ser introduzidas por meio de reparos realizados. Tem como consequência a interrupção do fornecimento dos serviços (produzir prejuízos) ou roubo de informações privilegiadas.
    - *Falhas não-maliciosas*: são introduzidas sem um objetivo danoso, mas que podem ser originadas de tomadas de decisão erradas, corte de custos, peças inadequadas em reparos ou de baixa qualidade.

#### *3.2.5.4 Erros*

Um erro é uma parte de um estado total do sistema que pode levar a um defeito. Este ocorre quando um erro causa um desvio no serviço entregue do serviço correto. A causa de um erro pode ser chamada de **falha**.

Um erro é **detectado** se a presença dele é percebida por algum outro sistema ou é indicada por um *alarme de erro*. Erros que estão presentes, mas não são detectados, são erros **latentes**.

Um sistema consiste em um conjunto de componentes que interajem entre si, cujo estado total é o conjunto de estados dos componentes. A definição implica que originalmente uma falha deve causar um erro no estado de um (ou mais) componente(s),

mas o defeito de serviços não ocorrerá enquanto o estado externo do componente não fizer parte do estado externo do sistema. Não importa quando o erro se torna parte do estado externo do componente, uma vez que o defeito sempre ocorrerá, mas, sim, quando o erro permanece interno em todo o sistema.

#### *3.2.5.5 Relação entre falhas, erros e defeitos*

O mecanismo de criação e de manifestação de falhas, erros e defeitos, em geral segue uma ordem definida. A Figura 3.2 representa a ordem de propagação de erros, mostrando a relação entre falha, erro e defeito conforme a seguir:

1. Uma falha é *ativada* e um erro é produzido. A *ativação* da falha é a aplicação de uma entrada a um componente que causa a ativação de uma falha dormente.
2. A *propagação* de um erro dentro de um dado componente é causada pelo processo do sistema. Um erro é sucessivamente transformado em outros erros.
3. O defeito de um serviço ocorre quando um erro é propagado para a interface do serviço e *causa* para o serviço, disponibilizado pelo sistema, o desvio do serviço correto.

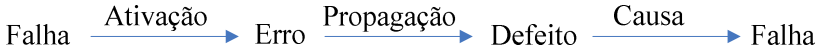



Figura 3.2: Mecanismo de propagação do erro (AVIZIENIS ET AL., 2004).

Nas próximas seções, será estendido o uso dos conceitos destas técnicas de tolerância a falhas para serem aplicados neste trabalho. Entre elas, a detecção, diagnóstico e predição ou monitoramento de falhas em processos industriais.

## 3.3 Detecção e Diagnóstico de Falhas

A primeira técnica para alcançar dependabilidade abordada neste estudo é a tarefa para Detecção e Diagnóstico de Falhas (DDF), apresentada por (KATIPAMULA; BRAMBLEY, 2005). Esta é uma área que aborda a automatização do processo a fim de detectar falhas e diagnosticar suas causas. É uma técnica baseada nos conceitos de *testes e tolerância a falhas* que visa evitar a ocorrência ou continuidade das falhas, pela *detecção* de erros e *recuperação* do sistema por meio de reparos nas partes identificadas que causaram a falha (VENKATASUBRAMANIAN; RENGASWAMY; YIN; ET AL., 2003). Essa técnica é fundamental para a Manutenção Inteligente (Seção 2.3) em sistemas industriais.

### 3.3.1 Processo para DDF em uma aplicação genérica

Segundo (KATIPAMULA; BRAMBLEY, 2005), o objetivo de um Sistema Automático de DDF é detectar antecipadamente a ocorrência das falhas e diagnosticar suas causas, permitindo a remoção das falhas antes que causem danos adicionais ao sistema ou perda do fornecimento de serviços.

Esse sistema é realizado por um monitoramento contínuo das condições de operação do sistema aplicado. Além disso, visa detectar e diagnosticar as condições anormais de funcionamento e associar a elas as falhas encontradas, avaliando a importância de cada uma e decidindo como combatê-las.

Um processo de operação e manutenção em um sistema ou equipamento que contém um Sistema Automático de DDF pode conter quatro funções, conforme a Figura 3.3:

1. Monitorar o sistema ou equipamento até detectar alguma condição de anormalidade (problemas). Essa etapa é referente à *Detecção da Falha*.
2. Quando é detectada uma condição anormal, o *Diagnóstico da Falha* é acionado para avaliar a falha e determinar sua causa. As duas primeiras etapas constituem o *Processo de DDF*.
3. Após o diagnóstico, é aplicada uma *Avaliação da Falha*, que objetiva estimar o tamanho e a importância do seu impacto no desempenho do sistema (por exemplo, em termos de consumo de energia, segurança de operadores humanos, danos que pode causar, disponibilidade, custos, impacto ao ambiente, etc.).
4. Com base na avaliação anterior, a etapa de *Tomada de Decisão* é acionada, com a finalidade de decidir como responder à falha (por exemplo, uma ação de reparos pela manutenção). Essa etapa é acompanhada com o auxílio de um especialista humano para tomar as decisões convenientes à situação ocorrida.

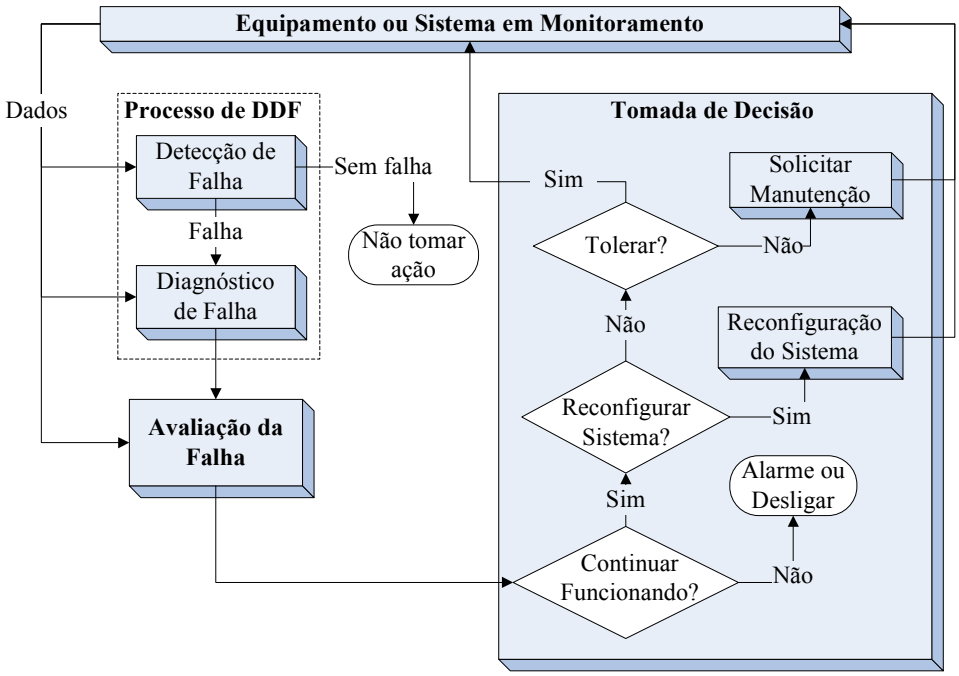



Figura 3.3: Estrutura geral de um Sistema Automático de DDF para uma aplicação genérica (KATIPAMULA; BRAMBLEY, 2005).

Na etapa de tomada de decisão, o sistema pode se reconfigurar de modo a suportar a falha ocorrida e continuar em funcionamento, mesmo em condição degradada, mas continuará emitindo um alarme de aviso do problema. Ou seja, a decisão de reconfiguração do sistema visa reduzir o esforço e talvez amenizar a causa da falha e estender por um tempo maior a continuidade do sistema em funcionamento correto.

Todas as decisões que venham a ser tomadas dependem muito do ponto de vista do especialista sobre as ameaças a dependabilidade do sistema, vistas na Seção 3.2.5. Dependendo da análise de cada caso, a detecção de falhas pode ser relativamente mais

fácil do que o diagnóstico, por causa das falhas ou da avaliação da origem dos impactos das falhas.

Muitas abordagens para implementar um Sistema de DDF foram propostas (MULLER ET AL., 2008) (JARDINE ET AL., 2006) (VENKATASUBRAMANIAN; RENGASWAMY; YIN; ET AL., 2003), apresentado estruturas e formas diferentes para a sequência de etapas de detecção e diagnóstico, que não é definida como fixa, mas pode ser variável, dependendo do objetivo. Em alguns casos, o sistema de detecção funciona continuamente, enquanto em outro o diagnóstico é ativado depois da detecção da falha. Também, pode ocorrer de ambas executarem em paralelo, ou em outros casos, cada etapa é executada uma vez.

### 3.3.2 Fundamentos de um Sistema de DDF

De acordo com (KATIPAMULA; BRAMBLEY, 2005) diferentes abordagens podem ser utilizadas para implementar a técnica de detectar e diagnosticar falhas. A principal diferença entre elas está nas abordagens de conhecimento ou informações adquiridas, que serão utilizadas para formular o diagnóstico ou classificação.

Em geral, as técnicas para diagnósticos são baseadas em modelos que podem ser de conhecimento *a priori* (p. ex., modelos de comportamento) ou completamente empírico com base nos dados (p. ex., modelo caixa preta). Ambas as abordagens utilizam modelos e dados, mas para formular a base para o diagnóstico é necessário abordagens diferentes. Também pode ser por meio de dados de histórico do processo.

Na abordagem baseada no conhecimento *a priori*, usa-se esse conhecimento (relações físicas) para especificar um modelo que serve como base para identificar e avaliar diferenças entre o estado atual de operação, o estado de operação esperado e os valores das características obtidos do modelo.

A abordagem puramente empírica (baseada nos dados) não utiliza conhecimento *a priori* do processo, em vez disso, cria-se um modelo apenas com base nas medições do comportamento físico dos dados e do próprio processo.

Métodos baseados em modelo podem ser *quantitativos* ou *qualitativos*. Modelos quantitativos são conjuntos de relações matemáticas baseadas nos fundamentos físicos de construção dos processos. Modelos qualitativos são modelos formados pelas relações derivadas dos fundamentos físicos e incluem também o uso de sistemas com base em regras.

Em terceiro lugar, têm-se os modelos baseados inteiramente no processamento de dados do histórico, em que assume-se uma grande quantidade de dados históricos está disponível. Esses modelos incluem métodos que são derivados puramente dos dados coletados ou a aplicação de técnicas para extrair características (processamento de sinais). Estas técnicas, por sua vez, incluem a aplicação de métodos derivados da estatística, de redes neurais e classificação de padrões.

Neste trabalho, foi adotada a aplicação da técnica com base no processamento de dados do histórico.

### 3.3.3 Processamento de Dados do Histórico

O método tem como base o processamento de dados históricos do processo, não necessitando conhecimento extra sobre o sistema.

Do ponto de vista industrial, uma grande quantidade de aplicações de diagnóstico de falhas é baseada em abordagens de processamento do histórico. Isso é devido ao fato de o processamento de histórico ser uma abordagem de fácil implementação e que requer pequeno esforço de modelagem e conhecimento *a priori*.

Existem algumas técnicas de transformação que são aplicadas a esses dados e preparam como um "conhecimento *a priori*" para o sistema de diagnóstico. Essas técnicas são conhecidas como extração de características, e podem ser classificadas como não-estatísticas ou estatísticas. As redes neurais (HAYKIN, 2001) se destacam como uma importante classe de ferramentas classificadas como não-estatísticas, é adotada no presente trabalho e será abordada na Seção 3.5..

### 3.3.4 Características desejáveis em um Sistema de DDF

Como existem diversas abordagens para detecção e diagnóstico na literatura, este trabalho foi baseado nas pesquisas de (VENKATASUBRAMANIAN; RENGASWAMY; KAVURI; ET AL., 2003) e (VENKATASUBRAMANIAN; RENGASWAMY; YIN; ET AL., 2003), na qual é apresentado um conjunto de características e requisitos desejáveis em sistemas de detecção e diagnóstico.

Essas características são úteis para avaliar os vários métodos de implementação de um Sistema de DDF, como os dados disponíveis, confiabilidade da solução, generalização, eficiência da computação, etc. Assim, não importando quando uma anormalidade ocorre no sistema, o sistema, para um caso geral, deve ser capaz de sugerir um conjunto de hipóteses ou falhas que expliquem a anormalidade.

A seguir são apresentadas algumas características para projeto um Sistema de DDF:

- **Detecção e diagnóstico rápidos:** o sistema deve responder rapidamente ao detectar e diagnosticar falhas do sistema. Um sistema que é projetado para detectar falhas rapidamente (particularmente mudanças bruscas) poderá ser sensível a influências externas, como ruídos, o que pode levar a falsos alarmes.
- **Isolabilidade:** é a habilidade do sistema em distinguir entre diferentes tipos de falhas. Sob condições ideais, livre de ruído e incertezas do modelo, o sistema de diagnóstico deve gerar uma saída que classifique corretamente a falha ocorrida.
- **Robustez:** deve ser robusto o suficiente em condições com ruído e incertezas.
- **Identificação de novos defeitos:** deve ser capaz de identificar, dado as condições atuais do sistema, se está funcionando em modo normal ou anormal e, se anormal, verificar se a causa é um defeito conhecido ou não. Mesmo sobre condições não previstas no projeto, o sistema de diagnóstico deve reconhecer a ocorrência de uma nova falha e não fazer uma classificação errônea.
- **Estimar erro de classificação:** um requisito importante e prático para o sistema é construir um valor de confiança para o usuário. Idealmente, o sistema de diagnóstico pode estimar um erro de classificação, como uma medida para estimar níveis de confiança para as decisões decorrentes do diagnóstico.

- **Adaptabilidade:** processos, em geral, passam por aperfeiçoamentos, sofrem mudanças das entradas externas ou mudanças na estrutura e assim por diante. As condições de operação do processo sofrem alterações não somente em função das perturbações, mas também das mudanças no ambiente, tais como, na quantidade de produção, na qualidade dos materiais, na vazão de uma tubulação, etc. Assim, o sistema de diagnóstico deve ser adaptável a tais mudanças, sendo possível de customizar gradualmente o escopo para adicionar os novos casos que surgem.
- **Facilidade de explicação:** além da habilidade de identificar a origem das falhas, o sistema deve fornecer explicações como a falha se originou e se propagou até a situação atual. Isso é um fator importante para um Sistema *On-line* de Suporte à Decisão. Por fim, o sistema deve justificar suas recomendações ao operador conforme for a avaliação.
- **Requisito computacional e armazenamento:** geralmente, soluções de tempo real exigem requisitos de armazenamento, processamento e consumo de potência. O ideal é alcançar um sistema de diagnóstico que encontre um ponto de equilíbrio entre eles.

## 3.4 Detecção, Diagnóstico e Predição de Falhas

A técnica de Detecção, Diagnóstico e Predição de Falhas (DDPF), apresenta muitas semelhanças com a Detecção e Diagnóstico de Falhas (DDF) da seção anterior. Na DDPF, aplicam-se técnicas automáticas para detectar e estimar a degradação no desempenho de sistemas físicos, projetar uma tendência no comportamento, antecipar a ocorrência de falhas futuras e calcular o tempo restante de vida do sistema, mantendo–o em um estado operacional aceitável.

É com base nos conceitos da *tolerância a falhas* que se evita a ocorrência de falhas antes de sua manifestação pela *detecção da piora do desempenho* e *recuperação* do sistema por meio de reparos nas partes que sofreram degradação. Essa técnica é fundamental para a Manutenção Inteligente (Capítulo 2) em sistemas industriais.

### 3.4.1 Processo de DDPF para uma aplicação genérica

O objetivo de um Sistema Automático de DDPF é detectar, com antecedência, um desvio no comportamento normal do sistema, evitando a ocorrência das falhas dentro de um tempo hábil. Isso permite a remoção das falhas antes que causem danos adicionais ou prejudiquem o fornecimento dos serviços pelo sistema.

A predição é importante para a avaliação do impacto nas decisões de operação e manutenção em um sistema, pois permite a transição da manutenção corretiva ou preventiva para a manutenção proativa (manutenção inteligente). A manutenção proativa tem como base a antecipação das condições futuras do sistema, tempo restante antes da falha (ou tempo antes de alcançar um nível inaceitável no desempenho), a taxa de degradação e a origem da falha.

Em um Sistema DDPF, é realizado um monitoramento contínuo das condições de operação do processo industrial em análise. Ademais, o monitoramento visa detectar desvios de comportamento em relação às condições normais de funcionamento. Quando ocorrerem os desvios, é preciso identificar a causa e estimar um tempo até a manifestação de um defeito relacionado à degradação que vem ocorrendo. Pode-se

também diagnosticar a causa da falha encontrada, avaliando sua importância e decidir como combatê-la.

O processo de monitoramento em um sistema ou equipamento que contém um Sistema Automático de DDPF é apresentado na Figura 3.4:

1. Monitorar o sistema ou equipamento até detectar alguma condição de anormalidade (problemas). Essa etapa é referente à *Detecção de Desvio de Comportamento*.
2. Quando é detectada uma condição anormal, o sistema deve avaliar se é a manifestação de uma falha ou uma tendência de degradação no desempenho do equipamento. Caso seja uma falha, aplica-se o *Processo de DDF* visto na Seção 3.3. Caso seja detectado o início de uma degradação, aplica-se a *Predição de Comportamento*.
3. A *Predição de Comportamento* visa avaliar o comportamento do sistema utilizando como base um *Banco de Conhecimento ou Histórico*. Utiliza-se como premissa que comportamentos passados podem se manifestar novamente em um tempo futuro.
4. Após, é aplicada a *Avaliação do Comportamento*, que objetiva calcular uma estimativa, identificar, classificar uma tendência que leva o sistema até a ocorrência da falha e suas possíveis consequências. Por exemplo, em termos de probabilidade de defeito, detecção e classificação de uma tendência, tempo até a manifestação da falha, origem do defeito, etc.
5. Com base na avaliação anterior, a etapa de *Tomada de Decisão* é acionada e pode utilizar também os resultados do Sistema DDF. Ela propõe decidir como responder à percepção da degradação ou ocorrência da falha (p. ex., solicitar reparos por meio de uma manutenção, devido à degradação). Essa etapa pode ser acompanhada de um especialista humano, para tomar as decisões convenientes à situação ocorrida, mas não é obrigatório.

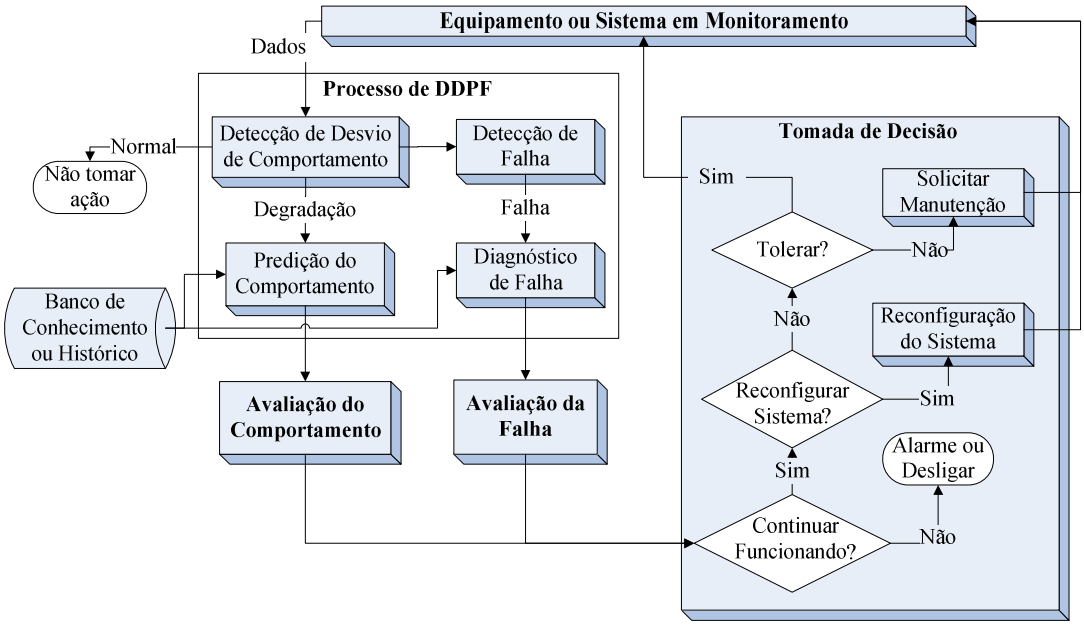



Figura 3.4: Exemplo de um Sistema de Detecção, Diagnóstico e Predição de Falhas (DDPF) para uma aplicação genérica.

A etapa de tomada de decisão, para o caso de um Sistema DDPF, pode apresentar o mesmo comportamento do Sistema DDF, como também pode funcionar de modo autônomo. Por exemplo, ao detectar a ocorrência de uma degradação, o sistema pode tomar uma decisão sem consentimento de um especialista, tolerando uma degradação por esta estar prematura. Também pode aplicar a autorreconfiguração no equipamento, limitando um parâmetro de construção para aumentar o tempo restante de vida útil do equipamento, mesmo operando em estado degradado.

Outro ponto importante da tomada de decisão de um Sistema DDPF, é durante a avaliação do comportamento, em que a predição pode disponibilizar informações mais precisas da condição de funcionamento do sistema. Também auxilia a equipe de especialistas em manutenção a tomar decisões corretas, baseadas em variáveis realistas do equipamento, como o índice de desempenho, a taxa de degradação de partes ou o tempo restante até falhar.

Na predição de falhas, é importante a integração junto com a detecção de falhas, pois elas se completam, formando um sistema mais robusto a falhas. Na Tabela 3.1, é apresentado um comparativo entre as duas abordagens, conforme apresentado por (AGARWAL ET AL., 2007):

Tabela 3.1: Comparação entre predição e detecção de falhas.

| **Predição de Falhas** | **Detecção de Falhas** |
|---|---|
| Aplica-se antes de surgir falhas | Aplicável quando as falhas se manifestam |
| Detecta a degradação nos dados coletados antes de predizer a falha | Devido aos erros e manifestação da falha, pode causar problema de integridade nos dados coletados |
| Custo de manutenção mais barato | Custo de manutenção mais caro |
| Possibilidade de tomada de decisão autônoma | Tomada de decisão limitada pela ação de especialista |
| Uma predição incorreta pode ser um grande problema | Cobertura de falhas insuficientes pode ser um problema |
| Não é possível predizer todas as falhas | Aplicado a falhas mais comuns |

As etapas de detecção, diagnóstico e predição seguem um certo padrão de ativação. É importante manter funcionando continuamente em tempo real a etapa de *Detecção de Desvio de Comportamento*, pois, a partir desta, são acionadas as próximas etapas.

Caso seja detectado um princípio de degradação, a etapa de predição é executada ou se uma falha ou uma degradação muito elevada for detectada, o Sistema de DDF é acionado. Essa sequência não é definida como fixa, mas pode ser variável dependendo do objetivo do sistema.

Este trabalho foca no projeto de um Sistema DDPF utilizando técnicas de redes neurais. Segundo (KOHONEN ET AL., 1996) a rede neural SOM tem algumas características importantes que proporcionam o monitoramento da ocorrência de falhas em processos dinâmicos. Deste modo, adotou-se aplicá-lo como técnica para testes *on-line* e será apresentado na seção a seguir.

## 3.5 SOM – Mapas Auto-Organizáveis de Kohonen

Uma importante característica das redes neurais é a habilidade de *aprender* sobre um determinado ambiente a partir de observações de amostras. A aprendizagem é o meio pela qual a rede neural adquire conhecimento do ambiente.

O processo de aprendizagem de uma rede neural é um processo iterativo de ajustes aplicados aos seus pesos sinápticos, tornando-se apta a exercer sua ação no ambiente (classificação, inferência, etc.). Existe uma grande variedade de algoritmos são classificados de acordo com o *paradigma de aprendizagem*, isto é, a maneira como a rede se relaciona com o ambiente. De acordo com o tipo do ambiente para o qual a rede é aplicada, os métodos podem ser: *aprendizagem supervisionada* ou *aprendizagem não-supervisionada* (HAYKIN, 2001).

O modelo de aprendizagem supervisionada pode ser visto na Figura 3.5. Nas redes neurais baseadas nesse modelo, o treinamento é feito a partir de amostras de entrada e saída do sistema. Um conjunto de amostras são rotuladas (uma entrada com a sua respectiva saída desejada). O conhecimento que se tem inicialmente sobre o ambiente é chamado de *professor*. A diferença entre a resposta desejada (do professor) e a resposta do sistema (a rede neural) alimenta novamente o sistema para aprendizado, e ajustes para corrigir os pesos sinápticos da rede são realizados. Esses ajustes são feitos até que o conhecimento do professor seja transferido para a rede. Com o passar do tempo, o professor pode ser dispensado.

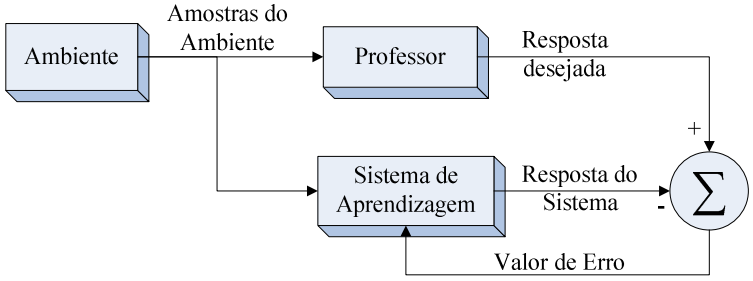



Figura 3.5: Modelo de aprendizado supervisionado

O modelo de aprendizagem não-supervisionada pode ser visto na Figura 3.6. Nas redes neurais baseadas nesse modelo, o treinamento é feito diretamente a partir das características dos dados de entrada, sem necessitar de um professor externo para ensinar. A rede aprende diretamente do ambiente, criando automaticamente novas classes. É geralmente utilizada em tarefas de classificação e detecção de padrões, em que é possível separar as amostras em grupos, levando em consideração apenas as proximidades entre seus atributos. Neste trabalho, é adotado esse modelo de aprendizagem, que também pode ser chamado de *aprendizagem auto-organizada*.

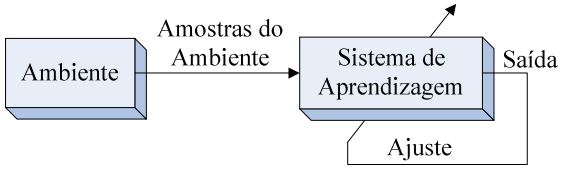



Figura 3.6: Modelo de aprendizado não-supervisionado.

O objetivo de um algoritmo para aprendizagem não-supervisionada é descobrir padrões de características nos dados de entrada e fazer essa descoberta sem informações

prévias. O algoritmo dispõe de um conjunto de regras confinadas à vizinhança imediata do neurônio que o capacita a aprender e calcular um mapeamento de entrada e saída, com base principalmente na similaridade dos dados.

A arquitetura de um sistema auto-organizável consiste de uma *camada de entrada (fonte de dados)* e uma *camada de saída (de representação)* com conexões alimentadas da entrada até a saída e conexões laterais entre neurônios na camada de saída. O processo de aprendizagem consiste na modificação repetitiva dos pesos sinápticos em resposta a padrões de entrada e de acordo com regras predeterminadas, até se desenvolver a configuração final do sistema.

Neste trabalho, é adotada a classe de redes neurais conhecidas como Mapas Auto-Organizáveis (SOM – *Self-Organizing Maps*) (KOHONEN, 1990) (KOHONEN, 2001), que é um modelo de arquitetura para um sistema auto-organizável proposto no final da década de 1980. Nesse modelo, a ideia consiste em que os neurônios da rede devem competir entre si. Como resultado dessa competição, tem-se o *neurônio vencedor*.

Em um *Mapa Auto-Organizável*, os neurônios são alocados em uma forma bidimensional ou unidimensional. Os neurônios são ajustados por estímulos externos vindos da entrada. Com o tempo, as localizações espaciais dos neurônios se tornam ordenadas entre si, formando um sistema de coordenadas para as diferentes *características* extraídas da entrada.

Um Mapa Auto-Organizável é caracterizado pela formação de um *mapa topográfico* dos estímulos de entrada, em que as localizações espaciais (coordenadas) dos neurônios, na grade, são indicativas das características estatísticas intrínsecas contidas nos padrões de entrada.

### 3.5.1 Fundamentos

A fundamentação teórica para o Mapa Auto-Organizável (SOM) está baseado em (HAYKIN, 2001). O principal objetivo do SOM é a transformação do padrão de um sinal de dimensão arbitrária em um mapa discreto uni ou bidimensional. Essa transformação é realizada de uma maneira adaptativa em que se mantém a mesma ordem topológica dos dados de entrada.

Na Figura 3.7, é apresentada a arquitetura clássica a partir da qual o SOM é modelado (KOHONEN, 2001). O formato básico da rede é composto por *neurônios* arranjados em linhas (*Dimensão L*) e colunas (*Dimensão C*) dispostos em forma bi ou unidimensional, chamada de *Camada de Saída*, formando o *Mapa de Características*. Cada neurônio é conectado com todos os nós da *Camada de Entrada*. Na *Camada de Entrada*, são apresentados os padrões ou estímulos de entrada para a rede. Cada neurônio contém um *Peso Sináptico* que representa as características individuais de cada um.

O algoritmo do mapa auto-organizável inicializa os pesos sinápticos na rede, atribuindo-lhes valores com números aleatórios, assegurando, dessa forma, que nenhuma organização prévia seja imposta ao mapa de características. Uma vez inicializada a rede, são aplicadas três fases durante o *treinamento*:

- *Competição*: cada padrão de entrada é apresentado para todos os neurônios da rede que calculam seus respectivos valores com base em uma função de similaridade, que atribui crédito para cada neurônio. Essa função fornece a base para a competição entre os neurônios. O neurônio que apresentar maior

valor da função de similaridade (mais próximo do padrão de entrada) é declarado como “vencedor” da competição.

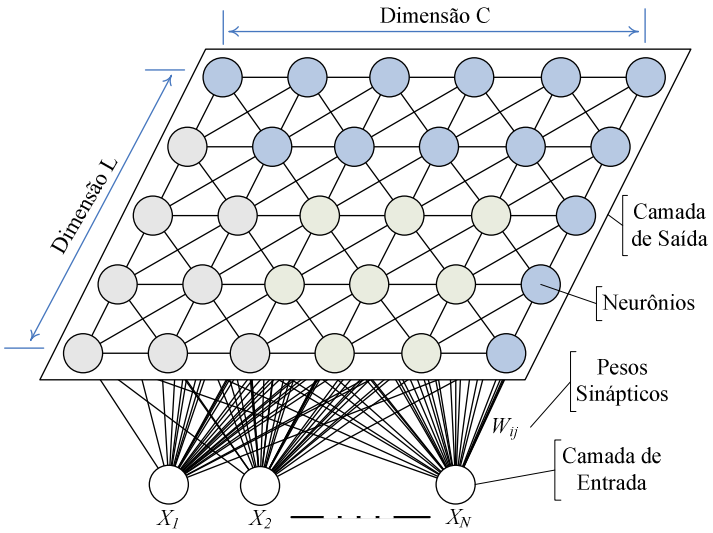



Figura 3.7: Arquitetura da rede neural SOM proposta por (KOHONEN, 2001).

- *Cooperação*: o neurônio “vencedor” determina uma localização espacial, que forma uma vizinhança topológica para excitação dos neurônios mais próximos. Isso fornece a base de cooperação entre os neurônios contidos na vizinhança.
- *Adaptação Sináptica*: os neurônios excitados recalculam seus pesos sinápticos para melhorar seus valores individuais da função de similaridade em relação ao padrão de entrada. Os ajustes são aplicados de tal modo que, quando uma nova apresentação de um padrão de entrada for similar à resposta do neurônio vencedor, este recebe ajustes finos a fim de melhorar seus resultados.

#### *3.5.1.1 Processo de competição*

Considere que *N* representa a dimensão do espaço de entrada de dados. Um padrão ou vetor de entrada **x** selecionado aleatoriamente em um conjunto do espaço de entrada *X*, é representado por

$$\mathbf{x} = \left[x_1, x_2, \cdots, x_N\right]^T . \tag{1}$$

O vetor de peso sináptico para cada neurônio da rede tem a mesma dimensão do espaço de entrada. Considere que o vetor peso sináptico **w** do neurônio *j* seja representado por

$$\mathbf{w}_j = \left[w_{j1}, w_{j2}, \cdots, w_{jN}\right]^T , \tag{2}$$

onde *j=1,2,...,l*, e *l* é o número total de neurônios na rede.

Para encontrar o melhor casamento do vetor de entrada **x** com os vetores pesos sinápticos $\mathbf{w}_j$, é preciso comparar os produtos internos $\mathbf{w}_j^T\mathbf{x}$ para *j=1,2,...,l* e selecionar o maior. Assim, selecionando o neurônio com o maior produto interno, determinamos a localização onde a vizinhança topológica dos neurônios excitados deve ser centrada.

Um critério para definir o melhor casamento entre os vetores **x** e $\mathbf{w}_j$ é minimizar a distância euclidiana entre eles. O índice $i(\mathbf{w}_j)$ será utilizado para identificar o neurônio que melhor se casa com o vetor de entrada **x**, então seja

$$i(\mathbf{x}) = \arg\min_j \left\| \mathbf{x} - \mathbf{w}_j \right\|, \tag{3}$$

onde o neurônio particular *i* é chamado de *neurônio vencedor* ou BMU (*Best-Matching Unit*) para o vetor de entrada **x**.

Em geral, o SOM utiliza a técnica de casamento entre padrões para dar crédito às entradas similares, formando um grupo especializado. A similaridade entre padrões é calculada pela medida da distância euclidiana. Quando um padrão de entrada é avaliado como não similar em relação aos agrupamentos anteriormente formados, um novo agrupamento é estabelecido. Seu padrão é guardado como definição para o novo e as avaliações subsequentes de similaridade serão em relação a esse novo padrão.

A ideia de formar agrupamentos visa maximizar a separação entre eles. Durante a formação dos agrupamentos, a distância euclidiana é utilizada para fornecer crédito aos padrões com características mais similares.

#### *3.5.1.2 Processo de cooperação*

O neurônio vencedor tem a função de localizar o centro de um agrupamento mais próximo ao padrão de entrada e atualizar a vizinhança topológica de neurônios cooperativos.

A definição correta de uma vizinhança topológica, do ponto de vista neurobiológico, vem da evidência de uma interação lateral entre um agrupamento de neurônios excitados. Um neurônio que está emitindo um estímulo (sinapse) tende a excitar com maior intensidade os neurônios na sua vizinhança imediata do que aqueles distantes dele. Então, deve-se fazer com que a vizinhança topológica em torno do neurônio vencedor *i* decaia suavemente com a distância lateral.

Considere que $h_{i,j}$ represente a vizinhança topológica centrada no neurônio vencedor *i* e que contenha um conjunto de neurônios excitados (cooperativos), sendo um neurônio típico desse conjunto representado por *j*.

Seja $d_{i,j}$ a representação da distância lateral entre o neurônio vencedor *i* e o neurônio excitador *j*. Assumindo que a vizinhança topológica $h_{i,j}$ é uma função unimodal da distância $d_{i,j}$, pois satisfaz as seguintes exigências:

- A vizinhança topológica $h_{i,j}$ é simétrica em relação ao ponto máximo definido por $d_{i,j}=0$, ou seja, alcança o seu valor máximo de excitação no neurônio vencedor *i* quando a distância $d_{i,j}$ é zero.
- A amplitude da excitação da vizinhança topológica $h_{i,j}$ decresce monotonicamente com o aumento da distância lateral $d_{i,j}$, decaindo a zero quando a distância tende ao infinito. Esta condição é necessária para a convergência.

Uma escolha típica de $h_{i,j}$ é a função gaussiana, ou seja,

$$h_{i(\mathbf{x})} = e^{\left( -\frac{d^2_{j,i(\mathbf{x})}}{2\sigma^2} \right)}, \tag{4}$$

que é invariante à translação (independentemente da localização do neurônio vencedor). O parâmetro σ é a "largura efetiva" da vizinhança topológica, medindo o grau com que os neurônios excitados dentro da vizinhança do vencedor participam do processo de aprendizagem. Em um sentido qualitativo, a vizinhança topológica gaussiana da Equação 4 é mais apropriada biologicamente do que uma vizinhança definida como retangular (HAYKIN, 2001).

Para que a cooperação entre os neurônios vizinhos se mantenha, é necessário que a vizinhança topológica $h_{i,j}$ seja dependente da distância lateral $d_{i,j}$ entre o neurônio vencedor $i$ e o neurônio excitado $j$ no espaço de saída. Conforme a Equação 4, para o caso de uma rede bidimensional, a distância é definida por

$$d_{j,i}^{2} = \left\| \mathbf{r}_j - \mathbf{r}_i \right\|^2 , \tag{5}$$

onde o vetor $\mathbf{r}_j$ define a posição do neurônio excitado $j$ e $\mathbf{r}_i$ define a posição do neurônio vencedor $i$, sendo ambos medidos no espaço de saída.

Outra característica importante do SOM é que o tamanho da vizinhança topológica diminui com o tempo. Essa exigência é satisfeita fazendo-se com que a largura σ da função de vizinhança $h_{i,j}$ diminua com o tempo, durante a execução do algoritmo de treinamento.

#### *3.5.1.3 Processo de adaptação*

Por último, tem-se o processo que realiza adaptação dos pesos sinápticos para a formação auto-organizada do mapa de características. Para que a rede seja auto-organizável, é necessário que o vetor de peso sináptico $\mathbf{w}_j$ do neurônio $j$ da rede se modifique em relação ao vetor de entrada $\mathbf{x}$.

Usando um formalismo de tempo discreto, dado o vetor de peso sináptico $\mathbf{w}_j(n)$, do neurônio $j$ no tempo $n$, o vetor de peso atualizado $\mathbf{w}_j(n+1)$, no tempo $n+1$, é definido por

$$\mathbf{w}_j(n+1) = \mathbf{w}_j(n) + \eta(n) h_{j,i(\mathbf{x})}(\mathbf{x})(\mathbf{x} - \mathbf{w}_j(n)), \tag{6}$$

onde $\eta(n)$ é o parâmetro da taxa de aprendizagem, $h_{j,i(\mathbf{x})}(\mathbf{x})$ é função da vizinhança topológica. Essa equação deve ser aplicada a todos os neurônios da rede que se encontram dentro da vizinhança topológica do neurônio vencedor $i$.

A adaptação tem o efeito de mover o vetor peso sináptico $\mathbf{w}_i$ do neurônio vencedor $i$ em direção ao vetor de entrada $\mathbf{x}$. Para isso, deve ser aplicada a apresentação repetitiva dos dados de entrada. Desse modo, os vetores de pesos sinápticos tendem a seguir a distribuição dos vetores de entrada devido à atualização da vizinhança.

Nessa parte do algoritmo, leva-se a uma *ordenação topológica* do mapa de características, no sentido de que neurônios adjacentes da rede tenderão a ter vetores de peso sináptico similares. Na Figura 3.8, é apresentado um exemplo do funcionamento do processo adaptativo.

Após ser definido o neurônio "vencedor" (BMU), os demais neurônios da mesma vizinhança são ajustados em direção ao dado de entrada $\mathbf{x}$. Durante a adaptação, o SOM consegue gradualmente, a partir de um estado inicial em completa desordem (inicialização aleatória), atingir uma representação organizada com base nos padrões do espaço de entrada.

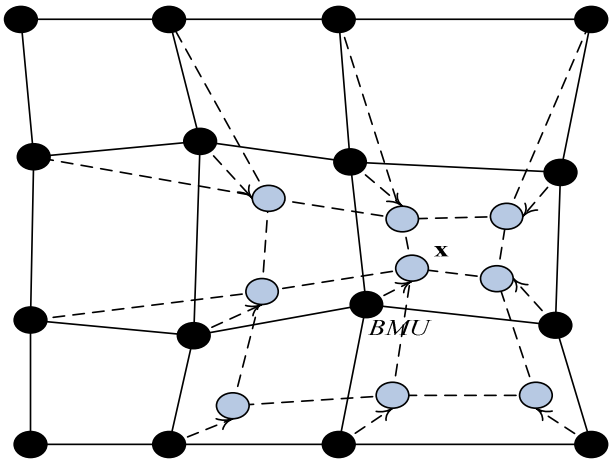



Figura 3.8: Processo de adaptação dos pesos sinápticos.

Além disso, a adaptação dos pesos sinápticos calculada de acordo com a Equação 6, pode ser decomposta em duas fases:

- *Fase de auto-organização* ou *de ordenação*: é durante esta primeira fase do processo adaptativo que ocorre a ordenação topológica dos vetores de peso. A fase de ordenação pode exigir 1.000 iterações do algoritmo SOM e, possivelmente, até mais. É preciso prestar atenção à escolha do parâmetro de aprendizagem e à função de vizinhança.
- *Fase de convergência*: esta segunda fase do processo adaptativo é necessária para realizar uma sintonia fina no mapa de características e, assim, produzir uma quantização estatística precisa do espaço de entrada. Portanto, essa fase pode durar milhares de iterações.

Mais detalhes sobre as heurísticas utilizadas durante o funcionamento do algoritmo podem ser vistos em (HAYKIN, 2001) e (KOHONEN, 2001).

### 3.5.2 Algoritmo de treinamento

O algoritmo padrão de treinamento do SOM consiste na apresentação das três fases em uma sequência preestabelecida. Na primeira fase, *processo de competição*, os neurônios da camada de saída competem entre si atribuindo créditos para cada um, seguindo o critério da distância euclidiana para encontrar o neurônio *vencedor*. Na segunda fase, *processo de cooperação*, é definida uma vizinhança topológica para o neurônio vencedor. Na terceira e última fase, *processo adaptativo*, os vetores de peso sináptico do neurônio vencedor e de sua vizinhança são ajustados.

Seja $X$ o conjunto de padrões de entrada composto pelos vetores $\mathbf{x}_k$, $k = 1,..., m$, onde $m$ é a quantidade de vetores de entrada, tem-se o seguinte algoritmo:

```
Iniciar os vetores de pesos sinápticos com valores aleatórios e
parâmetros η e h;
Para cada época n, faça um treinamento:
    Para todo x_k ∈ X para o tempo discreto n, faça
        Localizar o neurônio vencedor i seguindo o critério da
        distância euclidiana (Equação 3). A ordem de apresentação
        dos vetores de entrada deve ser aleatória.
        Atualizar os vetores de pesos sinápticos w_i do neurônio
        vencedor e dos vizinhos conforme a Equação 6.
    Fim
```

```
    Atualizar os parâmetros η e h.
Fim
```

*3.5.2.1 Avaliação do treinamento*

Após o treinamento, é importante um método para avaliar a qualidade do mapa gerado. O *erro médio de quantização* ($Em_q$) corresponde à média do erro correspondente à diferença entre o vetor de pesos sinápticos $\mathbf{x}_k$ e o vetor vencedor $i(\mathbf{x}_k)$, seguindo a equação

$$Em_q = \frac{\sum_{k=1}^{N} \left\| \mathbf{x}_k - \mathbf{w}_{i(\mathbf{x})} \right\|}{N}, \tag{7}$$

onde $N$ é a quantidade de vetores de entrada e $\mathbf{w}_{i(\mathbf{x})}$ é o vetor de peso sináptico do vencedor.

A medida do *erro médio de quantização* revela a similaridade dos dados de entrada em relação ao conhecimento adquirido pelo SOM durante o treinamento. É importante para acompanhar a evolução das adaptações durante as várias épocas de treinamento.

Quando é utilizado apenas um padrão como vetor de entrada, o *erro de quantização* revela a similaridade deste em relação ao mapa. Quando um valor baixo desse erro é obtido, representa uma grande similaridade com os dados treinados e pleno reconhecimento pelo mapa. Um valor alto do erro representa uma dissimilaridade ou desvio de características, revelando que o mapa não conhece as características do padrão apresentado (KOHONEN ET AL., 1996).

A equação do *erro de quantização* fica conforme a seguir

$$E_q = \left\| \mathbf{x}_k - \mathbf{w}_{i(\mathbf{x})} \right\|, \tag{8}$$

onde $\mathbf{w}_{i(\mathbf{x})}$ é o vetor de peso sináptico do vencedor.

A representação gráfica dessa equação é visualizada na Figura 3.9. No gráfico, é mostrado o espaço de características de um SOM treinado e o vetor de entrada $\mathbf{x}(t)$ que se aproxima do SOM. O Erro de Quantização ($E_q$) é apresentado como sendo a diferença vetorial entre o vetor de entrada e a aproximação feita pelo SOM (neurônio vencedor).

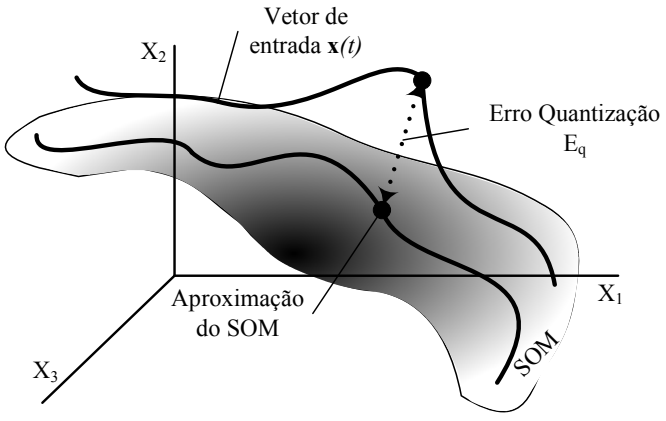



Figura 3.9: Ilustração do cálculo do erro de quantização.

No algoritmo padrão de treinamento, os vetores de entrada do conjunto *X* devem ser inseridos de forma aleatória durante o treinamento, para que haja garantia de uniformidade na apresentação dos padrões.

Define-se uma quantidade de épocas para carregar os vetores de entrada à rede. Além disso, se aplica uma normalização nos vetores de entrada.

O *processo de competição,* dentre as três fases do treinamento, é o que apresenta maior custo computacional. Nessa fase, é aplicada uma busca sequencial em todo o mapa para localizar o neurônio vencedor.

A dimensão do mapa auto-organizável e a dimensão *N* dos vetores dependem do problema da aplicação. Na literatura, existem diversas propostas para determinar a dimensão do SOM, mas normalmente é realizado por um processo empírico. Em (VESANTO, JUHA, 1999) (ENDO ET AL., 2000), foi proposta uma técnica automática que visa analisar os dados de entrada e encontrar uma dimensão que minimize o custo de memória e de processamento, apresentando resultados de treinamento satisfatório.

Outra informação importante para definir a dimensão do mapa é a quantidade de amostras de entrada como padrões utilizados para treinamento. Quando são apresentados grandes volumes de amostras de entrada, geralmente é mais indicado utilizar mapas maiores. Todavia, mapas grandes comprometem o desempenho do algoritmo e mapas muito pequenos comprometem a formação topológica.

A determinação dos parâmetros da Equação 6 é feita de modo empírico e fortemente baseada nos experimentos e métodos de tentativa e erro. A taxa de aprendizagem $\eta(n)$ assume um valor pré-fixado $\eta(0)<1$, e deve decair com o tempo $n$ até um valor próximo de zero. Analogamente, a função de vizinhança $h_{j,i(\mathbf{x})}$ também assume um valor pré-fixado que seja adequado para maximizar a qualidade da formação dos agrupamentos no mapa. Mais detalhes dos parâmetros consultar (HAYKIN, 2001) e (KOHONEN, 2001).

### 3.5.3 Propriedades do mapa de características

Uma vez treinado, o *mapa* gerado apresenta algumas características estatísticas que são importantes de serem revisadas conforme (HAYKIN, 2001):

1. *Aproximação do espaço de entrada*: o mapa de características deve fornecer uma aproximação favorável ao espaço de entrada. O objetivo básico é armazenar um grande conjunto de vetores de entrada, em um conjunto reduzido de neurônios, de modo que forneça uma boa aproximação para o espaço de entrada original. Essa ideia está fundamentada na *teoria da quantização vetorial*, cuja motivação é a redução da dimensionalidade ou compressão de dados.
2. *Ordenação topológica*: o mapa de característica deve estar topologicamente ordenado, no sentido de que a localização espacial de um neurônio corresponda a um domínio particular ou a uma característica dos padrões de entrada. Isso é devido à Equação 6 forçar o neurônio vencedor a mover-se em direção ao vetor de entrada, o que por consequência move também os neurônios mais próximos dentro da vizinhança.
3. *Casamento de densidade*: o mapa de características deve refletir as variações da distribuição de entrada. Regiões do espaço de entrada com uma alta probabilidade de ocorrência (estímulos que ocorrem com frequência) serão

mapeadas para domínios maiores no mapa, do que regiões com baixa probabilidade de ocorrência.

### 3.5.4 Interpretação do resultado do SOM

Embora o SOM não apresente um módulo em particular para resultados de saída, como no caso das redes neurais de *aprendizagem supervisionada*, é necessária uma etapa de processamento para extrair os resultados, conhecida por *teste*, *atuação* ou *recuperação*.

Dependendo da aplicação-alvo utilizada, a resposta do sistema pode ser tanto o índice ou o peso sináptico do neurônio vencedor. O índice do neurônio vencedor infere a localização espacial de acordo com a ordem topológica, indicando em qual agrupamento do mapa o dado de entrada foi mapeado. Este é o princípio para a classificação de padrões. O peso sináptico do neurônio vencedor é uma generalização para o dado de entrada, fornecendo uma aproximação para o espaço de entrada.

Após o processo de treinamento, também se deseja avaliar visualmente o resultado da formação topológica e a densidade do mapa. Destacam-se aqui duas formas de representação visual propostas por (KOHONEN ET AL., 1996) (VESANTO, JUHA, 1999) (HAYKIN, 2001): a mais simples é reproduzir graficamente os vetores de pesos sinápticos em um espaço de coordenadas; e a outra forma é a matriz de distância entre os vetores de pesos sinápticos que reproduz uma imagem (fotografia) do mapa.

#### *3.5.4.1 Algoritmo de teste*

Este algoritmo é importante para se extrair resultados do SOM.

Seja $T$ o conjunto de padrões de entrada de teste composto pelos vetores $\mathbf{x}_k$, $k = 1,...,m$, onde $m$ é a quantidade de vetores de teste. Tem-se o seguinte algoritmo:

```
Para cada x_k ∈ T faça
    Localizar o neurônio vencedor i seguindo o critério da
    distância euclidiana (Equação 3);
    Escrever na saída o peso sináptico do neurônio vencedor;
    Escrever na saída o índice do neurônio vencedor;
    Escrever na saída o valor da distância euclidiana como erro de
    quantização;
Fim
Escrever na saída o valor do erro de quantização médio.
```

#### *3.5.4.2 Vetores de pesos sinápticos no espaço $R^d$*

Para o caso da dimensão dos vetores de pesos sinápticos ser $d \leq 3$, pode-se adotar o uso destes como coordenadas de um produto cartesiano em um espaço dimensional $R^d$ para à visualização da organização dos neurônios.

Seja a Figura 3.10 a sequência de treinamento, em que foram reproduzidas graficamente as coordenadas dos neurônios em épocas diferentes, em uma dimensão $d=2$. A imagem superior à esquerda, corresponde ao mapa inicializado aleatoriamente revelando nenhuma uniformidade. Durante o treinamento, pode-se visualizar a convergência para um estado de equilíbrio formando uma grade. Como resultado final desse treinamento, após várias épocas, é visualizada, na imagem inferior à direita, uma formação mais uniforme da rede e oposta à inicial. Isso deve-se ao SOM atingir um ponto de equilíbrio durante o treinamento.

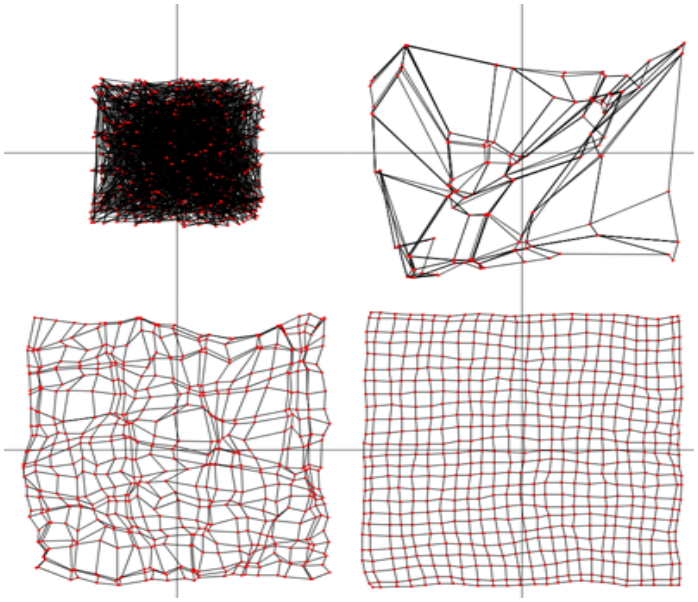


Figura 3.10: Exemplo de sequência de treinamento, representando os pesos sinápticos de um SOM em épocas diferentes.

### *3.5.4.3 Matriz de distância unificada*

Para os casos em que é necessário avaliar as formações topológicas dos neurônios no mapa, quando a dimensão dos vetores de pesos sinápticos é $d>3$, o método utilizado é o da *matriz de distâncias unificada* (*U-Matrix – Unified Distance Matrix*) (VESANTO, JUHA, 1999) (ENDO ET AL., 2000) (SILVA, 2004). A *U-Matrix* é uma matriz composta pelas distâncias entre todos os neurônios vizinhos, em que tais distâncias são obtidas pela média aritmética entre os pesos sinápticos de toda a vizinhança do neurônio e de seu próprio peso.

A *U-Matrix* é representada por uma imagem do mapa, em que o nível de intensidade de cada *pixel* corresponde a uma distância calculada. Um mapa 2-D, de dimensão $N$x$M$, gera uma imagem $(2N-1)$x$(2M-1)$.

Na imagem, a coloração dos *pixels* é de acordo com a intensidade de cada componente da matriz. Valores altos (cores vermelhas) correspondem a neurônios vizinhos dissimilares, que equivale à formação de fronteiras no mapa. Valores baixos (cores azuis) correspondem a neurônios vizinhos similares, correspondendo à formação de vales onde se agrupam neurônios com padrões semelhantes.

Na Figura 3.11, é apresentado o exemplo de uma *U-Matrix*, para uma rede neural treinada para classificação das subespécies da planta Íris, um problema clássico de redes neurais. Na figura, é visível a existência de fronteiras de separação entre agrupamentos. Quanto mais dissimilares forem os agrupamentos em relação aos vizinhos, maior será a distinção entre eles e a fronteira. Essa técnica é importante para a descoberta de similaridades entre os agrupamentos.

Utilizar a *U-Matrix* como saída tem por objetivo permitir a detecção visual das relações topológicas dos neurônios (SILVA, 2004). Essa técnica é extremamente útil quando os vetores de pesos sinápticos têm dimensão superior a três.

Pelo fato da *U-Matrix* ser uma imagem relativamente complexa para alguns problemas, pode ser difícil a visualização e a interpretação para usuários pouco

experientes. Existem alguns estudos que visam realizar um processamento sobre o mapa, a fim de fazer uma detecção automática da formação dos agrupamentos por meio de técnicas de processamento de imagens (SILVA, 2004).

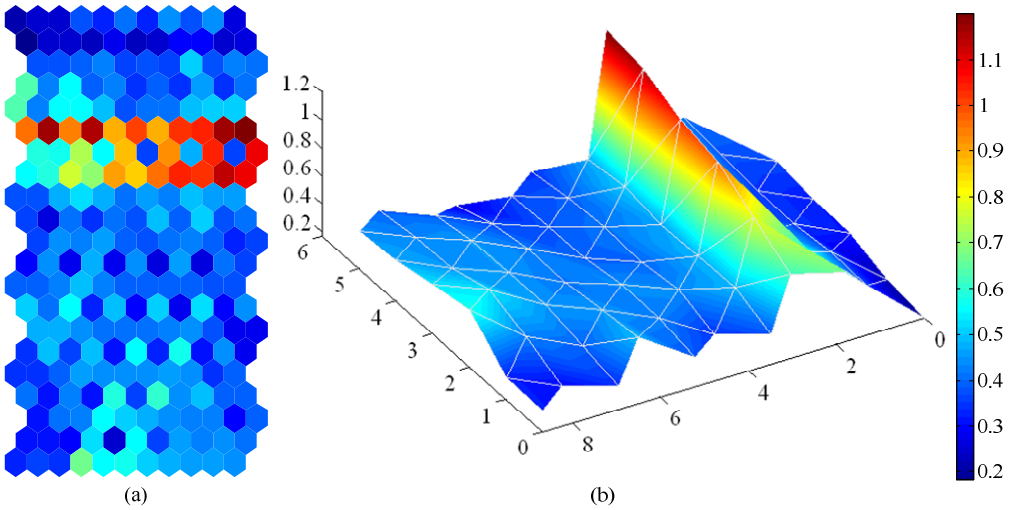



Figura 3.11: Exemplo de visualização pela *U-Matrix*. (a) visualização padrão em duas dimensões e (b) visualização em três dimensões.

## 3.6 Mapa Temporal de Kohonen

A proposta original de Kohonen em criar o SOM foi baseada no processamento de dados *estáticos*, que são relacionados *espacialmente*. Contudo, também pode existir uma correlação *temporal* entre esses dados, para o caso dos padrões que ocorrem sequencialmente no tempo, em que um padrão individual não tenha significado sozinho. Para tais padrões, denominados *temporais* ou *dinâmicos*, a ordem em que eles são observados no tempo e o conjunto de dados que precedem ou seguem, tem um papel fundamental e deve ser levado em consideração pelo modelo de processamento neural (BASTOS, 2007).

Dessa forma, é importante utilizar, neste trabalho, um caso de rede neural que tenha incorporado operações de processamento temporal, pois muitas das tarefas no mundo real acontecem em tempos diferentes. Por exemplo, nos sistemas industriais as tarefas são desempenhadas por equipamentos eletrônicos que analisam variáveis de sensores em instantes de tempo diferentes e acionam atuadores em tempos diversos (usina hidroelétrica, computador, automóvel, sistema de um avião, etc.). Além disso, os seres humanos também tomam decisões e têm respostas comportamentais a estímulos ocorridos no espaço e no tempo (visão, fala, decisões empresariais, economia, etc.). A capacidade de processar tais padrões temporais deve ser uma propriedade relevante em um sistema inteligente.

Nos últimos anos, as pesquisas se concentraram em redes neurais com base no aprendizado supervisionado para processarem informações temporais, entre elas cita-se a Rede Elman (ELMAN, 1990). Todavia, existem alguns modelos para lidar com sequências temporais em redes não-supervisionadas, apenas recentemente essa classe de redes neurais tem recebido atenção dos pesquisadores.

Neste trabalho, adotou-se seguir a mesma linha de raciocínio e utilizar uma rede neural baseada na aprendizagem não-supervisionada para tratar a questão temporal. Seus conceitos e fundamentos são apresentados a seguir.

### 3.6.1 Conceitos básicos de séries temporais

De acordo com (BARRETO, 1998) (BARRETO, 2002) (BARRETO ET AL., 2001), um padrão temporal é uma sequência espaço-temporal formada por um conjunto finito e ordenado no tempo, composto por *N* vetores de características. Para o propósito computacional, assume-se que uma sequência temporal é composta de componentes discretos ordenados no tempo:

$$X_N = \{\mathbf{x}(t), \mathbf{x}(t-1), \mathbf{x}(t-2), \cdots, \mathbf{x}(t-N)\}, \quad (9)$$

onde cada componente é denotado por $\mathbf{x}(t) = [x_1(t), x_2(t), \cdots, x_n(t)]^T$ e a variável *t* é usada para representar um instante particular de tempo.

O termo *processamento de séries temporais* diz respeito a algumas das seguintes tarefas:

- Classificação: a rede produz um sinal em resposta a uma sequência de entrada quando esta encontrar similaridade com algum padrão na rede. Pode ser chamada também de reconhecimento de sequências.
- Teste: a rede deve reproduzir previamente a sequência aprendida na ordem temporal correta.
- Predição: estimar itens do futuro de uma dada sequência de entrada com base nos valores observados no passado.

As séries temporais têm algumas propriedades devido à diversidade dos sinais temporais. O conhecimento dessas propriedades é útil para entender as aplicações envolvidas nas redes neurais temporais:

1. *Ordem*: posição relativa de cada item da sequência em relação ao outro.
2. *Métrica*: duração de um padrão no tempo em relação a outros padrões da sequência.
3. *Densidade*: número de vetores de características que compõem uma sequência, ou seja, a taxa de amostragem.
4. *Assimetria*: sentido do tempo não pode ser invertido.
5. *Contexto temporal*: é a menor subsequência formada por itens passados que possibilitam determinar sem ambiguidades o item atual da sequência.
6. *Grau*: número de itens que compõem o contexto temporal.

Essas propriedades estão envolvidas geralmente na construção do modelo, sendo codificadas como parâmetros.

#### *3.6.1.1 Memória de curta duração*

Para estabelecer associações e extrair correlações temporais entre padrões consecutivos de uma dada sequência, uma rede neural deve ser capaz de reter e manter provisoriamente informações sobre itens de um passado próximo. O mecanismo de

retenção é denominado de *memória de curto prazo* (STM – *short-term memory*) e é amplamente utilizado no processamento de sequências temporais (BARRETO, 2002).

Basicamente, memórias de curto prazo convertem informação temporal em espacial, possibilitando que redes estáticas convencionais sejam empregadas em tarefas de processamento de informação temporal. Essa abordagem proporciona construir um sistema dinâmico não-linear, que forneça uma clara separação de responsabilidades: a rede estática responsável pela não-linearidade e a memória responsável pelo tempo.

Uma memória de curto prazo não tem nenhum tipo de aprendizado e, por essa razão, o seu comportamento é totalmente determinado pelos parâmetros de funcionamento do modelo.

Em (BARRETO, 2002) é apresentado um modelo de memória de curto prazo, baseado na utilização de integradores na entrada ou na saída da rede. Tem-se que o vetor de entrada da rede passa a ser representado por $\bar{\mathbf{x}}(t)$ e os itens atuais da sequência por $\mathbf{x}(t)$, então a função que expressa a relação com as entradas anteriores em um instante $t$ fica:

$$\bar{\mathbf{x}}(t) = (1 - \lambda)\bar{\mathbf{x}}(t-1) + \lambda \mathbf{x}(t), \tag{10}$$

onde $\lambda$ é a constante de integração e $0<\lambda\leq 1$ é chamada de *profundidade da memória*, pois determina o grau de influência das entradas anteriores sobre a entrada atual. A determinação desse parâmetro é feita por tentativa e erro em experimentos empíricos.

Quando a rede TKM está com o parâmetro ($\lambda=1$), ela se comporta como um SOM convencional com entradas estáticas. Quando a rede TKM está com o parâmetro ($0<\lambda\leq 1$), ela se comporta dinamicamente. No primeiro caso, separa-se de um conjunto de treinamento as categorias, de acordo com semelhanças espaciais, enquanto no segundo, separam-se os vetores de entrada por uma combinação de semelhança espacial e proximidade temporal, em que a contribuição relativa dos dois fatores é determinada pelo parâmetro $\lambda$.

Em (KANGAS, 1990), ao utilizar a rede SOM original, foi substituído o vetor de entrada original $\mathbf{x}(t)$ por uma versão temporal de acordo com a Equação 10. O SOM modificado (inserção de uma memória de curto prazo) foi aplicado no problema de reconhecimento de fonemas e obteve taxa de acertos superiores ao SOM convencional.

Para mais detalhes sobre os principais conceitos de séries temporais e modelos de memórias de curto prazo relatados na literatura, consultar os trabalhos de (MOSER, 2004) (BASTOS, 2007) (BARRETO, 1998) (BARRETO, 2002) (BARRETO ET AL., 2001).

### 3.6.2 Algoritmo de Treinamento

Neste trabalho, foi adotada a utilização do Mapa Temporal de Kohonen (TKM – *Temporal Kohonen Map*), que é uma proposta de (CHAPPEL; TAYLOR, 1993). Eles modificaram o SOM original por meio da inclusão de uma memória de curto prazo, com base em integradores na saída de cada neurônio da rede (Equação 10).

A rede agora apresenta uma técnica que mantém o histórico de ativações $a_i(t)$ para cada um dos neurônios da seguinte forma:

$$a_j(t) = (1 - \lambda)a_j(t-1) - \frac{1}{2}\left\|\mathbf{x}(t) - \mathbf{w}_j(t)\right\|^2, \tag{11}$$

onde $0<\lambda\leq 1$ é a profundidade da memória, $\mathbf{x}(t)$ é o item atual da sequência e $\mathbf{w}_j(t)$ é o vetor de peso sináptico do neurônio $j$. Um neurônio da rede TKM calcula sua ativação no instante $t$, que decai a uma taxa dada pela constante de tempo $|1/(1-\lambda)|$.

O neurônio vencedor é calculado e localizado pela máxima ativação dos dados no histórico, que são formados pela entrada atual e pelos vetores de entrada prévios. Então, cada sequência é processada mapeando um vetor por instante de tempo, em que o último neurônio vencedor serve para representar a sequência inteira. O cálculo do vencedor é conforme a equação:

$$i\{\mathbf{x}(t)\} = \max_{j \in A} \{a_j(t)\}, \tag{12}$$

onde, a vizinhança topológica e a atualização dos pesos sinápticos são ajustadas da mesma forma em que o SOM convencional, conforme visto na seção 3.5.

O algoritmo de treinamento para o TKM segue as mesmas etapas vistas na seção 3.5. Entretanto, com uma modificação no momento da escolha dos neurônios vencedores, em que além de se calcular a distância euclidiana, também é realizada uma avaliação da combinação espaço-temporal da série de entrada. É importante lembrar que no TKM a entrada da rede neural, agora, é composta por uma sequência de dados, e não apenas por um dado estático como no SOM convencional.

A estratégia utilizada pelo TKM resulta em neurônios que respondem a sequências específicas. (CHAPPEL; TAYLOR, 1993) aplicaram o TKM com sucesso no reconhecimento de fonemas que ocorrem em uma mesma posição, porém em contextos diferentes. Por exemplo, as sequências **a-b-c-d** e **r-s-t-d** são classificadas como distintas por que a letra **d** tem claramente contextos temporais distintos.

Entretanto, o TKM apresenta algumas falhas na identificação de sequências que requerem informação contextual de longo prazo e gera ambiguidade em todos os casos que tenham a mesma média móvel. Isso ocorre porque a determinação do neurônio vencedor depende principalmente das entradas mais recentes e a profundidade da memória é limitada pelo parâmetro $\lambda$.

Em (BARRETO ET AL., 2001), é apresentadao uma revisão das principais aplicações para o qual o TKM foi utilizado para solucionar problemas. Por exemplo, o TKM foi aplicado em uma arquitetura de controle adaptativa para aprender um trajeto de um caminho percorrido por um robô. Também foi aplicado para localizar objetos de duas dimensões em imagens que são captadas em tempo real por uma linha de produção para classificação de produtos.

### 3.6.3 Algoritmo de Teste

Seja $T$ uma série de padrões de entrada para teste, com janela temporal $t$, composta pelos vetores $\mathbf{x}_k$, $k = 1,\ldots, t$, tem-se o seguinte algoritmo:

```
Para cada série∈T faça
    Para cada k de 1 até t faça
        Calcular a distância euclidiana entre x_k e os neurônios;
        Calcular ativação para os neurônios, utilizando as ativações
        passadas, conforme Equação 11.
    Fim
    Localizar o neurônio com a maior ativação e defini-lo como
    vencedor, conforme Equação 12;
    Escrever na saída o peso sináptico do neurônio vencedor;
```

```
    Escrever na saída o índice do neurônio vencedor;
Fim
```

### 3.6.4 Resultados do TKM

A interpretação dos resultados para o TKM é realizada da mesma forma como é realizada no SOM convencional. Como resultados a serem extraídos têm-se, os pesos sinápticos como coordenadas em um gráfico, ou a imagem da matriz de distância unificada (U-Matrix).

A principal diferença entre os resultados do TKM e o SOM convencional está na correlação espaço-temporal entre cada amostra de entrada. Para o caso da U-Matrix, pode-se projetar a trajetória da formação dos neurônios vencedores, que proporciona visualizar o caminho percorrido pelos dados entre os agrupamentos. Nesse caso, pode-se identificar uma tendência (ou padrão) de comportamento, acompanhando a evolução em um período de tempo. Ver Figura 3.12(a).

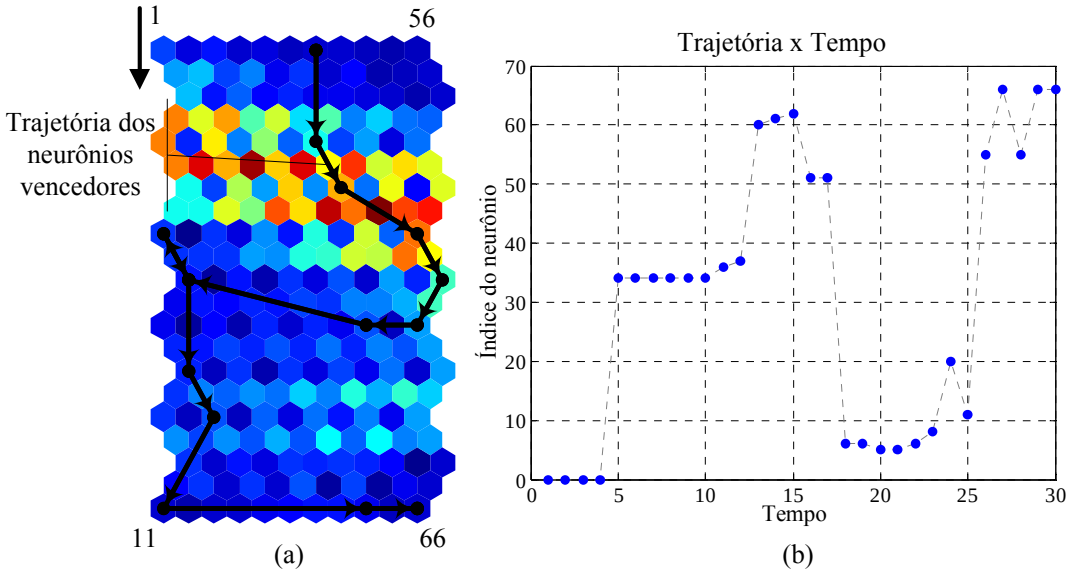



Figura 3.12: (a) Visualização da trajetória dos neurônios vencedores e (b) localização da trajetória em cada instante de tempo.

O TKM também pode ser interpretado por meio da projeção em um gráfico da relação entre os vencedores para cada série em um período de tempo. Além disso, proporciona a observação do instante em que ocorrem as mudanças de vencedores, identificando uma tendência para a classificação das séries. Geralmente, quando ocorre a troca de vencedores, quer dizer que a série de entrada também sofreu modificações em relação a seus valores prévios. Isso é importante para analisar as mudanças de comportamento do sistema. Ver Figura 3.12(b).

Outra análise que pode ser feita sobre o gráfico da relação entre os vencedores, em um período de tempo, é avaliar a generalização do TKM para diferentes amostras de um mesmo tipo de dado. Essas amostras podem ter sofrido ruídos externos. Quando o gráfico é produzido, a generalização do TKM é avaliada comparando o resultado de cada série com as outras, onde deve-se obter a mesma sequência de neurônios vencedores entre elas.

Isso diz respeito ao TKM ter a capacidade de reconhecer e identificar corretamente as séries de amostras, mesmo pertencendo a situações semelhantes e com incidência de ruídos deve apresentar o mesmo comportamento.

### 3.6.5 Comparação entre o SOM e TKM

As redes neurais SOM e TKM têm muitas semelhanças entre si. Do ponto de vista do algoritmo de treinamento, a única etapa que apresenta diferenças no algoritmo é a de competição, em que ocorre a escolha do neurônio vencedor.

Do ponto de vista do tratamento dos dados de entrada, o SOM convencional trabalha com dados estáticos e processa um padrão de entrada por vez. Já o TKM processa uma série de padrões de entrada, em que o neurônio vencedor é escolhido sempre quando o último padrão da série é apresentado para a rede.

A etapa de teste também apresenta diferenças significativas, em que no SOM padrão usa-se basicamente a distância euclidiana, e no TKM necessita-se, além da distância euclidiana, do cálculo do histórico de ativações para cada neurônio.

Na Tabela 3.2, é apresentada uma comparação evidenciando as principais diferenças entre as duas abordagens.

Tabela 3.2: Comparação entre SOM e TKM.

| | **SOM** | **TKM** |
|---|---|---|
| **Padrões de entrada** | Trata os padrões de entrada como dados independentes. | Trata séries de dados como padrões de entrada arranjados em um período de tempo. |
| **Tratamento dos dados** | Dados avaliados apenas nas questões espaciais. | Dados avaliados nas questões espaciais e temporais. |
| **Neurônio vencedor** | Menor distância euclidiana em relação ao padrão de entrada e os neurônios. | Critério da maior ativação para uma série como entrada em relação aos neurônios. |
| **Competição** | Calcular a distância euclidiana para todos os neurônios do mapa. Escolhe o neurônio vencedor. | Além da distância euclidiana entre todos os neurônios do mapa, são calculadas as ativações. |
| **Cooperação** | Definição da vizinhança de formação topológica. | Idem ao SOM. |
| **Adaptação** | Atualização dos pesos sinápticos dos neurônios. | Idem ao SOM. |
| **Recuperação** | Baseada na menor distância euclidiana. | Baseada na maior ativação. |

## 3.7 Motivação de usar o SOM para Analisar Falhas

Segundo (KOHONEN ET AL., 1996) o SOM tem algumas características importantes que proporcionam o monitoramento da ocorrência de falhas em processos dinâmicos. Por exemplo, um processo de produção industrial, em que, devido à grande

quantidade de variáveis e dados coletados, as falhas se mostram difíceis de analisar. Então, é necessário um processamento nesses dados que viabilize o estudo das relações entre as informações adquiridas.

O modelo do SOM implementa uma projeção não-linear de características de um espaço sensorial em um arranjo de neurônios com dimensão reduzida. Além de mapear sinais de entrada com dimensão elevada em uma estrutura com dimensão inferior, o SOM assegura a forma ordenada dos dados, durante o mapeamento, preservando as relações topológicas no domínio do sinal. Devido a isso, tende a criar agrupamentos de informações e relacionamentos dos dados de entrada em uma representação espacial formando um mapa.

A mais importante característica do SOM é a visualização de sistemas e processos com dimensões elevadas e a descoberta de categorias e abstrações nos dados de entrada (KOHONEN ET AL., 1996) (DÍAZ, IGNACIO ET AL., 2008).

Para exemplificar uma aplicação-chave do SOM em casos reais, cita-se o campo da engenharia, que envolve muitas áreas da vida do ser humano, como as indústrias, os veículos, as aeronaves, os equipamentos médicos, etc. Uma importante aplicação do SOM no campo da engenharia é seu uso na identificação e no monitoramento de falhas, cuja ideia é acompanhar a evolução dos estados de operação em máquinas ou processos complexos, que são muito difíceis de observar e interpretar (KOHONEN ET AL., 1996). Neste trabalho, é abordada a aplicação do SOM em ambientes industriais, sobretudo, no processo de transporte em uma refinaria de petróleo.

Em uma planta ou máquina industrial complexa, os possíveis estados de suas variáveis, durante a operação, podem ser de ordem numérica muito elevada e serem expressos de uma forma não-linear. Entretanto, é muito difícil elaborar um modelo analítico que reflita todas as particularidades de comportamento do processo. Como o SOM é um método de projeção não-linear, as características de cada estado podem ser processadas e passam a ser conhecidas pelo SOM sem a necessidade de uma modelagem explícita de todo o comportamento do sistema a ser monitorado (DÍAZ, IGNACIO ET AL., 2008).

Muitas das pesquisas que focam na análise do comportamento de processos (monitorar o estado, detectar, identificar e diagnosticar as falhas e, em trabalhos mais recentes, prever estados ou falhas), têm adotado princípios de aprendizagem não-supervisionada para resolver tais problemas. Essa tendência é função principalmente de não existirem informações disponíveis *a priori* sobre as situações de falhas.

Nas seções a seguir, serão apresentados alguns trabalhos relacionados ao uso do SOM em problemas de engenharia, que visam detectar e diagnosticar falhas, monitorar e prever comportamentos futuros em sistemas industriais.

### 3.7.1 Detectar falhas

Durante o treinamento do SOM, os pesos sinápticos dos neurônios se adaptam de acordo com o domínio do espaço de estados refletidos pelas amostras adquiridas. Assim, o espaço dos estados pode ser dividido em duas partes: *espaço reconhecido* pelo SOM e o *espaço complementar*. Por exemplo, no SOM treinado com dados normais, somente situações que incluem dados normais serão identificadas (espaço reconhecido), caso seja alguma situação de falha, esta será alocada em algum lugar nas bordas do mapa (espaço complementar).

Devido a tal comportamento do SOM, a detecção de falhas pode ser baseada na medida do *erro de quantização* (ver Equação 8), lembrando que é preciso utilizar apenas um único vetor de entrada por vez. Na maioria dos trabalhos que utiliza o SOM para detectar anomalias ou falhas no sistema, este é treinado apenas para reconhecer os dados do *modo de operação normal*. Quando for apresentado na entrada um vetor diferente de *normal*, este terá uma grande dissimilaridade em relação às informações armazenadas pelo SOM.

Considere que uma sequência de vetores de entrada, que correspondam a informações medidas e coletadas no sistema, é apresentada à entrada do SOM. Cada vetor da sequência é comparado com o peso sináptico de todos os neurônios do mapa. Se a menor distância euclidiana calculada exceder um limite predeterminado, isso é um indício de que o processo está, provavelmente, em situação de falha.

Essa conclusão está baseada na hipótese de que um erro de quantização grande corresponde a um ponto de operação pertencente ao espaço complementar não coberto pelos dados utilizados no treinamento (QIU, HAI ET AL., 2003).

Para concluir, a detecção de falhas ou de qualquer outra anomalia está baseada na afirmação de que tudo o que não é *normal* é considerado como *anormal*. Geralmente, esta técnica é aplicada juntamente com a de diagnóstico de falhas, que visa identificar qual a falha ocorrida no sistema após a detecção.

A seguir, são apresentados alguns trabalhos relacionados que fazem uso da detecção de falhas utilizando o SOM como ferramenta:

- Em (BARRETO ET AL., 2004), o SOM é utilizado como ferramenta para detecção de falhas em infraestrutura de transmissão de dados em redes de celulares 3G. A detecção é realizada pelo erro de quantização, que extrai um valor como principal medida da relação entre as falhas. Para uma falha ser detectada, é preciso que o erro de quantização extrapole um limite preestabelecido.

- Outro trabalho que utiliza a distância euclidiana é o de (QIU; LEE, 2004), que utiliza o SOM para detectar a degradação e estimar o tempo restante de vida útil de equipamentos mecânicos que utilizam rolamentos. O sinal de vibração é adquirido nos rolamentos por meio de um acelerômetro e são pré-processados pela transformada *Wavelet.* O erro de quantização é utilizado para detectar a degradação e as falhas que ocorrerem no sistema. O treinamento do SOM utiliza sinais em condições normais de funcionamento. Vale ressaltar que a medida do erro de quantização também é utilizada como taxa de degradação do sistema e, mais tarde, para calcular o tempo de vida restante do equipamento.

- Outro trabalho interessante que utiliza o SOM para detectar falhas em rolamentos internos de motores de indução é apresentado por (ZHONG ET AL., 2005). São coletados sinais de vibração dos rolamentos e projetados quatro casos de teste para experimentos, um utilizando rolamentos normais e três rolamentos com falhas injetadas. Os sinais são pré-processados pela Transformada de Fourier (FFT). No trabalho, o SOM é treinado com um conjunto de dados referentes aos modos de comportamento normal e com falhas nos rolamentos. As falhas são detectadas monitorando o nível do erro de quantização ao longo do tempo.

- Outra técnica é apresentada por em (WONG ET AL., 2006) para detectar anomalias em sistemas. O SOM detecta falhas em sistemas industriais que tenham partes mecânicas rotativas por meio de análise dos sinais de vibração. No trabalho, existe um mapa para cada tipo de falha. O sinal é pré-processado pela FFT e a seguir aplicado como entrada no SOM. Modificando-se o SOM de forma a apresentar como resposta não apenas um neurônio vencedor, mas todos aqueles neurônios que pertençam à sua vizinhança topológica. Estes são aplicados em uma comparação que utiliza uma nova medida, a distância de *Mahalanobis*, que substitui a distância euclidiana para definição dos vencedores.

### 3.7.2 Diagnosticar falha

Para o diagnóstico de falhas pelo SOM é preciso que a rede neural tenha capacidade de identificar e "discernir" as condições de operação do sistema em análise. É desejável que seja capaz de separar entre os estados de operação normal, degradação e falha .

O principal problema do diagnóstico está em como determinar se uma informação é anormal. As falhas podem ser muito raras de ocorrer, mas quando ocorrem podem causar grandes problemas e prejuízos. Para o SOM ter condições de classificar uma falha, é preciso que tais informações estejam disponíveis para treinamento, a fim de que os neurônios se adaptem ao domínio de falhas, criando, assim, um *espaço de operação reconhecido* pelo SOM, que forma o espaço das falhas.

Só que em alguns casos, as informações sobre degradação e falhas não estão disponíveis, devido a muitas circunstâncias, como custo, falta de conhecimento técnico, dificuldade em coletar sinal, etc. Desse modo, produzir um conjunto de falhas abrangente para o caso de processos industriais e grandes máquinas, pode exigir um custo muito elevado, inviabilizando todo o projeto. Entretanto, existe a alternativa em que muitas das situações em degradação ou falha podem ser produzidas por uma ferramenta de simulação.

Se a situação da falha e suas razões são conhecidas, é possível simular tais dados para serem utilizados para o treinamento do SOM. Em (KOHONEN ET AL., 1996), são apresentados alguns dos tipos mais comuns de falhas que podem ocorrer no campo da engenharia:

1. Suspensão do fornecimento de um sinal, resultando em uma queda abrupta do serviço.
2. Distúrbios pesados no sinal, causando uma grande mudança esporádica no fornecimento do serviço.
3. Perturbação nas medidas de um valor, frequentemente causada por uma falha mecânica no equipamento de medição, que prejudica o fornecimento do serviço.
4. Piora da qualidade dos sinais, devido à degradação (envelhecimento) natural do dispositivo, causando perda de desempenho no fornecimento do serviço.
5. Mudanças repentinas em alguns valores de parâmetros previamente regulados, indicando uma falha no controle do mecanismo.

Todos esses tipos de falhas mencionadas são possíveis de ser simulado, mas é claro que depende muito da situação, do tipo do equipamento, das causas da falha, etc.

Assumindo que seja possível construir um modelo matemático para simular o comportamento de falhas, pode-se criar artificialmente os vetores de características usados para treinamento do SOM. Por meio do simulador, pode-se, por exemplo, causar distúrbios mecânicos ou elétricos no modelo do equipamento, gerar sinais para situações "perigosas" ou até mesmo injetar falhas únicas.

Caso as informações sobre as falhas mais típicas estejam disponíveis, o SOM pode ser usado como uma ferramenta classificadora das do sistema (KOHONEN ET AL., 1996).

Em muitos trabalhos que fazem uso do SOM para diagnosticar falhas, este é utilizado em dois níveis para realizar uma análise completa das situações de falhas. O primeiro nível, a *detecção de falhas*, é baseado no erro de quantização, em que se aplica a mesma técnica vista na subseção anterior. O segundo nível utiliza um mapa mais detalhado (pode-se chamar de "atlas", pois apresenta todas as situações possíveis) que é usado para identificar a razão da ocorrência de uma falha. Quando um sinal é identificado em uma região do mapa, e essa região está rotulada como falha (X ou Y) ou degradação (X ou Y), é possível classificar o sinal apresentado na entrada como pertencente à classe da falha rotulada no mapa.

O diagnóstico de falhas faz uso de um SOM treinado com um conjunto completo de informações sobre a situação normal, degradação (caso exista) e a falha (essencial). Em um mesmo mapa, falhas múltiplas podem ser inseridas, uma diferente da outra, proporcionando ao mapa o poder de classificar um número maior de falhas (QIU, HAI ET AL., 2003).

Para concluir, o segredo do diagnóstico do SOM está ligado diretamente ao conjunto de informações utilizadas para o treinamento, ou seja, quanto mais completas e abrangentes forem às características para as diversas falhas, melhor será o diagnóstico.

Para realizar a classificação, o SOM faz uso do algoritmo de teste (seção 3.5.4.1), para identifica qual grupo pertence o sinal, apresentando como saída o índice do neurônio vencedor. Também pode-se utilizar o mapa de características visto pelo U-Matrix (seção 3.5.4.3) em que é possível visualizar a localização espacial do neurônio que classificou a falha.

A seguir, serão apresentados alguns trabalhos relacionados que fazem uso da detecção e classificação para diagnosticar falhas em um sistema utilizando o SOM como ferramenta:

- No trabalho de Vachkov (VACHKOV ET AL., 2004), o SOM é utilizado para diagnosticar falhas em partes hidráulicas de máquinas retro-escavadeiras, analisando variáveis de velocidade do motor, força exercida para levantar a pá, pressão de óleo e consumo de combustível. A detecção de falhas também é realizada através da medida da distância euclidiana. Para o diagnóstico de falhas, utiliza-se o mesmo sinal de entrada apresentado durante a detecção e é aplicado para a classificação pelo SOM. Durante o treinamento, os sinais são mapeados em agrupamentos separados que representam cada um, um modo de operação do equipamento. Foram feitos experimentos por meio de uma ferramenta de simulação, em razão do alto custo de injetar falhas em uma retro-escavadeira real.
- Outro trabalho que analisa falhas em ambientes industriais é o de (SAXENA; SAAD, 2004), em que o SOM é utilizado para classificar as falhas em

rolamentos industriais e, ao mesmo tempo, monitorar a condição de funcionamento ao longo do tempo. O SOM é treinado com dados de vibração coletados de rolamentos em boas condições e demais que apresentam falhas injetadas. O diagnóstico e monitoramento da condição são feitos com base no uso da trajetória dos neurônios vencedores, que apresenta a evolução de uma tendência para o funcionamento dos rolamentos, possibilitando identificar para qual grupo de falhas está se deslocando o comportamento do rolamento (diagnóstico). Não utiliza rede neural temporal.

- Uma proposta de modificação no funcionamento do SOM foi apresentada por (YANG, B. S. ET AL., 2004), em que uma nova rede neural, chamada de ART-SOM, é aplicada para diagnosticar falhas em máquinas rotativas (rolamentos e mancais), analisando dados de vibração. O ART-SOM tem uma mudança na sua dinâmica de funcionamento, pois aplica técnicas para treinamentos subsequentes de uma forma adaptativa durante a etapa de testes da rede, o que possibilita que o treinamento e o diagnóstico de falhas sejam realizados juntos. Durante o treinamento dessa rede, são criados os agrupamentos que aprenderão os dados de entrada, possibilitando identificar e classificar os padrões de falhas. Foram realizados experimentos reais e uma taxa de referência foi calculada para avaliar o sucesso da classificação das falhas.

- O trabalho de (ZHONG ET AL., 2005), apresentado na seção anterior, também realiza o diagnóstico de falhas em rolamentos internos de motores de indução. São utilizados os mesmos sinais de vibração e pré-processados pela Transformada de Fourier (FFT). O SOM é treinado com um conjunto de dados referente a todos os modos de operação dos rolamentos, efetuando o aprendizado dos sinais normais e das falhas. Durante o treinamento, são formados agrupamentos distintos para cada tipo de falha injetada. O diagnóstico das falhas é realizado pela identificação espacial de qual agrupamento o sinal foi mapeado, baseado no critério da menor distância euclidiana.

- Uma análise comparativa de três técnicas para classificação de falhas foi apresentado por (YANG, BO-SUK ET AL., 2005). O estudo de caso foi feito com base na ocorrência de falhas em compressores de refrigeração, por meio da análise de ruído e vibração presentes no equipamento. O trabalho é focado em uma comparação entre três técnicas de redes neurais para classificação de padrões, SOM, LVQ (*Learning Vector Quantization)* e SVM (*Support Vector Machine*). Todas as redes neurais são treinadas com sinais iguais e pré-processadas pela Transformada *Wavelet*. Os resultados dos experimentos mostraram que o SOM apresentou resultados satisfatórios para a classificação de falhas, mas a que obteve os melhores resultados foi a rede neural SVM, e o menos satisfatório foi a LVQ.

- Com os avanços tecnológicos e principalmente a inserção da Internet em redes industriais, o trabalho de (DOMÍNGUEZ, M. ET AL., 2007) apresenta uma proposta de arquitetura para um sistema distribuído de monitoramento da condição de equipamentos, tendo como principal função a detecção e diagnóstico de falhas. Essa arquitetura é baseada nos padrões atuais de tecnologia de desenvolvimento de sistemas, sendo um sistema totalmente

para a *Web*, que visa utilizar a Internet como canal de comunicação para fornecer flexibilidade, interoperabilidade e escalabilidade ao sistema. O SOM é inserido em uma camada localizada no meio, entre o Cliente (*hardware* e aquisição de sinais) e o Servidor (Sistema de Monitoramento Central), em que a camada do SOM tem a tarefa de realizar o treinamento da rede neural, detecção e diagnóstico de falhas. A detecção de falhas também é feita pelo erro de quantização, semelhante ao apresentado na seção anterior. O diagnóstico é realizado por uma projeção da trajetória utilizada para monitorar a condição de operação do equipamento ao longo do tempo de operação.

- No trabalho de (GÉRMEN ET AL., 2007), o SOM é utilizado para detectar e classificar falhas de quebra de rotores e problemas mecânicos em motores de indução, que são largamente utilizados nas indústrias. O sinal utilizado é a amostra de corrente nominal do motor, que é pré-processado e utilizado como entrada para treinamento ou testes do SOM. A classificação dos sinais pelo SOM é feita por meio do algoritmo LVQ (*Learning Vector Quantization*), uma técnica desenvolvida por Kohonen para delimitar e ajustar a formação de fronteiras nos agrupamentos formados no mapa. O trabalho apresenta experimentos reais aplicados em motores com boas condições e motores com falhas injetadas fisicamente (causando dano no motor). O resultado alcançado foi de 100% de acerto na classificação de falhas ocorridas nos testes.

### 3.7.3 Predição e Monitoramento do comportamento

Entender e modelar as relações complexas entre as diversas variáveis que formam um grande sistema é uma tarefa que exige muito esforço e de grande dificuldade.

Segundo (DÍAZ, IGNACIO ET AL., 2008), as informações que são adquiridas por um sistema de aquisição de dados, produzem uma enorme quantidade de informações ao longo do tempo, que se torna muito difícil ou mesmo impossível de ser interpretada. Em tal situação, o mínimo conhecimento a respeito de alguma característica do sistema se torna muito útil.

No entanto existem técnicas de processamento de dados que auxiliam nesta análise. As informações coletadas podem ser convertidas em uma forma que auxilie na simples visualização e compreensão, em que mesmo com uma possível redução da dimensionalidade, preserva os relacionamentos estatísticos entre as variáveis do sistema. Com esse tipo de transformação disponível seria possível:

- O operador visualizar e acompanhar o desenvolvimento dos estados do sistema ao longo do tempo.
- A compreensão dos dados facilita a estimativa de comportamentos futuros do sistema.
- Anormalidade no comportamento presente ou previsto do processo torna possível a detecção de situações de falhas.
- O controle do sistema pode ser baseado na análise dos estados.

Com a utilização do SOM, é possível atender a essas afirmações. É uma ferramenta importante para monitorar o comportamento de sistemas complexos, possibilita analisar

e localizar relações entre as variáveis e da dinâmica do processo. Além disso, abre um campo de pesquisa sobre análise exploratória do comportamento de processos não-lineares e não-estacionários (KOHONEN ET AL., 1996) (GOH ET AL., 2006).

Os métodos de visualização presentes no SOM são uma poderosa ferramenta para descobrir relacionamentos nas estruturas gerais do espaço de dados de entrada, além de revelar o comportamento do sistema. No mapa resultante do SOM, cada neurônio passa a representar um estado de comportamento local do processo.

Para monitorar o comportamento, o SOM faz uso das mesmas informações e formas de treinamento utilizadas para o diagnóstico de falhas visto na subseção anterior. A novidade está em se projetar uma trajetória percorrida pelos neurônios vencedores. A trajetória é formada pelo armazenamento da sequência de neurônios vencedores, cujos vetores de entrada são apresentados seguindo uma ordem temporal (KOHONEN ET AL., 1996).

Uma trajetória consiste em formar um conjunto de imagens sucessivas a respeito dos estados de operação do sistema. A trajetória pode ser desenhada e visualizada no mapa de características do SOM (U-Matrix), indicando os estados percorridos pelo sistema ao longo do tempo. Além disso, percorrendo esse caminho, pode-se, por exemplo, inferir os valores futuros dos estados do processo, pois a tendência se concretiza no decorrer do tempo. Isso possibilita analisar com mais detalhes o comportamento do processo antes e durante a ocorrência da falha.

Para concluir, a principal ideia de monitorar o comportamento é identificar a condição atual do sistema e a tendência do que acontecerá caso o sistema continue se desviando da normalidade. Para realizar o monitoramento e projetar a trajetória de tendência, o SOM faz uso dos princípios do diagnóstico e a projeção da imagem U-Matrix para desenhar a trajetória espacial.

A seguir, serão apresentados alguns trabalhos relacionados que fazem uso do SOM para realizar o monitoramento do comportamento para evitar falhas em um sistema:

- Um trabalho aplicado integralmente em um problema real de uma empresa petroquímica foi apresentado por (JÄMSÄ-JOUNELA ET AL., 2003). O SOM foi utilizado para monitorar o funcionamento de um processo de produção em uma empresa petroquímica, a fim de evitar as falhas que provocam paradas na produção. No trabalho, os sinais são processados para que a rede neural aprenda a probabilidade do estado mais provável do sistema. Entretanto, a principal contribuição do trabalho é a de desenvolver um protótipo de um sistema para monitoramento em *software*, em que é apresentado aos usuários uma aplicação que mostra o mapa do SOM (U-Matrix), gráficos de tendência, falhas ativas num instante, histórico de falhas, uma interface para treinamento, ajustes de parâmetros e formulas utilizadas pelo algoritmo do SOM. O diagnóstico de falhas é baseado em uma série de regras condicionais que conferem os níveis de probabilidade de uma falha ocorrer.
- O SOM também é apresentado como uma ferramenta para representação visual de processos dinâmicos no trabalho elaborado por (VACHKOV ET AL., 2004). Pelo mapa resultante, cada neurônio representa um modo de comportamento local da condição do processo. Isso proporciona analisar e localizar relações entre as variáveis e da dinâmica do processo, abrindo

espaço de pesquisa para análise exploratória do comportamento em processos não-lineares e não-estacionários. Devido às características intrínsecas do SOM, como a redução da dimensionalidade e visualização de dados com alta dimensão, ele tem sido aplicado com sucesso em problemas de engenharia, como na análise e supervisão de processos industriais. O trabalho focou no uso do SOM para revelar a distribuição de parâmetros do processo para os diferentes estados alcançáveis durante sua operação. Foram apresentados resultados práticos para um estudo de caso de um sistema de controle do nível de líquidos em um reservatório.

- O monitoramento da condição em sistemas possibilita também analisar a formação de uma tendência e, com isso, estimar o tempo de vida restante de funcionamento do processo. No trabalho de (QIU, HAI ET AL., 2006), o SOM é utilizado para monitorar a degradação e estimar o tempo de vida útil restante do equipamento. Como estudo de caso foram utilizados mancais com rolamentos. O sinal de vibração é adquirido por um acelerômetro e são pré-processados pela Transformada *Wavelet Packet.* O erro de quantização é utilizado para detectar o comportamento da degradação no sistema. A trajetória percorrida pelos neurônios vencedores é utilizada para monitorar os estados de operação do equipamento. A principal contribuição do trabalho é utilizar a medida do erro de quantização, que também é utilizada como referencial da taxa de degradação, para calcular o tempo de vida restante do equipamento. Utilizando essa técnica, é necessário ter informações *a priori* de todo o ciclo de vida do rolamento, pois as informações coletadas devem refletir os sinais desde o início da operação, deteriorando até falhar naturalmente. Como experimentos, foram apresentados quatro casos de testes reais, em que foram coletadas informações de todo o ciclo de vida. Com base nessas informações, foi calculado o tempo de vida útil e um valor limite que o rolamento possa desempenhar suas funções dentro de condições normais, sem prejudicar o fornecimento de serviços.
- A integração de outras redes neurais ao SOM também foi estudada por alguns pesquisadores. Em (HUANG, R. ET AL., 2007), a rede neural *back propagation* é integrada na saída do SOM. Essa ideia consiste em monitorar o comportamento, além de detectar e diagnosticar as falhas, fazer uma estimação do tempo de vida útil restante do equipamento. São analisados sinais de vibração em rolamentos mecânicos. Para estimar o tempo restante, utiliza-se o erro de quantização do SOM, que revela o comportamento da degradação até a ocorrência de falhas (quando exceder um limite predeterminado). A rede neural *back propagation* é treinada utilizando o erro de quantização como saída do SOM para fazer uma aproximação de função desses sinais, possibilitando, com isso, estimar o valor em qualquer tempo. É importante lembrar que a aquisição de dados foi feita com a utilização de um equipamento real, a partir de todo o ciclo de vida, desde o início até a ocorrência da falha. Esse trabalho é uma continuação das pesquisas de (QIU, HAI ET AL., 2003).

## 3.8 Resumo do Capítulo

Neste capítulo, foi abordada uma grande quantidade de informações referentes a conceitos, fundamentos e técnicas de implementação de detecção, diagnóstico, predição e monitoramento de falhas aplicadas em sistemas. Tais conceitos são uma parte das pesquisas do campo da tolerância a falhas em sistemas que podem ser aplicados também em sistemas industriais, que é o foco deste trabalho. Todavia, não se limitam à aplicação apenas na área de engenharia, pois podem, inclusive, abranger qualquer ambiente onde se deseja aumentar os índices de confiabilidade.

Aumentar a confiabilidade em sistemas industriais tem se mostrado um grande desafio para os engenheiros projetistas desses ambientes. Entretanto, uma tarefa de grande importância que aumenta esses índices nos sistemas industriais é a de manter uma política de manutenção sempre atualizada. Hoje, conta-se com sistemas de manutenção inteligente, que são sistemas computacionais específicos para monitorar partes do processo industrial, em busca de anomalias que possam afetar o correto desempenho e causar prejuízos ao sistema produtivo. Esses sistemas inteligentes são utilizados pelos engenheiros para auxiliar a equipe de manutenção e para estarem sempre à frente das manifestações de falhas.

Os ambientes industriais são conhecidos por ser estarem sujeitos a muitas interferências externas que causam degradação no desempenho, prejudicam o correto funcionamento, a paralisação casual, etc. A manutenção é uma técnica que visa reparar ou restaurar o correto funcionamento dos processos industriais, retornando ao desempenho especificado no projeto.

O foco do presente trabalho está no estudo e projeto de ferramentas alternativas para melhorar o desempenho das ações de manutenção sobre os equipamentos. A motivação básica é que, descobrindo com antecedência a possibilidade de ocorrência de falha e sabendo onde esta se manifestará, tem-se uma informação muito valiosa para a equipe responsável pelos reparos nos equipamentos.

Com a implantação de sistemas de manutenção inteligente, é possível aumentar a confiabilidade nos sistemas industriais, ou seja, aplicar técnicas de tolerância a falhas. A principal função desses sistemas é a detecção, diagnóstico, predição e monitoramento de falhas que podem ocorrer nos equipamentos que compõem um processo industrial.

Existem diversas abordagens que podem ser utilizadas para solucionar cada um dos problemas nos ambientes industriais, mas, neste trabalho, adotou-se o uso de um sistema embarcado computacional. Este tem por tarefa aplicar técnicas de inteligência artificial para realizar os conceitos de tolerância a falhas.

Um Sistema de DDF ou DDPF proporciona ganhos significativos no desempenho do processo industrial, melhorando a disponibilidade, confiabilidade, segurança e mantenabilidade dos processos produtivos. Esse sistema também tem como objetivo a prevenção, a tolerância, a remoção, o diagnóstico, o monitoramento e a predição das falhas em processos industriais.

Um Sistema de DDPF aplica técnicas automáticas para detectar e estimar a degradação no desempenho de sistemas físicos, antecipar a ocorrência de falhas futuras, e calcular o tempo restante de vida do sistema, mantendo-o em um estado operacional aceitável. Essa proposta tem uma modificação em relação ao sistema DDF, que é a adição de uma estrutura para aquisição de conhecimentos ou históricos sobre o comportamento passado do sistema.

Por meio de conhecimentos adquiridos, *a priori*, sobre o funcionamento físico e comportamento dos equipamentos, pode-se criar um modelo matemático capaz de reproduzir o histórico de comportamentos do sistema. Deste modo, o Sistema DDPF pode ser implementado pelo uso de redes neurais, que são utilizadas para armazenar os conhecimentos ou históricos. As redes neurais aprendem o comportamento do sistema, adquirindo conhecimento a partir dos estados passados. Assume-se que o processo industrial apresenta o mesmo comportamento em tempos futuros.

No presente trabalho, foi adotado o uso da rede neural conhecida por mapas auto-organizáveis (SOM) como ferramental capaz de adquirir conhecimento sobre informações do ambiente e que possui características que possibilitam detectar, diagnosticar e, com algumas modificações, até prever a ocorrência das falhas. O SOM é uma rede neural não-supervisionada que é baseada apenas nas informações presentes nos dados de entrada utilizados para treinamento.

Outras características importantes para a escolha do SOM foram:

- Capacidade de tratar dados em grandes quantidades (geralmente dados históricos aparecem em grande quantidade).
- Ótima aproximação do espaço de entrada,
- Preserva a ordem topológica dos dados de entrada mesmo com redução da dimensionalidade,
- Descoberta de novas características nos dados de entrada (revelando os relacionamentos presentes entre os dados de entrada).

Também foi apresentada a rede neural para tratamento de informações temporais, que leva em consideração, durante operação, a ordem de tempo em que os dados são observados. O *Temporal Kohonen Maps* (TKM) é uma proposta baseada no SOM, pois com algumas modificações é possível tratar dados temporais. Foram apresentados conceitos básicos de tempo e memória de curto prazo, que são importantes para o entendimento dessa rede neural.

Por fim, foram apresentados alguns trabalhos práticos encontrados na literatura científica que utilizam o SOM como ferramenta para analisar as falhas em processos industriais.

# 4 PROPOSTA DE PROTÓTIPO DE UM SISTEMA DE MANUTENÇÃO INTELIGENTE PARA ATUADORES ELÉTRICOS

## 4.1 Introdução

Neste capítulo, apresenta-se a construção de um protótipo em uma plataforma de sistema embarcado, com o objetivo de aplicar as técnicas de manutenção inteligente em uma válvula elétrica utilizada em redes dutoviárias para transporte de petróleo. A plataforma adotada para o sistema embarcado é baseada na tecnologia de FPGA.

Esse protótipo foi projetado de modo a ser reutilizado e integrado por outras ferramentas ou plataformas, e a ser inserido no projeto geral do sistema de manutenção inteligente.

Primeiro, foi definida a arquitetura do sistema de detecção, diagnóstico e predição de falhas, do ponto de vista da utilização do SOM como técnica computacional para realizar essa tarefa. Ao longo deste capítulo, são apresentados, com detalhes, cada um dos algoritmos responsáveis por realizar as tarefas de tolerância a falhas, preparando a base para o desenvolvimento do protótipo em um sistema embarcado.

Em segundo lugar, foi desenvolvido um conjunto de ferramentas em *software* para MATLAB, a fim de possibilitar a simulação do funcionamento integral do sistema de manutenção inteligente. Por meio de experimentos, validou-se a arquitetura do sistema de detecção, diagnóstico e predição e realizou-se a avaliação nos conjuntos de dados utilizados para treinamento e testes do SOM.

Após a validação do projeto em *software*, foi iniciada a construção do protótipo do sistema de manutenção inteligente. Ademais, foi definida a utilização da plataforma FPGA para o sistema embarcado, pois possibilita o desenvolvimento em descrição VHDL e a reutilização dos componentes computacionais internos da plataforma. O desenvolvimento do protótipo do *hardware* do SOM foi baseado na metodologia de projeto para sistemas em FPGA.

Como a rede neural SOM faz parte do sistema de manutenção inteligente, foi um desafio o projeto de implantação desta em um sistema de *hardware*, diga-se FPGA. Sabe-se que as redes neurais exigem muitos recursos computacionais para treinamento, entre eles o elevado uso de memória para armazenar os pesos sinápticos dos neurônios.

Grande parte deste custo computacional é devido à dinâmica de funcionamento do algoritmo do SOM, pois nas implementações de redes neurais é assumido que os recursos computacionais são “infinitos”. Isso exige técnicas para limitar os requisitos do algoritmo, como o uso de memória e treinamento, para que possa ser executado num sistema embarcado.

Após a finalização do projeto do *hardware* do SOM e validado o seu funcionamento, este foi integrado em um sistema embarcado computacional. Este é composto por microprocessador, memória e barramento e executa um sistema operacional embarcado. Por meio de um *software* aplicação foram aplicados casos de teste.

Nas próximas seções, será apresentada cada uma das etapas necessárias para projetar a rede neural SOM em *hardware* e como integrá-la ao sistema embarcado, formando, em nível mais alto, o sistema de manutenção inteligente que pode ser integrado à válvula elétrica.

## 4.2 Proposta do Sistema para Manutenção Inteligente

Apresenta-se, nesta seção, uma visão geral do Sistema de Manutenção Inteligente para Válvulas Elétricas. Esse protótipo será implantado em uma válvula que controla o fluxo de petróleo pela rede dutoviária (oleodutos) de uma Refinaria da Petrobras.

O Sistema de Manutenção Inteligente (SMI) tem como função quatro tarefas:

- Monitorar em tempo real as condições de operação do sistema.
- Estimar uma tendência no comportamento do processo ao longo do tempo até a manifestação da falha.
- Detectar desvios no comportamento de normalidade, evitar falhas.
- Classificar e diagnosticar as causas das falhas.

Na Figura 4.1, é apresentada a estrutura do SMI proposto no presente trabalho, mostrando um panorama dos componentes e tarefas que compõem o sistema.

Essa estrutura do SMI foi baseada na Figura 3.4 que apresentou uma estrutura básica e geral para um sistema de DDPF aplicado para um problema qualquer. A proposta, na Figura 4.1, apresenta poucas diferenças em relação à original utilizada como base. Entre as modificações, destacam-se a inserção de uma tarefa para o Processamento de Sinais e uma mudança na estrutura interna do Processo de Detecção, Diagnóstico e Predição de Falhas.

O SMI realiza a operação de aquisição de dados no Processo de Transporte por Oleodutos e, em seguida, são extraídas as informações sobre o torque exercido pelo motor (neste caso assume-se apenas este sinal) do atuador elétrico presente na válvula. O sinal de torque é aplicado ao Processamento de Sinais para extração da energia dos componentes que formam o sinal. Estes dados são utilizados como padrão de entrada para os demais componentes do sistema. Após essas etapas de pré-processamento, o sinal está pronto e adequado para alimentar o Processo de DDPF.

No Processo de DDPF e no Banco de Conhecimento ou Histórico, está o segredo da técnica de tolerância a falha, em que são feitas as análises e inferências sobre o sinal adquirido.

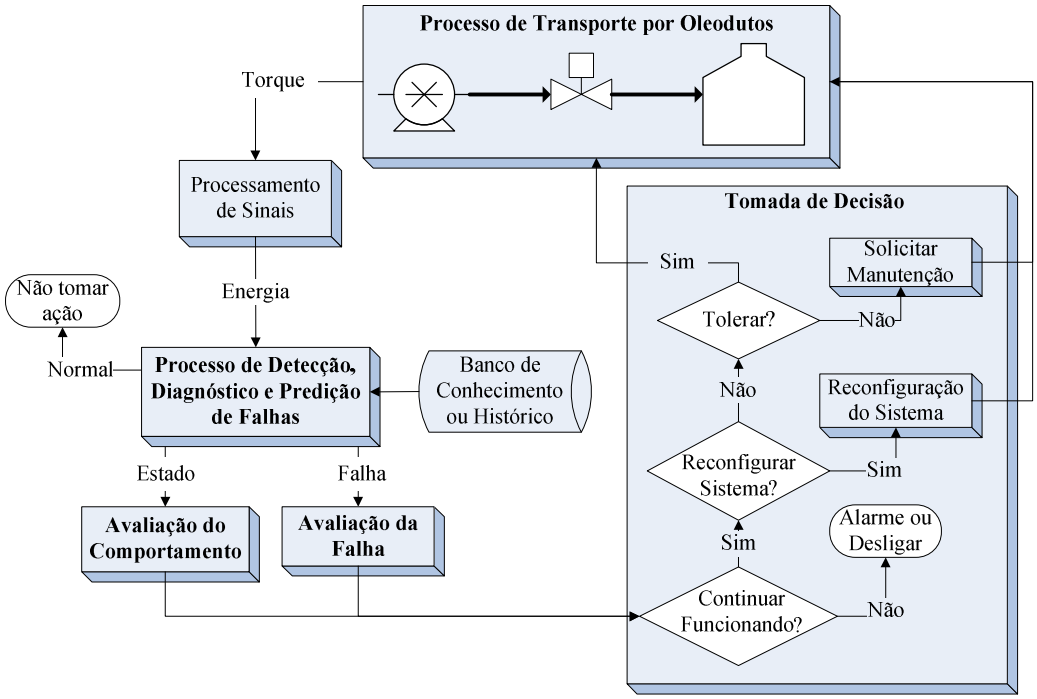



Figura 4.1: Visão geral do Sistema de Manutenção Inteligente para Atuadores Elétricos.

A seguir, após a análise do sinal, é aplicada a Avaliação no resultado obtido, que será utilizada como informação válida para a etapa de Tomada de Decisão. Por fim, o SMI realimenta o Processo de Transporte por Oleodutos e fecha-se o ciclo de monitoramento contínuo das condições de funcionamento do equipamento industrial.

Depois da apresentação deste panorama que forma o SMI, serão exploradas, com mais detalhes, as demais estruturas internas que compõem o projeto, em especial a etapa do Processo de DDPF e do Banco de Conhecimentos ou Histórico. Essas partes formam a base do SMI e são compostas pela aplicação de redes neurais (SOM) como ferramenta para solucionar o problema da tolerância a falhas, conforme a Figura 4.2.

Como o SMI é formado por dois componentes básicos, em primeiro lugar serão abordados os componentes que formam o *Banco de Conhecimento ou Histórico*:

- **SOM para Detecção de Anormalidade**: função de armazenar informações de uma rede neural SOM, que capacite o SMI a aprender o comportamento do modo de operação *normal* do sistema. É a base para a detecção de anormalidades.
- **SOM para Diagnóstico de Falha**: função de armazenar informações de uma rede neural SOM, que capacite o SMI a aprender os padrões que possam distinguir uma falha das outras. É importante notar que quanto mais completo for o conjunto de padrões aprendidos, melhores deverão ser as classificações das falhas. É a base para a classificação e diagnóstico de falhas.
- **TKM para Predição do Comportamento**: função de armazenar informações de uma rede neural TKM, que capacite o sistema a aprender o ciclo de vida completo (*normal*, *degradação* e *falha*) para cada uma das falhas que serão identificadas pelo sistema. É a base para o monitoramento da condição e para projetar uma trajetória de operação ao longo do tempo.

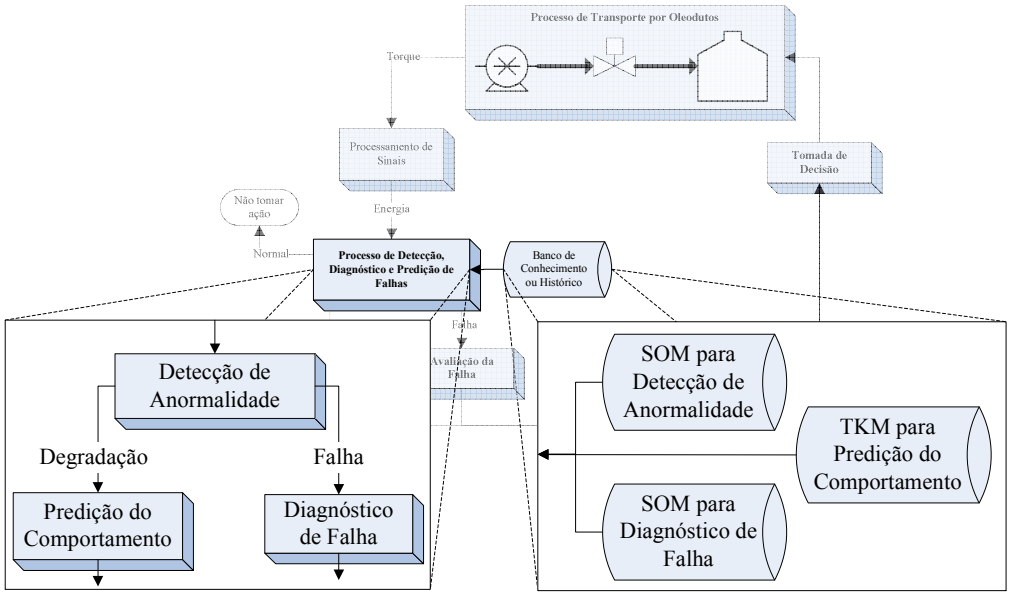



Figura 4.2: Composição interna das partes do Sistema de Manutenção Inteligente para Atuadores Elétricos, em que o SOM é utilizado como base.

Em segundo lugar, serão abordados os componentes que formam o *Processo DDPF*:

- **Detecção de Anormalidade**: função de monitorar o processo até ser detectada alguma condição de anormalidade (possíveis desvios no comportamento). Quando for detectada uma condição anormal, mesmo que mínima, o sistema deve avaliar essa situação como uma possível manifestação de falha ou tendência de degradação. Caso o SMI julgue como *falha*, será acionada a tarefa de Diagnóstico de Falha. Se o SMI julgar como princípio de *degradação*, será acionada a tarefa de Predição do Comportamento.

- **Diagnóstico de Falha**: função de classificar e diagnosticar uma falha detectada. Essa tarefa consiste em identificar qual falha ocorreu com base nos padrões armazenados no Banco de Conhecimento ou Histórico, a fim de identificar uma nova falha que seja semelhante à ocorrida no passado. Depois de identificar a falha, é realizado o diagnóstico, apresentando as possíveis causas e danos que a falha pode causar ao processo. Essa etapa está ligada diretamente ao conhecimento de um especialista sobre as manifestações das falhas.

- **Predição do Comportamento**: função de monitorar a evolução da degradação ao longo do tempo, apontando uma tendência de comportamentos futuros, acompanhando as condições dos estados de operação do sistema. Para avaliar o comportamento do sistema, é utilizado o Banco de Conhecimento ou Histórico, que para cada falha identificada, apresenta informações do ciclo de vida completo. Ademais, a predição possibilita, por exemplo, estimar o tempo restante até a mudança do estado de degradação para falha.

Após explanação das funcionalidades de cada parte que compõe o SMI, será descrito como estas partes foram implementadas. Na Figura 4.3, são apresentadas as redes neurais que compõem o Banco de Conhecimento ou Histórico.

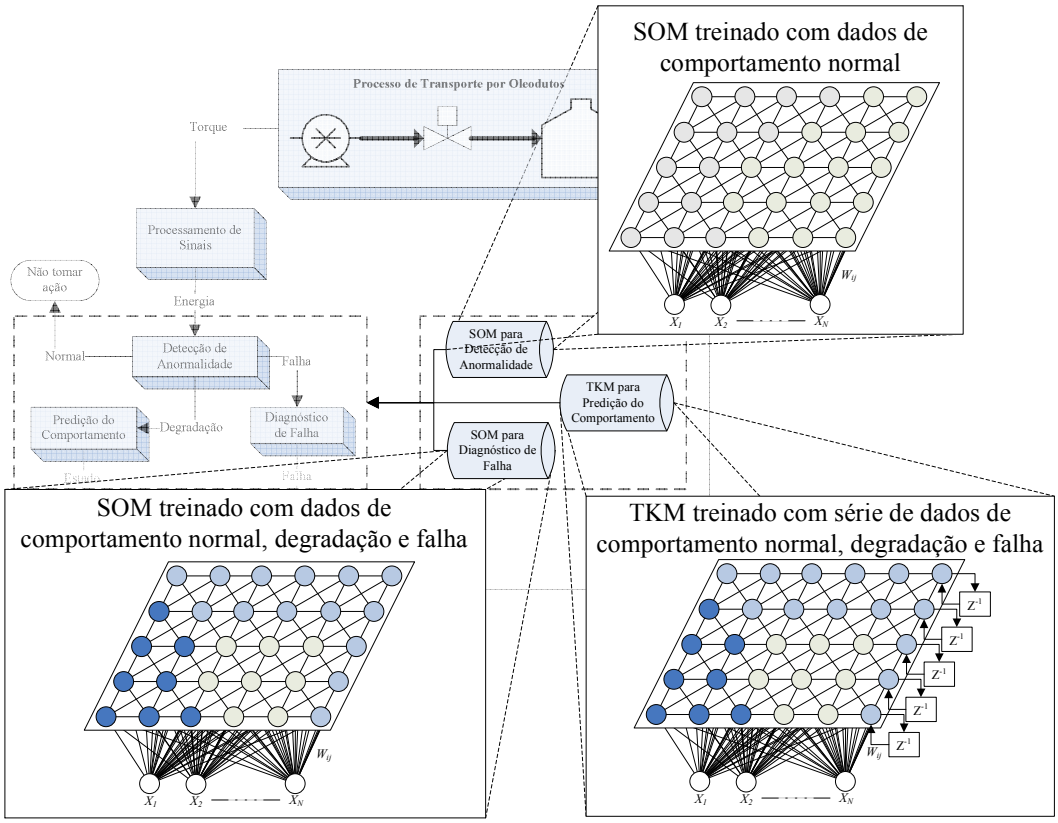



Figura 4.3: Composição do Banco de Conhecimento ou Histórico.

O Banco de Conhecimento ou Histórico é composto por três redes neurais, duas SOM e uma TKM. Essas redes neurais têm por função armazenar o conhecimento sobre o histórico de comportamento do processo em monitoramento. Cada uma delas tem características únicas dentro do processo de tolerância a falhas:

- **SOM para Detecção de Anormalidade**: contém uma rede neural do tipo SOM, que utiliza um conjunto de dados que representam o comportamento *normal* de operação do sistema. Esses dados são utilizados para o treinamento desse SOM, a fim de aprender a identificar apenas dados de normalidade. Esse tipo de treinamento segue os princípios dos trabalhos relacionados, apresentados na Seção 3.7.1.
- **SOM para Diagnóstico de Falha**: contém uma rede neural do tipo SOM, que utiliza um conjunto de dados que representam o ciclo completo de vida de um componente do sistema que será monitorado. Esse ciclo deve representar o comportamento normal, degradação e falha. Esses dados são utilizados para treinamento desse SOM, a fim de aprender a classificar as características de um componente de forma separada. Esse princípio é baseado na classificação de padrões, capaz de identificar os padrões de entrada comparando-os com os dados armazenados nos neurônios. Esse tipo de treinamento segue os princípios dos trabalhos relacionados, apresentados na Seção 3.7.2.
- **TKM para Predição do Comportamento**: contém uma rede neural do tipo TKM, que utiliza série de dados para representar o ciclo completo de vida de um componente do sistema que será monitorado por um período de tempo. Esse ciclo de vida deve representar o comportamento normal, degradação e

falha ao longo do tempo. Esses dados são utilizados para treinamento do TKM, a fim de aprender a classificar os padrões temporais de cada série de dados separadamente. Esse princípio é baseado na classificação de padrões da série temporal pelo TKM. Esse tipo de treinamento segue os algoritmos apresentados na seção 3.6.

Após definição das redes neurais que integram o SMI, será abordado como foram implementadas as partes que extraem os resultados necessários das redes. Na Figura 4.4, estas partes são apresentadas: Processo de Detecção, Diagnóstico e Predição de Falhas:

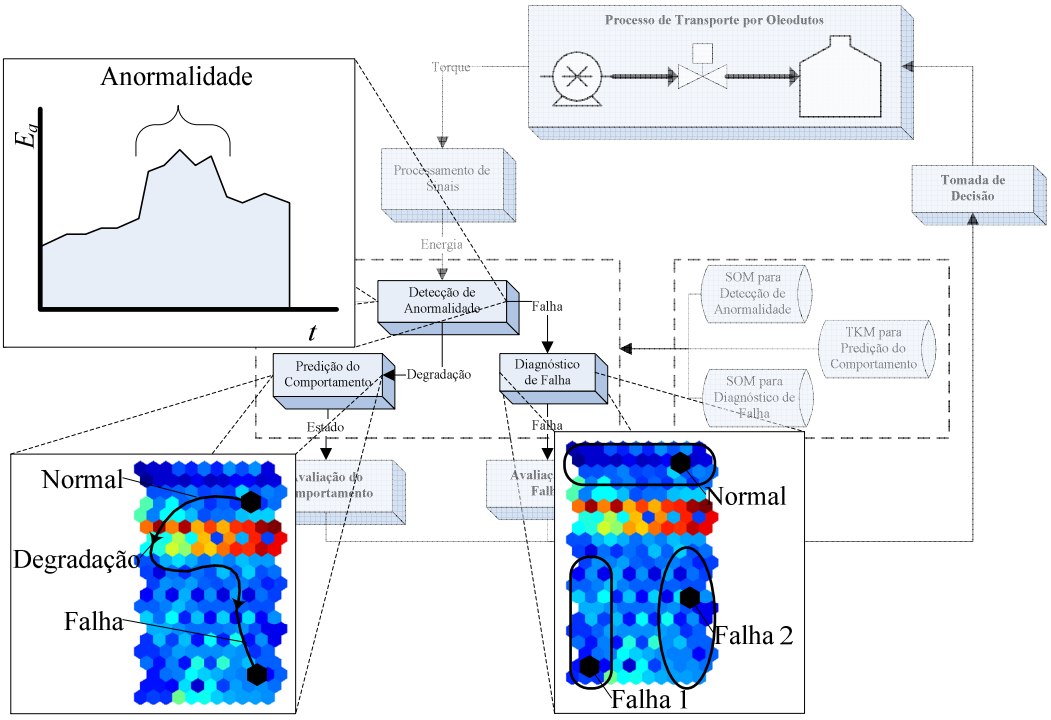



Figura 4.4: Partes internas que compões o Processo de Detecção, Diagnóstico e Predição de Falhas.

O Processo de Detecção, Diagnóstico e Predição de Falhas é composto por três algoritmos para extrair resultados das redes neurais, a fim de analisar o comportamento dos estados do sistema em monitoramento. Esses algoritmos são baseados nas equações vistas na Seção 3.5.1, em especial o algoritmo da *Distância Euclidiana,* que é o ponto-chave para todos os cálculos de processamento do SOM ou TKM. Cada um dos algoritmos apresentados desempenha uma função especial:

- **Detecção de Anormalidade**: utiliza o *SOM para Detecção de Anormalidade,* aplicando-se o algoritmo de teste (ver Seção 3.5.4.1). A detecção é feita pela avaliação dos erros de quantização para cada padrão de entrada. Quando o valor para o erro de quantização estiver se desviando da região de normalidade, então provavelmente estará se dirigindo a uma região de degradação ou falha. Isso significa que os dados de entrada não foram reconhecidos pelo SOM, porque ele apenas reconhece os dados normais. Essa função está baseada nos trabalhos relacionados, apresentados na Seção 3.7.1.

- **Diagnóstico de Falha**: utiliza o *SOM para Diagnóstico de Falha* e o modo de visualização *U-Matrix* (Seção 3.5.4.3). Durante o treinamento, são criados agrupamentos que identificam as classes de dados para as quais os padrões de comportamento foram mapeados. Pelo *índice* do neurônio vencedor é possível identificar a que classe o dado pertence. Essa será uma informação útil para classificar o padrão de entrada com base no conhecimento adquirido pelo SOM. Essa função faz uso do algoritmo de teste (Seção 3.5.4.1) e está baseada em conceitos dos trabalhos relacionados, conforme Seção 3.7.2.
- **Predição do Comportamento**: utiliza o *TKM para Predição do Comportamento* e o modo de visualização *U-Matrix* (Seção 3.5.4.3). Durante o treinamento, são criados os agrupamentos seguindo os padrões temporais das séries. Com isso, possibilita-se a identificação de uma tendência no comportamento dos padrões ao longo do tempo. Isso é feito pelo *índice* do neurônio vencedor, sendo possível acompanhar o seu movimento e identificar a formação da tendência para as séries de dados apresentadas na entrada. Essa será a informação utilizada para identificar a tendência no comportamento, baseando-se no conhecimento adquirido pelo TKM. Identificando com antecedência para onde está se direcionando o estado do sistema e, talvez, calcular um tempo até alcançar o estado "crítico". Essa função utiliza o algoritmo de recuperação do TKM visto na Seção 3.6.4.

### 4.2.1 Algoritmos base do SOM

As três etapas necessárias para implementar um SMI, utilizando o SOM e TKM como ferramenta de redes neurais, possibilitam a aquisição de conhecimento sobre o comportamento do sistema em monitoramento. Para implementar estes algoritmos, é necessário entender o funcionamento do SOM.

O SOM é baseado nas equações estudadas na Seção 3.5.1. Em especial, a principal base do algoritmo para treinamento e teste do SOM é o calculo da *Distância Euclidiana*. Esta medida de distância é o ponto-chave para o funcionamento da rede neural, pois faz parte de todas as etapas do algoritmo.

Nesta seção, será apresentado o algoritmo de teste do SOM e a aplicação deste para implementar as funções de tolerância a falhas ao SMI. Não será apresentado o algoritmo de treinamento, pois no SMI será utilizado o SOM já treinado por uma função externa.

A primeira parte do algoritmo é calcular a *Distância Euclidiana*, o que é ilustrado na Figura 4.5. Esse algoritmo é baseado na Equação 4 e será utilizado na etapa de teste do SOM, a equação será redefinida com base nos dados de entrada apresentado na figura, conforme a seguir:

$$D = \|x_k - w_k\| = \sqrt{(x_1 - w_1)^2 + (x_2 - w_2)^2 + \cdots + (x_k - w_k)^2}\ , \tag{13}$$

onde $\boldsymbol{x}$ e $\boldsymbol{w}$ são os dados de entrada do algoritmo e $k$ representa a dimensão do vetor de entrada.

A segunda parte do algoritmo consiste em implementar a etapa de *teste* ou *recuperação*, que tem por função utilizar a *Distância Euclidiana* para encontrar o neurônio vencedor e calcular o valor do erro de quantização, ver Figura 4.6. Esse algoritmo é baseado na Equação 3 e utilizado pelo SOM nas etapas de treinamento e de

teste, esta equação será redefinida com base nos dados de entrada apresentados na figura, conforme a seguir:

$$\begin{aligned} \text{ERRO}\,(\mathbf{x}) &= \arg\min_j \left\| \mathbf{x} - \mathbf{w}_j \right\| \\ \text{BMU}\,(\mathbf{x}) &= j \end{aligned}, \tag{14}$$

onde **x** é o vetor de entrada do SOM, **w** são os pesos sinápticos dos neurônios do SOM, e *j* representa o índice do neurônio na rede.

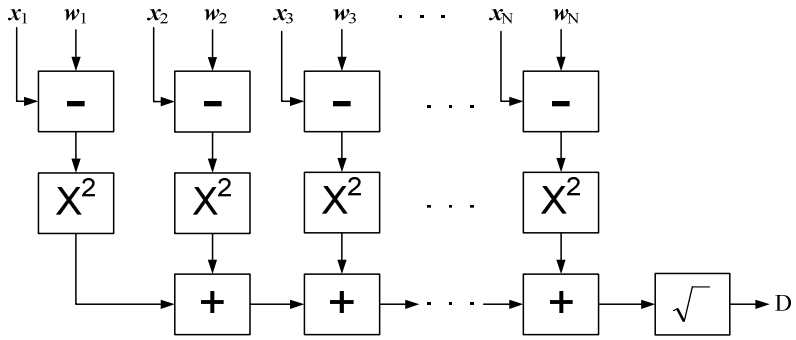



Figura 4.5: Algoritmo de cálculo da *Distância Euclidiana (D)*.

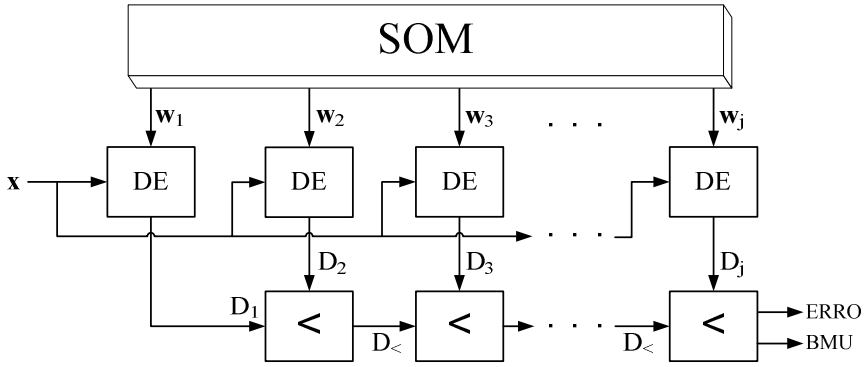



Figura 4.6: Algoritmo de recuperação (teste) do SOM

O algoritmo da Figura 4.6 realiza uma varredura em todos os neurônios do SOM para localizar aquele que apresenta a menor *Distância Euclidiana (D)* em relação ao vetor de entrada **x**. Na saída do algoritmo, são apresentados o erro de quantização (menor Distância Euclidiana) e o índice do neurônio vencedor.

Com esses dados disponíveis, é possível aplicá-los para implementar as etapas de tolerância a falhas, construindo um SMI. Nas seções posteriores, será detalhado o funcionamento de cada uma dessas etapas. Além disso, será mostrado como aplicar o algoritmo de recuperação no SOM, desde a apresentação do padrão de entrada até a saída da informação útil para o SMI.

### 4.2.2 Detecção de Anormalidade

A primeira etapa a ser implementada é a de Detecção de Anormalidade. Ela será detalhada, nesta parte do trabalho, com a apresentação de todo o funcionamento do algoritmo necessário para implementá-la.

Conforme visto anteriormente, essa etapa visa detectar qualquer anormalidade que ocorrer na entrada da rede, com base no conhecimento adquirido durante o treinamento.

Para essa detecção, é fundamental estabelecer um valor limite para detecção, de forma que tudo que exceder esse valor será considerado como anormal.

Seja **x**(*t*) o vetor de entrada de teste no tempo *t*, então o seguinte algoritmo é proposto:

```
Apresentar o vetor x(t) na entrada do SOM;
Enquanto x(t) for apresentado na entrada faça:
    Executar o algoritmo de teste conforme a Figura 4.6;
    Encontrar o neurônio vencedor j;
    Calcular o erro de quantização;
    Escrever na saída o erro de quantização (ERRO);
    Se ERRO >= Limite de Anormalidade então,
        Anormalidade detectada, sistema em condições anormais;
    Senão,
        Anormalidade não-detectada, Sistema em condições normais;
    Fim
Fim
```

Na Figura 4.7, é apresentada a visão geral do algoritmo de detecção, do ponto de vista da aplicação do SOM, como ferramenta para solucionar o problema.

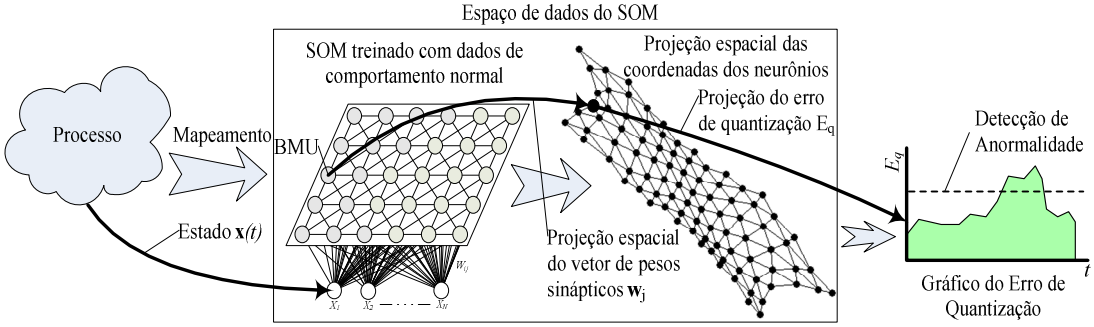



Figura 4.7: Visão geral do algoritmo de detecção usando o SOM.

O processo de detecção é executado do seguinte modo:

1. A etapa de treinamento realiza o mapeamento dos estados de operação do processo para o SOM.
2. Um sinal **x**(*t*) é extraído do processo e apresentado na entrada do SOM. O algoritmo de teste inicia a execução e realiza o mapeamento do padrão de entrada para um agrupamento no SOM.
3. O neurônio vencedor (BMU) é calculado e a saída é apresentada. Na Figura 4.7, dentro do *Espaço de dados do SOM*, são vistas duas representações, uma da arquitetura do SOM e outra a projeção espacial das coordenadas do peso sináptico dos neurônios. Nessas duas representações, é possível visualizar o neurônio vencedor calculado e sua localização no espaço de características.
4. É apresentado como saída do algoritmo de teste o valor do erro de quantização (ERRO ou $E_q$) para o vetor de entrada. Este é comparado com um valor predefinido de *Limite de Anormalidade* para julgar se o processo encontra-se *normal* ou *anormal*. Qualquer valor do erro que extrapolar o limite é considerado como *anormal*.
5. Se forem considerados todos os valores de erro de quantização ao longo do tempo, pode-se projetar um gráfico. Neste, pode-se visualizar o

comportamento das anormalidades e os exatos momentos em que foram detectadas pelo SMI.

6. Esse algoritmo tem funcionamento contínuo, em todo o tempo de operação do SMI. Como essa tarefa é a primeira etapa do SMI a ser executada, a partir dela, são acionadas as demais, diagnóstico e predição.

### 4.2.3 Diagnóstico de Falha

Após a detecção de anormalidade, é aplicada a etapa de Diagnóstico. Ela será detalhada, nesta parte do trabalho, com a apresentação do funcionamento do algoritmo necessário para implementá-la.

Como já visto, essa etapa visa diagnosticar as falhas ocorridas no processo. Caso um especialista, que conheça os comportamentos do processo, esteja disponível, pode-se, por seu intermédio, diagnosticar o que as falhas identificadas podem causar ao processo como um todo.

Seja **x**(*t*) o vetor de entrada de teste no tempo *t*, então o seguinte algoritmo é proposto:

```
Apresentar o vetor x(t) na entrada do SOM;
Enquanto x(t) for apresentado na entrada faça:
    Executar o algoritmo de teste conforme a Figura 4.6;
    Encontrar o neurônio vencedor j;
    Calcular o erro de quantização;
    Escrever na saída o índice do neurônio vencedor (BMU);
    Localizar a quais dos agrupamentos o BMU pertence;
    Escrever na saída o rótulo do agrupamento;
Fim
```

Na Figura 4.8, é apresentada a visão geral do algoritmo de diagnóstico, do ponto de vista da aplicação do SOM, como ferramenta para solucionar o problema.

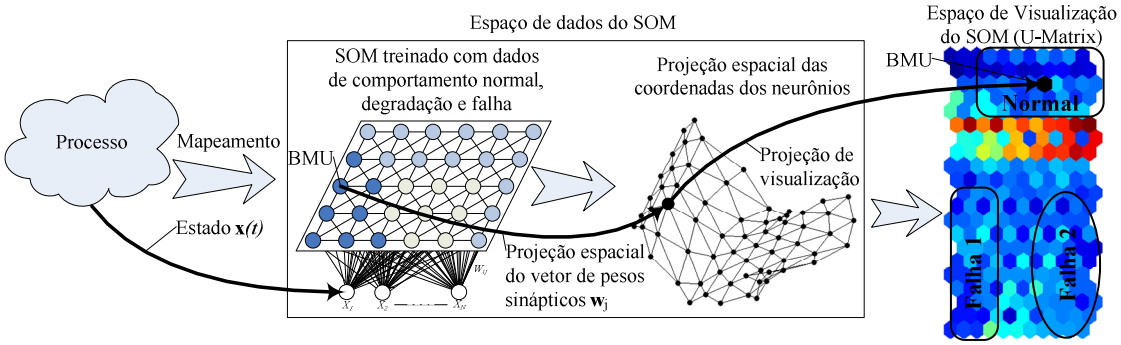



Figura 4.8: Visão geral do algoritmo de diagnóstico usando o SOM.

Todo o processo de diagnóstico é executado do seguinte modo:

1. A etapa de treinamento realiza o mapeamento dos estados de operação do processo para o SOM, formando os agrupamentos para os estados de cada componente monitorado.

2. Um sinal **x**(*t*) é extraído do processo e apresentado na entrada do SOM. O algoritmo de teste inicia a execução e realiza o mapeamento do padrão de entrada para um agrupamento no SOM.

3. O neurônio vencedor (BMU) é calculado e a saída é apresentada. Na Figura 4.8, dentro do *Espaço de dados do SOM*, são vistas as representações do

SOM, onde pode ser visualizada a arquitetura e a projeção espacial, revelando a formação topológica dos agrupamentos utilizados para classificação.

4. É apresentado, como saída do algoritmo de teste, o índice do neurônio vencedor (BMU) para o vetor de entrada. Esse valor é utilizado para localizar, dentro da lista de agrupamentos, em qual local o vetor de entrada foi mapeado.
5. Como saída do algoritmo de diagnóstico, é apresentado o rótulo do agrupamento (tipo do padrão que representa o agrupamento) que classificou o padrão de entrada. Os rótulos são dependentes dos padrões utilizados para treinamento, por exemplo, *falha X* ou *falha Y*, que para o caso do SMI, separam os tipos de falhas para cada componente do processo.
6. Após identificar o rótulo que classifica o padrão de entrada, o índice do neurônio vencedor é utilizado no algoritmo de visualização do SOM, seguindo o formato do *U-Matrix*. No lado direito da Figura 4.8, é apresentada a *U-Matrix*, destacado o neurônio para o qual o padrão de entrada foi mapeado e os agrupamentos, assim como os devidos rótulos.
7. Esse algoritmo tem seu funcionamento acionado pela etapa de Detecção. Essa tarefa pode executar em conjunto com a etapa de predição.

### 4.2.4 Predição do Comportamento

Após a detecção de anormalidade, é aplicada a etapa de Predição e Monitoramento da tendência de comportamento do SMI. Ela será detalhada, nesta parte do trabalho, com a apresentação do funcionamento do algoritmo necessário, como uma proposta de implementação. Essa etapa não será implementada neste trabalho, apenas serão feitas algumas simulações em *software* para analisar resultados.

Conforme já visto, essa etapa visa realizar a predição de estados futuros de operação do processo. Isso é feito pelo monitoramento contínuo dos estados de operação e pela identificação de uma tendência durante o tempo de operação do processo, baseando-se no conhecimento adquirido no treinamento.

Seja $X_N = \{\mathbf{x}(t), \mathbf{x}(t-1), \mathbf{x}(t-2), \cdots, \mathbf{x}(t-N)\}$ uma série de vetores de entrada para recuperação (teste) no intervalo de tempo *N*. Mesmo recebendo como entrada um conjunto formado por vários vetores em tempo diferentes, o TKM é capaz de processar apenas um padrão por instante de tempo, então a série de entrada deve ser decomposta em partes distintas para processamento. Seja o seguinte algoritmo:

```
Apresentar a série de dados X_n na entrada do TKM;
Enquanto X_n existir série de entrada faça:
    Para cada x(t) faça:
        Executar algoritmo de teste do TKM, ver Seção 3.6.2;
        Encontrar o neurônio vencedor j;
        Escrever na saída o índice do neurônio vencedor (BMU);
        Armazenar o BMU numa lista para o tempo t;
    Fim
Fim
```

Na Figura 4.9, é apresentada a visão geral do algoritmo de predição, do ponto de vista da aplicação do TKM, como ferramenta para solucionar o problema.

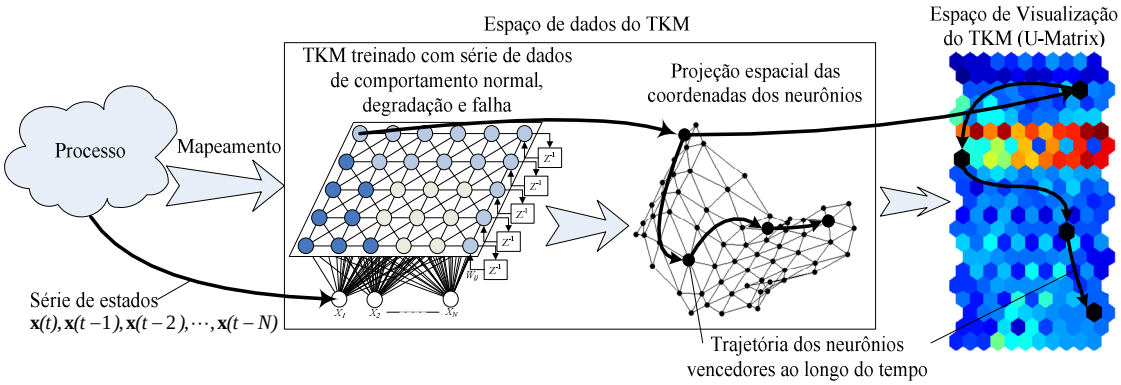



Figura 4.9: Visão geral do algoritmo de predição usando o TKM.

O processo de predição para identificar uma tendência no comportamento do processo deve ser executado do seguinte modo:

1. A etapa de treinamento realiza o mapeamento das séries de estados de operação do processo para o TKM.
2. Um sinal $\mathbf{x}(t), \mathbf{x}(t-1), \mathbf{x}(t-2), \cdots, \mathbf{x}(t-N)$ para certo período de tempo $N$ é extraído do processo em monitoramento e apresentado na entrada da rede. Este, por sua vez, extrai um vetor $\mathbf{x}(t)$ e apresenta na entrada do TKM. O algoritmo de teste inicia a execução e realiza o mapeamento do padrão de entrada para um agrupamento no TKM. É importante lembrar que o este algoritmo de teste é diferente do SOM. Para localizar o neurônio vencedor que represente toda a série de entrada, são executados novos testes para cada padrão da série.
3. O neurônio vencedor (BMU) é calculado e apresentado na saída. Na Figura 4.9, dentro do *Espaço de dados do TKM*, são vistas duas representações. A arquitetura do TKM é apresentada com suas memórias de curto prazo e na projeção espacial dos neurônios visualiza-se a trajetória percorrida pelos neurônios vencedores no decorrer do tempo.
4. Após a execução do algoritmo de teste do TKM, é apresentado na saída o índice do neurônio vencedor (BMU), possibilitando identificar em qual agrupamento a série de entrada foi classificada. Ao longo do tempo, os BMUs devem ser armazenados em uma lista, pois isso permite projetar a trajetória que revela a tendência de comportamento dos estados do sistema.
5. Com a projeção da trajetória e classificação dos rótulos dos agrupamentos pelo algoritmo de diagnóstico, é possível identificar uma tendência no comportamento dos estados de operação do processo. Por meio dessa tendência projetada, ao longo do tempo, pode-se, por exemplo, antecipar a ocorrência de um estado de falha para onde a projeção tende (agrupamento de *falha X*). Em alguns casos, pode-se, inclusive, calcular o tempo restante até atingir o agrupamento, caso seja conhecido o nível de degradação das partes monitoradas do processo.
6. Após, é aplicado o algoritmo de visualização do SOM no formato do *U-Matrix*, já que o TKM, depois de treinado, se comporta como o SOM. No lado direito da Figura 4.9, é apresentada a *U-Matrix* e são destacados os neurônios da lista de vencedores e é projetada entre eles a trajetória

percorrida, revelando a tendência de comportamento dos estados do processo.

7. Esse algoritmo tem seu funcionamento acionado pela etapa de Detecção. Essa tarefa pode ser executada em conjunto com a etapa de diagnóstico.

## 4.3 Projeto para Simulações do SOM em Software

Antes de iniciar o projeto do protótipo para a plataforma de *hardware*, é interessante realizar alguns experimentos em *software* que possam auxiliar na simulação de alguns casos de uso do SMI.

Essa ferramenta de simulação tem por objetivo realizar experimentos dos algoritmos apresentados na Seção 4.1, pelo uso do SOM como ferramenta de tolerância a falhas. As simulações são importantes para comprovar a eficácia do SOM em detectar e diagnosticar as falhas do processo em monitoramento, validando-se a técnica para realizar, a seguir, as demais etapas do projeto.

Embora o algoritmo padrão do SOM seja conceitualmente simples, a implementação requer uma série de cuidados fundamentais para garantir a convergência e generalização dos dados durante o treinamento. Devido a isso, o pesquisador Teuvo Kohonen[5], professor do *Laboratório de Ciência da Computação e Informação* na *Universidade Tecnológica de Helsinki*, é o idealizador e criador de uma implementação para experimentos no SOM (KOHONEN, 2001).

Kohonen afirma que a maioria das implementações não atende a todos os requisitos do processo de construção do algoritmo do SOM. Ciente do problema, a equipe de pesquisa desenvolveu dois pacotes de *softwares* com a implementação da rede neural, que estão disponíveis para a comunidade científica como código aberto.

Em primeiro lugar, foi projetado o SOM_PAK, desenvolvido em linguagem C e, após, foi lançada uma versão com mais recursos para o MATLAB, sendo conhecido por SOM Toolbox[6] (VESANTO, J. ET AL., 2000) (VESANTO, J. ET AL., 1999). Ambos são licenciados como *softwares* livres. Esses *softwares* são largamente difundidos entre os pesquisadores que trabalham com inteligência artificial, em especial redes neurais e mineração de dados.

Segundo (VESANTO, J. ET AL., 2000), o *software* do SOM deve apresentar um conjunto mínimo de requisitos importantes para realizar experimentos, tais como:

- Possibilidade de a rede (mapa ou arranjo de neurônios) ter dimensão dinâmica.
- Definição automática da dimensão da rede, com base em função dos autovalores da matriz de correlação dos vetores de entrada.
- Disposição dos neurônios no arranjo, em forma hexagonal e retangular.
- Algoritmo de aprendizagem-padrão (sequencial) ou em lote.
- Função de vizinhança topológica gaussiana ou bolha.

---

[5] Página web do professor Kohonen: http://www.cis.hut.fi/research/som-research/teuvo.html

[6] Página web do SOM Toolbox: http://www.cis.hut.fi/projects/somtoolbox/

- Inicialização linear ou aleatória.
- Algoritmos de visualização de resultados.
- Resultados de cálculos dos erros de quantização e topológico.

Vista a lista de requisitos, torna-se inviável reimplementar o SOM, a partir do zero, apenas para o trabalho em questão. Então, será utilizado, neste trabalho, o SOM Toolbox para MATLAB (versão 7). Reutilizar esse programa possibilita simulações em *software* do treinamento, testes de verificação e validação para os estudos de caso do SMI.

Inicialmente a ferramenta SOM Toolbox foi utilizada para aprender mais sobre o funcionamento do algoritmo do SOM e realizar diversos experimentos, a princípio apenas como questão didática, utilizando exemplos demonstrativos do *software*.

Após o desenvolvimento de um grau de aprendizado sobre o funcionamento do SOM, inicia-se a etapa de experimentos utilizando os dados coletados no processo monitorado pelo SMI. Essa parte do trabalho visa realizar um conjunto de experimentos a fim de aprender a utilizar o SOM Toolbox para processar os sinais coletados no processo.

Como o SOM contém diversos parâmetros de configuração, o algoritmo de treinamento é realizado de modo empírico. Foram experimentadas diversas configurações para buscar a que melhor se adaptava ao conjunto de dados utilizados para treinamento adquiridos do processo.

Para auxiliar na realização de todos os experimentos para o estudo de caso de um SMI para atuadores elétricos, foi desenvolvida uma ferramenta que utiliza o SOM Toolbox. Essa ferramenta tem como função principal a entrada de dados e saída de resultados extraídos do SOM para a implementação das etapas de detecção, diagnóstico e predição de falhas, conforme os algoritmos apresentados nas seções 4.1.2 a 4.1.4.

Na Figura 4.10, é apresentada uma imagem da tela da ferramenta de simulação utilizando o SOM Toolbox para o MATLAB. A ferramenta executa os algoritmos de treinamento, teste e visualização de resultados do SOM Toolbox.

Para exemplificar um algoritmo em MATLAB que utilize o SOM Toolbox, será apresentado o algoritmo para Detecção de Anormalidades conforme o código-fonte:

```
//ler conjunto de dados de entrada para treinamento
treinar = som_read_data('dados_normais.dat');
//executar algoritmo de treinamento do SOM e ajustar parâmetros
sMap = som_make(treinar,'training','long','size',[13 5]);
//ler conjunto de dados de entrada para teste
teste = som_read_data('dados_teste.dat');
//vetor para armazenar erros de quantização
qerr = [];
//calcular índices dos neurônios vencedores - BMU
bmus = som_bmus(sMap,teste.data);
for i=1:length(bmus),
    //calcular Distância Euclidiana entre BMU e Teste
    dx = sMap.codebook(bmus(i),:)-teste.data(i,:);
    qerr(i) = sqrt(sum(dx.^2));
end
//projeção do erro de quantização para os dados de teste
plot(qerr);
```

As demais etapas, diagnóstico de falhas (classificação) e carga de dados, também foram implementadas e integradas na ferramenta de simulação para o MATLAB.

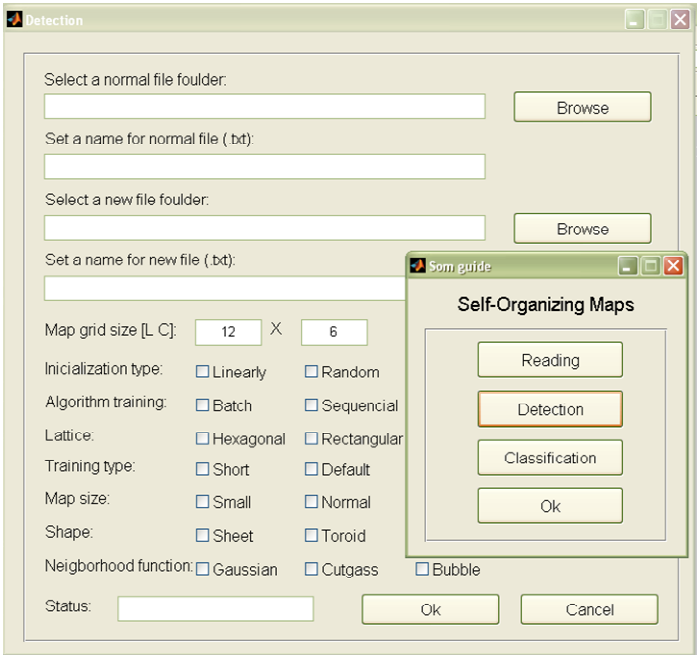



Figura 4.10: Tela da ferramenta de simulação em *software* para detecção e diagnóstico de falhas.

Para possibilitar o projeto do sistema embarcado em *hardware*, foi preciso desenvolver um *script* para converter dados do SOM treinado para um modelo em representação binária que o *hardware* possa interpretar.

Os resultados obtidos pela ferramenta são utilizados no projeto do protótipo do sistema embarcado em *hardware*, como será visto nas próximas seções.

## 4.4 Projeto do SOM em *Hardware*

Foi adotada, neste trabalho, a plataforma-alvo de FPGA (XILINX, 2008) para o desenvolvimento do componente de *hardware* que implementa as funcionalidades do algoritmo de teste do SOM. Assim foi decidido, devido à disponibilidade de um FPGA no laboratório de pesquisa e ao desenvolvimento de protótipo de um componente para ser integrado em um *chip* para ser implantado no futuro em cada atuador elétrico.

A metodologia de projeto do *hardware* do SOM adotada neste trabalho é baseada no fluxo de projeto apresentado na Figura 4.11 e em projeto de sistemas digitais (CARRO, 2001). A partir da especificação dos algoritmos de detecção, diagnóstico e predição, pode-se definir que todos eles fazem o uso de um mesmo algoritmo: o de teste ou recuperação do SOM.

Na seção 4.2.1, está especificado e detalhado o algoritmo de teste do SOM, que foi desenvolvido em *hardware*, utilizando uma descrição VHDL para FPGA da Xilinx. A partir do algoritmo, foi projetado um modelo em VHDL com parte operativa e de controle.

Após o desenvolvimento do modelo em VHDL, a próxima etapa é elaborar um conjunto de sinais para realizar o teste funcional no circuito, criando um outro projeto chamando de *Testbench*. Este gera estímulos as entrada do circuito para verificar o

correto funcionamento através dos sinais de saída. Então, a primeira simulação de teste realizada foi a baseada nos sinais comportamentais do projeto.

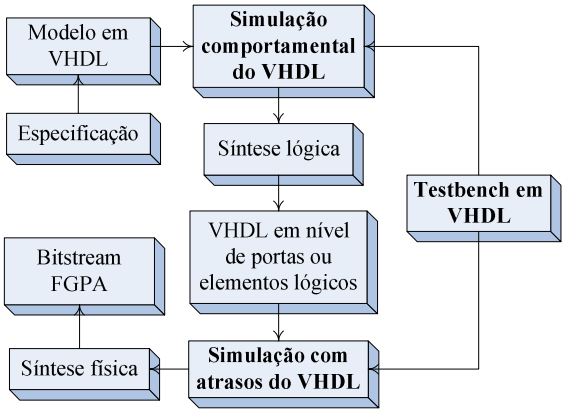



Figura 4.11: Fluxo de projeto para desenvolvimento de um sistema digital em FPGA.

Após a validação baseada na simulação comportamental, foi feita a síntese do modelo VHDL para a tecnologia FPGA, levando em consideração as restrições e requisitos técnicos da tecnologia selecionada. Como resultado, tem-se uma especificação do modelo de VHDL em nível de portas lógicas.

Outra etapa importante é a validação da descrição em nível de portas lógicas. Por meio da realização da simulação temporal (com atrasos), que leva em consideração os atrasos físicos presentes nos componentes elétricos (portas lógicas) do FPGA, é verificado o funcionamento do circuito em condições reais. Essa etapa é fundamental para validar o modelo VHDL, também aplicando teste funcional ao projeto agora em um ambiente mais realista.

Após a validação pela simulação temporal, pode ser realizada a síntese física para criar o *bitstream* para, por fim, ser possível configurar o FPGA e desempenhar a função descrita no modelo VHDL.

Na próxima seção será aprofundado o projeto do algoritmo de teste do SOM.

### 4.4.1 Arquitetura do *Hardware* do SOM

O modelo da descrição em VHDL para o algoritmo de teste do SOM é apresentado na Figura 4.12, revelando a relação entre os componentes de *hardware*, registradores e barramentos para transferências de dados.

Nessa estrutura, os vetores de peso ($\mathbf{w_i}$) dos neurônios, que compõem a arquitetura do SOM, estão armazenados na memória interna BRAM. O vetor de entrada é apresentado à rede neural pela porta de entrada ($\mathbf{x_j}$). Os demais componentes são o cálculo da Distância Euclidiana e um comparador utilizado para identificar o valor da menor distância (D).

O projeto foi baseado no conceito de implementação em série, em que registradores são utilizados para armazenar os estados intermediários do sistema. Foi definido implementar desse modo, devido aos vetores do SOM apresentarem dimensões (N). Assim, os vetores de entrada e os pesos dos neurônios devem ser sincronizados de modo a serem apresentados no mesmo instante de tempo.

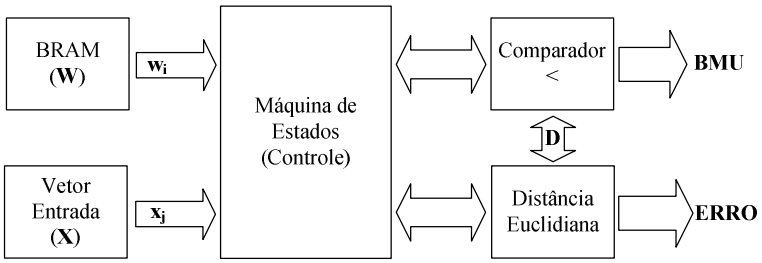



Figura 4.12: Visão geral da estrutura necessária para o projeto em *hardware* do algoritmo de teste do SOM.

A Máquina de Estados implementa o funcionamento do algoritmo de teste do SOM, em que são programados os estados para a transferência de dados entre os componentes que formam o sistema de *hardware*.

Na Figura 4.13, é apresentado o bloco de entradas e saídas para o projeto de *hardware* do algoritmo de teste do SOM. A largura de barramento dos sinais **x** e **ERRO** são de 32 bits devido a internamente utilizar a representação numérica de Ponto Flutuante IEEE 754. Os cálculos internos de comparação e Distância Euclidiana também utilizam componentes nesta representação numérica.

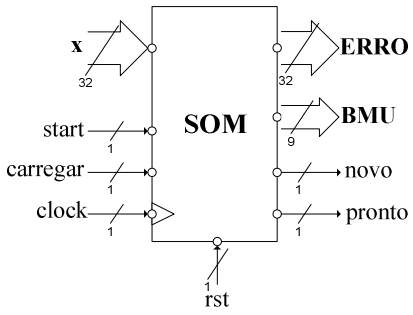



Figura 4.13: Entradas e saídas do componente de *hardware* do algoritmo de teste do SOM.

A entrada do sinal *start* é utilizada para inicializar o funcionamento do componente. A entrada *carregar* é utilizada para sincronizar os sinais do vetor de entrada, de acordo com a dimensão (N). Para cada componente do vetor apresentado esse sinal deve ser acionado. A saída *novo* é utilizada para informar a solicitação do próximo valor do vetor de entrada **x**. Esse sinal de saída é sincronizado com a entrada *carregar*. A saída *pronto* informa o final dos cálculos do SOM e apresenta nas saídas **ERRO** e **BMU** os valores calculados durante o funcionamento do bloco. A saída **BMU** é de nove bits, devido a uma definição interna no projeto de suportar um SOM com no máximo 512 neurônios.

#### *4.4.1.1 Projeto do componente para cálculo da Distância Euclidiana*

O primeiro componente fundamental para o algoritmo de teste do SOM é calcular a Distância Euclidiana entre dois vetores (conforme algoritmo detalhado na Seção 4.2.1).

A parte operativa do *hardware* pode ser vista na Figura 4.14. Neste projeto, foram reutilizados componentes que implementam operações numéricas (soma, subtração e extração de raiz quadrada) para o formato numérico de Ponto Flutuante IEEE 754, que possuem largura de barramento de 32 bits. Os componentes estão inclusos na biblioteca interna da ferramenta de projeto da Xilinx (XILINX, 2005a).

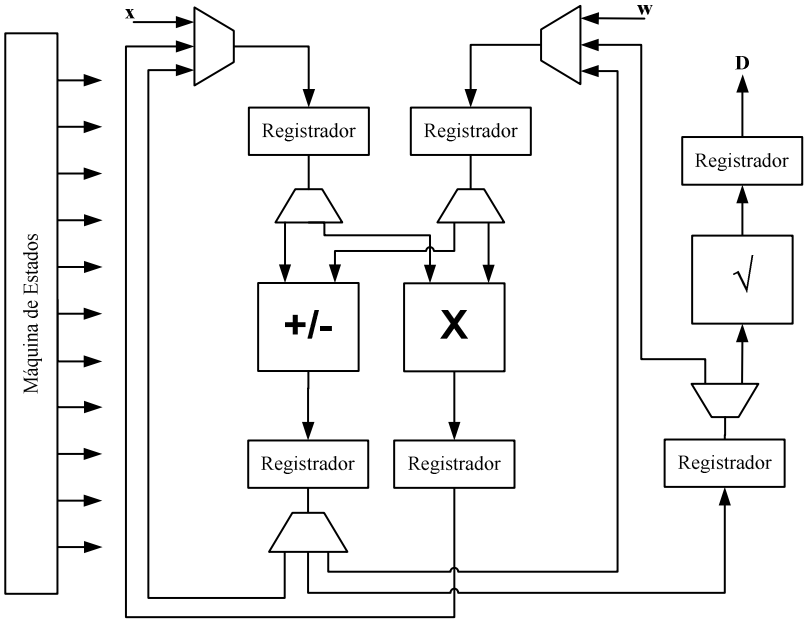



Figura 4.14: Parte operativa para o cálculo da Distância Euclidiana.

A parte de controle do *hardware*, onde é implementada a Máquina de Estados, gera os sinais de controle e o fluxo de dados entre os componentes e registradores, pode ser vista na Figura 4.15.

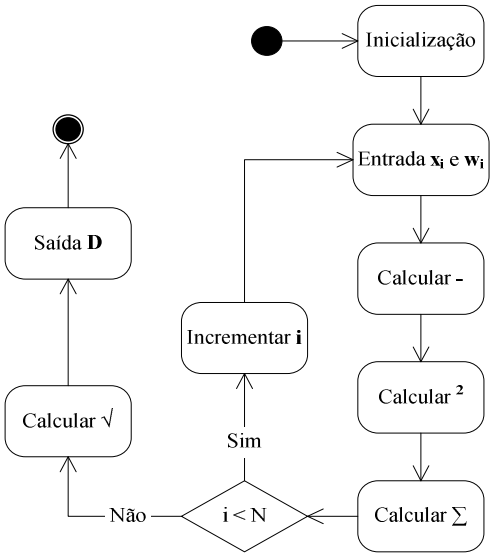



Figura 4.15: Máquina de Estados para o controle do cálculo da Distância Euclidiana.

O *hardware* para cálculo da Distância Euclidiana foi projetado com base na implementação em forma serial (SUDHA ET AL., 2003). Cada componente dos vetores **x** e **w** para um índice *i* é apresentado na entrada do circuito, seguindo uma sincronização até serem apresentados os N valores do vetor. Os sinais de controle aplicam uma série de cálculos numa sequência de subtração, multiplicação e acumulação em um somatório. Quando o índice *i* atingir a dimensão N, é feito o cálculo para extrair a raiz quadrada do valor acumulado no somatório. Por fim, é apresentado, na saída do *hardware*, o valor da distância euclidiana (**D**).

### *4.4.1.2 Projeto do algoritmo de teste do SOM*

O algoritmo de teste do SOM utiliza o componente da Distância Euclidiana como função básica nos cálculos. Desse modo, a segunda parte do projeto do *hardware* é integrar o componente aos demais para implementar o algoritmo de teste conforme visto na Seção 4.2.1. Essa parte do projeto diz respeito ao bloco apresentado na Figura 4.13.

Na Figura 4.16, é apresentada a parte operativa do algoritmo de teste do SOM. A entrada principal de dados é a apresentação dos vetores $\mathbf{x_i}$, para $i=0,1,2, ..., N$, sendo $N$ a dimensão dos vetores de entrada.

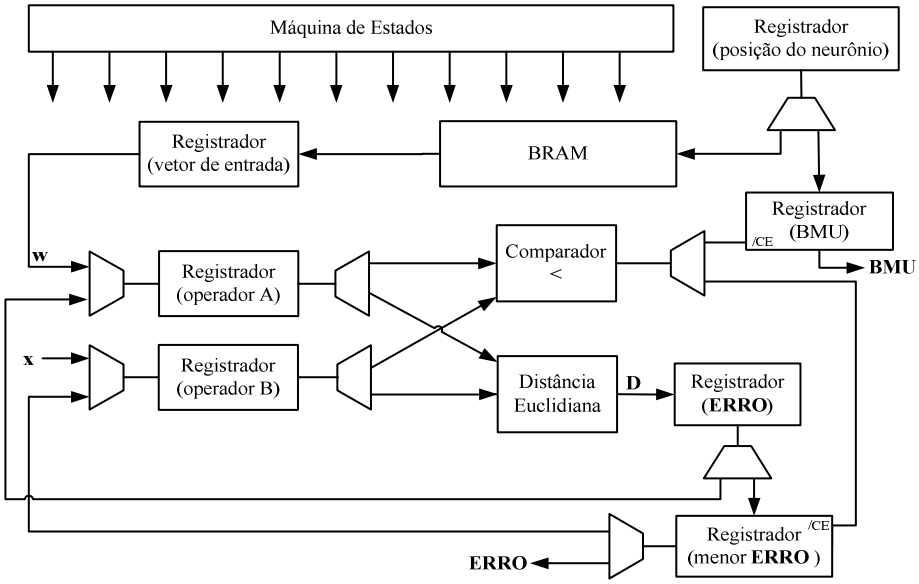



Figura 4.16: Parte operativa do algoritmo de teste do SOM.

A função básica desse bloco é identificar o neurônio do SOM que apresenta a menor Distância Euclidiana em relação ao vetor de entrada. Portanto, foi necessário utilizar um componente que implementa a operação de comparação para o formato numérico de Ponto Flutuante IEEE 754.

Na parte operativa, é visto um componente chamado de BRAM. Esse componente é uma memória interna do FPGA, utilizada para armazenar os pesos sinápticos de todos os neurônios que formam o SOM já treinado em *software*. Os neurônios são alocados na memória de forma linear e contínua conforme visto na Figura 4.17.

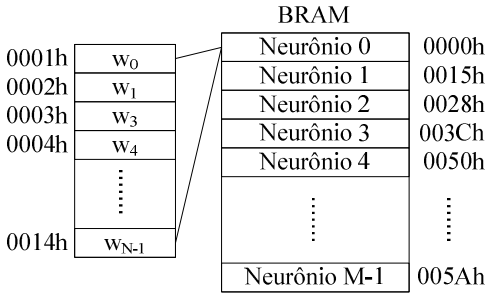



Figura 4.17: Armazenamento dos pesos sinápticos de cada neurônio na BRAM.

Cada célula da memória armazena um componente $i$ do vetor de peso sináptico $\boldsymbol{w}_{i+j}$,com dados de 32 bits, sendo $N$ a dimensão dos vetores de peso sináptico e M a quantidade de neurônios do SOM acessado pelo índice $j$.

Cada componente do vetor de peso dos neurônios pode ser acessado de forma direta, pelo endereço da memória, em que cada componente $w_{i+j}$ diz respeito a um valor da componente peso sináptico do neurônio *j* na posição *i*.

Também são utilizados dois registradores importantes para o funcionamento do algoritmo na parte operativa. Um registrador importante é o que armazena o menor valor para o erro de quantização (*menor* ***ERRO***) e outro é utilizado para armazenar o índice *j* deste neurônio (*BMU*). Ambos têm seus valores atualizados pelo componente comparador que autoriza a escrita neles.

A parte de controle do *hardware* é implementada por uma máquina de estados que gera os sinais de controle internos, fluxo de dados entre os registradores e componente da Distância Euclidiana, conforme mostra a Figura 4.18.

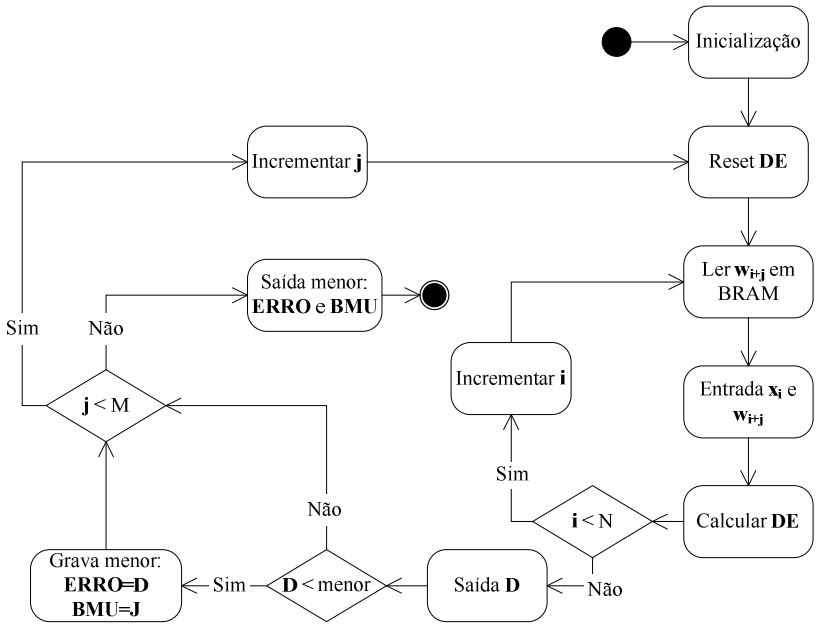



Figura 4.18: Máquina de Estados para o algoritmo de teste do SOM.

O *hardware* do algoritmo de teste do SOM foi projetado com base na implementação serial. Cada componente do vetor de entrada **x** para um índice *i* é apresentado na entrada do circuito, e ao mesmo tempo também é apresentado o vetor de peso sináptico do neurônio *j* obtido pela BRAM. Ambos são sincronizados.

Para cada vetor de entrada **x** é executado M vezes o cálculo da Distância Euclidiana e sempre, ao término de cada cálculo, é verificado se o novo valor do erro é menor do que aquele armazenado no registrador. Caso seja, o registrador é atualizado. Isso acontece porque o algoritmo precisa localizar qual é o menor valor entre todos os neurônios que constituem o SOM.

Por fim, após a execução de todos os estados, a Distância Euclidiana é calculada para todos os neurônios do SOM e localiza o neurônio que apresenta o menor erro de quantização. Como saída do circuito, serão apresentados o valor desse erro (**ERRO**) e o índice do neurônio (**BMU**).

### 4.4.2 Resultados de Projeto do *Hardware* do SOM

O projeto de *hardware* do SOM foi realizado utilizando um conjunto de ferramentas da Xilinx para FPGA, o *ISE* para projeto e síntese (XILINX, 2005a), o *ModelSIM*[7] para simulações comportamentais e temporais e o *XPower* para simulação de consumo de potência (incluso no *ISE*).

Os resultados apresentados, nesta seção foram adquiridos adotando a plataforma FPGA da Xilinx, modelo XUP Virtex 2 PRO utilizada para síntese da descrição VHDL (XILINX, 2008) (XILINX, 2005b).

Primeiro, foram definidos os parâmetros para especificação de um SOM conforme a Tabela 4.1. Adotou-se um mapa com dimensão de 15 linhas por 6 colunas, com um total de 90 neurônios (valor adequado para os experimentos), em que o vetor para cada neurônio é composto por 20 elementos, sendo que cada elemento armazena um valor numérico de 32 bits, ocupando um espaço total de 56,35 KBytes para armazenamento na memória BRAM. Esse SOM foi treinado e os vetores de pesos sinápticos dos neurônios foram salvos em separado para serem gravados na memória BRAM.

Tabela 4.1: Especificação do SOM treinado.

| **Parâmetro** | **Valor** |
|---|---|
| Dimensão do mapa | 15 linhas X 6 colunas |
| Número de neurônios | 90 Neurônios |
| Dimensão do vetor | 20 elementos |
| Cada elemento do vetor | 32 bits |
| Uso de memória | 56,35 Kbytes |

Os vetores salvos foram processados por um *script* para transformá-los em representação binária e serem carregados na BRAM interna do circuito. No momento da síntese do circuito, os valores binários dos pesos dos neurônios são armazenados na BRAM e estão prontos para execução do algoritmo de recuperação.

Utilizando a ferramenta ISE como base para o projeto, foi adicionado o conjunto de arquivos que descrevem o *hardware* do SOM. A descrição reutilizou alguns componentes da biblioteca interna do FPGA para implementar as operações numéricas de Ponto Flutuante 754, soma/subtração, multiplicação, raiz quadrada e comparação.

Na Tabela 4.2, são apresentados os resultados de área obtidos após a síntese do circuito. Esses resultados são referentes ao espaço físico ocupado dentro do FPGA, revelando a quantidade e a proporção de componentes utilizados pelo projeto.

Tabela 4.2: Resultados de área do circuito obtidos na síntese.

| **Parâmetro** | **Utilizados** | **Disponível** | **Utilização** |
|---|---|---|---|
| Flip-flops | 2597 | 27392 | 9% |
| Slices | 2158 | 13696 | 15% |
| LUTS | 2664 | 27392 | 9% |
| IOBS | 79 | 556 | 14% |
| BRAM | 29 | 136 | 21% |

[7] ModelSim Xilinx Edition-III Details: http://www.xilinx.com/ise/verification/mxe_details.html

Na Tabela 4.3, são apresentados os resultados de desempenho obtidos após a síntese do circuito. Esses resultados revelam a frequência de *clock* máximo, que pode ser aplicado no circuito, e as características de tempo dos registradores.

Tabela 4.3: Resultados de desempenho do circuito obtidos na síntese.

| **Parâmetro** | **Valor** |
|---|---|
| Frequência de operação | 152 MHz |
| Tempo mínimo de chegada da entrada antes do clock (setup time) | 2.520 ns |
| Tempo máximo de saída necessário depois do clock (hold time) | 3.636 ns |

Após a síntese do circuito e a validação pela simulação comportamental, são executadas as etapas de mapeamento e roteamento do circuito. Depois de finalizada essa parte, é feita a simulação temporal, que visa avaliar se o circuito está em condições de gerar o *bitstream*, pois leva em consideração os atrasos físicos de cada componente do circuito em nível elétrico.

Na Figura 4.19, é apresentado o resultado em formas de onda da simulação temporal, comprovando o correto funcionamento do projeto. Vale destacar que o tempo de execução para calcular o **BMU** e o **ERRO** para apenas um vetor de entrada é de aproximadamente de *864560 ns*.

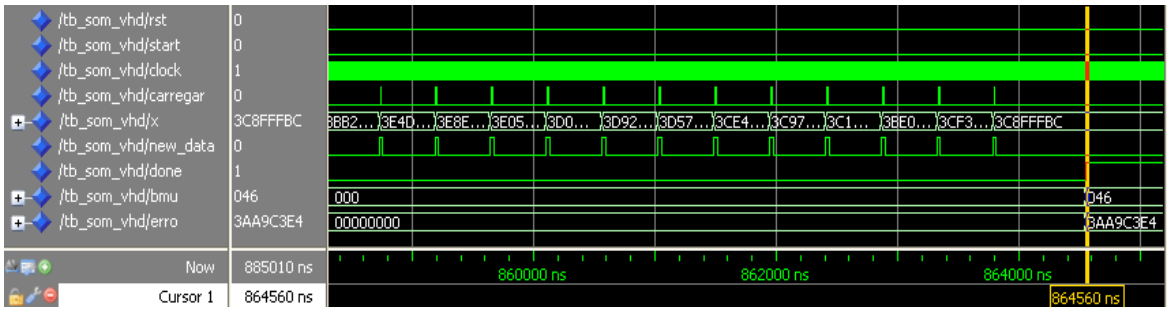



Figura 4.19: Simulação temporal do circuito do SOM e tempo de execução.

Na Tabela 4.4, são apresentados os resultados de potência obtidos após a síntese do circuito por meio da ferramenta *XPower*. Esses resultados revelam o consumo de energia do circuito. Vale destacar que foi utilizada uma memória BRAM, nesta simulação, com capacidade de armazenamento de 512 Kbytes, pois o projeto do SOM foi definido para suportar no máximo 512 neurônios de no máximo 32 elementos por vetor.

Tabela 4.4: Resultados de potência do circuito do SOM após mapeamento.

| **Parâmetro** | **Valor** |
|---|---|
| Potência estática | 0.02813 W |
| Potência dinâmica | 0.06805 W |
| Potência total | 0.09618 W |

Com a aplicação de todas as simulações e a validação pelo *testbench* para os casos de teste comportamental e temporal, foi validado o funcionamento do projeto do circuito do SOM, baseando-se em teste funcional. A partir de agora, pode-se gerar o *bitstream* para programar o FPGA.

O circuito do SOM ainda não está em condições de ser usado como protótipo, antes é preciso definir um sistema embarcado que seja capaz de utilizar esse circuito e extrair as informações necessárias para o Sistema de Manutenção Inteligente em Atuadores Elétricos. Na próxima seção, será estudado o projeto do Sistema Embarcado.

## 4.5 Sistema Embarcado para Manutenção Inteligente

Segundo (CARRO; WAGNER, 2003) um sistema embarcado é definido como uma combinação de componentes de *hardware* e *software*, projetados para desempenhar uma determinada tarefa. Normalmente, é formado por microprocessador, memória e periféricos para executar uma determinada aplicação. A diferença entre um sistema embarcado E um computador de propósito geral (*desktop*) está nos requisitos e aplicações alvo para as quais é projetado.

Pode-se dizer que os sistemas embarcados estão inseridos em várias aplicações do cotidiano das pessoas, por exemplo, telefones celulares, com câmera fotográfica digital, sistemas de caixa em supermercado, quiosques bancários, eletrodomésticos, sistemas de freios ABS e muito mais.

Um sistema embarcado consiste na combinação entre *hardware* e *software* projetados para desempenhar uma determinada aplicação. Os processadores embarcados podem ser de tipos diversos dependendo da aplicação. O *software* da aplicação pode ser composto por múltiplos processos.

Devido ao grande espaço de projeto em sistemas embarcados, a diversidade de soluções para a implementação de uma determinada aplicação torna o processo muito complexo. Por isso, o reuso de plataformas de *hardware* e *software* pode permitir uma redução no espaço de soluções e, portanto, no tempo de projeto.

Uma plataforma é uma arquitetura de *hardware* e *software* específica para determinado domínio de aplicação. É altamente parametrizável quanto ao número de componentes, na estrutura de comunicação, no tamanho da memória, nos dispositivos de entrada e saída, etc. O reuso dos componentes da plataforma pode ser ainda reforçado pela adoção de padrões na arquitetura e projeto do sistema.

Do ponto de vista do *hardware*, de um sistema embarcado é formado por um mínimo de componentes (CARRO; WAGNER, 2003), basicamente por:

- Microprocessador: é a unidade de processamento (CPU) e geralmente projetada para um domínio de uso específico (baixo consumo de área, potência e memória).
- Memórias: são as unidades de armazenamento de *softwares* (RAM, FLASH, etc.).
- Barramento de comunicação: é responsável pelo transporte dos sinais e dados entre os componentes que formam o sistema.
- Periféricos: são todos os componentes externos integrados ao sistema, que desempenham uma função especifica, por exemplo, porta serial, *ethernet*, conversor D/A, interfaces de comunicação, etc.

Do ponto de vista do *software*, este é geralmente armazenado em uma memória que compõe a plataforma. O *software* é compilado para a plataforma-alvo e está diretamente ligado à arquitetura do microprocessador adotado.

O projeto do *software* embarcado é influenciado por algumas características importantes que o diferenciam dos *softwares* tradicionais:

- Limite na memória disponível.
- Limite no desempenho do processador (pode exigir maior paralelismo).
- Limite no consumo de potência.

Devido à complexidade do projeto de *software* embarcado surgiu o conceito de utilização de Sistema Operacional Embarcado (SOE) para controlar e desempenhar todas as funções no dispositivo. Um SOE é classificado como um *software* embarcado específico que tem alta interação com o *hardware*. É composto por um conjunto de implementações em diferentes níveis de abstração, desde a linguagem de máquina até a de mais alto nível.

Um SOE dedicado a aplicações de sistemas embarcados deve estar sujeito aos requisitos naturais exigidos por esses sistemas. As principais restrições no projeto são: limite de memória disponível, desempenho e consumo de potência.

As principais características de um SOE serão apresentadas nas próximas seções. A seguir, será apresentada a construção do sistema embarcado, revelando as ferramentas e etapas necessárias para o projeto de *hardware* e *software*.

### 4.5.1 *Hardware* do Sistema Embarcado

Conforme (XILINX, 2008), FPGAs são *chips* que suportam a implementação de circuitos lógicos de *hardware*. O FPGA se comporta como um *hardware* programável. O dispositivo consiste de um arranjo de células lógicas configuráveis associadas a uma infraestrutura de interconexões, em que cada célula pode ser programada. O FPGA pode ser utilizado para a implementação de funções lógicas definidas pelo usuário.

A arquitetura interna de um FPGA é constituída por:

- Blocos lógicos programáveis (CLBs): implementam toda a lógica combinacional e sequencial de um circuito. São constituídos por flip-flops e LUTs.
- Blocos de entrada e saída (IOBs): são responsáveis pela interface dos CLBs com os pinos de entrada e saída do encapsulamento do FPGA e com o barramento de interconexões. São formados por *buffers*, que funcionarão como um canal bidirecional de entrada/saída.
- Interconexões programáveis: são trilhas ou barramentos para conectar as entradas e saídas dos CLBs e IOBs. O processo de escolha das interconexões é feito durante o roteamento. Cada fabricante de FPGA determina o número disponível de interconexões.
- Bloco de memória (BRAM): em alguns modelos de FPGA, está disponível um banco de memória para implementação de funções lógicas mais complexas ou apenas para armazenar dados.

Atualmente, existe no mercado uma grande variedade de modelos de FPGA de diversos fabricantes. Alguns exemplos de fabricantes importantes são Altera, Actel e Xilinx, entre outros. Estes oferecem dispositivos com capacidades de programação distintas, conjuntos de características específicas, tais como desempenho, consumo de

energia, testabilidade, diferentes métodos de programação, diferentes arranjos de interconexões e funcionalidades básicas distintas dos CLB.

No presente trabalho, foi adotado um modelo de FPGA que tem a programação das células baseada na tecnologia de SRAM. O *bitstream* (dados de configuração ou programa do FPGA) é carregado na SRAM, que configura os blocos lógicos do circuito. Devido à volatilidade da memória, o FPGA precisa de uma memória externa não-volátil para recarregar a configuração em cada momento que é energizado.

O FPGA pode ser utilizado para implementar praticamente qualquer projeto de *hardware*. Um dos usos mais comuns é a utilização do FPGA para a prototipação de circuitos digitais, que posteriormente pode ser implementado em um *chip* específico. No entanto, o FPGA está sendo cada vez mais inserido como produto final no mercado devido ao seu baixo custo de inserção no mercado.

Foi adotado, neste trabalho, utilizar o FPGA do modelo Virtex-2 PRO, fabricado pela empresa Xilinx. Este foi acoplado a uma plataforma de desenvolvimento em uma placa fabricada pela Digilent (*Virtex 2 Pro Development System*)[8] (XILINX, 2005b). Essa placa é um *hardware* utilizado para experimentos e desenvolvimento de projetos de sistemas embarcados.

A placa apresenta uma arquitetura propícia para projetos experimentais de sistemas embarcados, pois é totalmente configurável por meio do FPGA, utiliza periféricos bastante difundidos no mercado e padronizados. Esta contém os seguintes periféricos disponíveis: uma porta 10/100 Ethernet, porta USB, porta serial RS-232, porta de saída de vídeo VGA, um leitor para cartão de memória, interface para disco rígido, conversores A/D e D/A, interface para memória RAM DDR, entre outros. É um sistema computacional completo, explorando várias partes da área de arquitetura de computadores. Uma foto da placa pode ser vista na Figura 4.20.

O *chip* do FPGA incluído na placa é o modelo Virtex-2 PRO (XC2VP30) (XILINX, 2008) com 30816 blocos lógicos programáveis (CLB), 136 multiplicadores de 18-bits, 2448 Kb de memória BRAM e dois processadores PowerPC integrados.

Desse modo, tem-se definido para este trabalho a plataforma de *hardware* que será utilizada para a prototipação do sistema embarcado para a manutenção inteligente.

Na Figura 4.21, é apresentada a arquitetura do sistema embarcado para prototipar o sistema de manutenção inteligente.

O sistema embarcado será composto por um barramento de comunicação entre todos os componentes de *hardware*, um processador MicroBlaze (XILINX, 2003), memória RAM para armazenar dados, interface de comunicação com usuário (porta serial) e interface de E/S (para acionar atuadores ou ler sensores). Essa arquitetura foi programada no FPGA da placa de desenvolvimento, para configurar e formar o sistema embarcado necessário para os experimentos.

O projeto do sistema embarcado foi realizado por meio da ferramenta XPS (*Xilinx Platform Studio*) (XILINX, 2005c) desenvolvida pela Xilinx. Essa ferramenta é específica para o projeto de sistemas embarcados utilizando plataformas com FPGA. Possibilita, de forma automatizada, a adição ou remoção de componentes computacionais ao sistema, entre eles, processadores, barramentos, memórias,

[8] Digilent Inc. Products: http://www.digilentinc.com/Products/Detail.cfm?Prod=XUPV2P

periféricos, etc. Ou seja, é uma ferramenta para o desenvolvimento rápido de sistemas embarcados que usa a tecnologia de FPGA, com *softwares* de simulação e teste, e uma vasta biblioteca de componentes de *hardware*

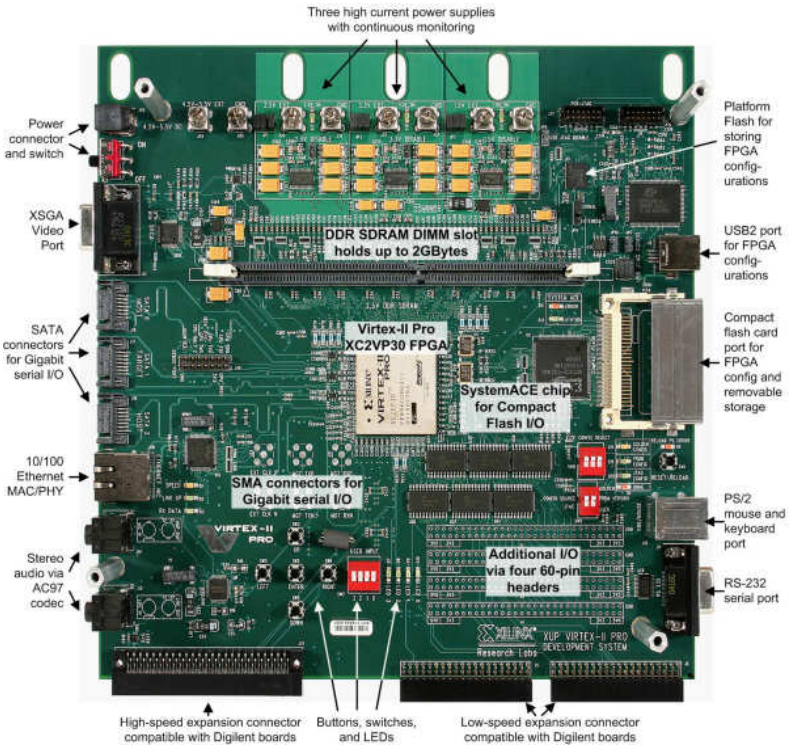



Figura 4.20: Placa de desenvolvimento Digilent Virtex-2 PRO.

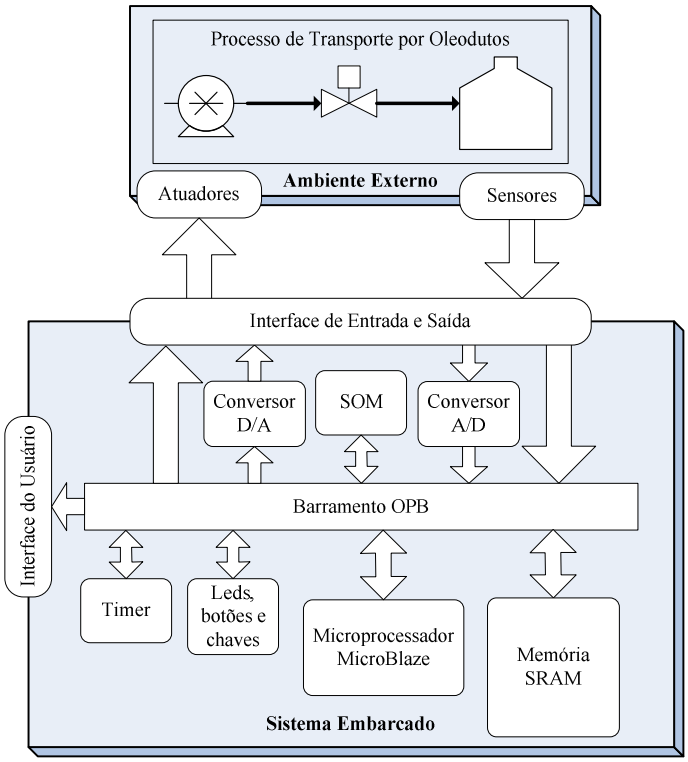



Figura 4.21: Arquitetura do sistema embarcado para projetar o Sistema de Manutenção Inteligente.

Já que será usado o SOE, optou-se por utilizar o microprocessador MicroBlaze como padrão. O SOE adotado tem todos os *drivers* necessários aos dispositivos presentes na plataforma já validados e será apresentado na próxima seção.

Desse modo, foi projetado um sistema embarcado (JESMAN ET AL., 2006) centrado no uso do microprocessador MicroBlaze (XILINX, 2005b), acoplando os periféricos da placa a um barramento de comunicação de dados com base no padrão OPB (*On-chip Peripheral Bus*) (XILINX, 2002) (XILINX, 2005d). Os seguintes componentes de *hardware* foram utilizados para o projeto da arquitetura do sistema embarcado:

- Microprocessador MicroBlaze de 100MHz, com memória cache de 8Kb.
- Módulo para debug, verificação e gravação de dados na memória.
- Porta RS232 para comunicação serial e para interface com o usuário, permitindo comunicação com PC.
- Módulos de periféricos: leds, botões e chaves.
- Módulo de memória SRAM com 512MB para programa e dados.
- Timer de 32bits para o escalonamento do SOE.
- Barramento de dados para periféricos OPB de 100MHz.
- Módulo de controle de interrupção.
- Módulo do *hardware* do SOM acoplado ao barramento OPB.
- Entre outros módulos extras.

Os periféricos conversores A/D e D/A estão disponíveis na placa de desenvolvimento, mas não foram utilizados nos experimentos no protótipo do sistema embarcado.

Na Tabela 4.5, são apresentados os resultados de área total obtidos por meio da síntese do protótipo do sistema embarcado que foi projetado com base na Figura 4.21. Esses foram extraídos da ferramenta XPS (XILINX, 2005c). Na parte inferior da tabela, são apresentadas as taxas de utilização do FGPA na plataforma. Em média foram utilizados 28% da área total.

Nota-se que, nestes resultados, foram adicionados todos os circuitos extras necessários para construção do sistema embarcado. Na coluna parâmetro, são apresentada as áreas ocupadas por cada um dos *cores* utilizados.

Tabela 4.5: Resultados de área total ocupada pelo sistema embarcado no FPGA.

| **Parâmetro** | **Flip-flops** | **Slices** | **LUTS** | **IOBS** | **BRAMS** |
|---|---|---|---|---|---|
| system | - | - | - | 134 | - |
| microblaze | 988 | 1344 | 2029 | - | 12 |
| mb_opb | 11 | 168 | 288 | - | - |
| debug_module | 118 | 72 | 45 | - | - |
| ilmb | 1 | 1 | 1 | - | - |
| dlmb | 1 | 1 | 1 | - | - |
| dlmb_cntlr | 1 | 3 | 5 | - | - |
| ilmb_cntlr | 1 | 3 | 5 | - | - |

| | | | | | |
|---|---|---|---|---|---|
| lmb_bram | - | - | - | - | 4 |
| rs232_uart | 60 | 51 | 92 | - | - |
| leds_4bit | 50 | 37 | 29 | - | - |
| dipsws_4bit | 33 | 26 | 23 | - | - |
| pushbottons_5bit | 36 | 29 | 23 | - | - |
| ddr_512mb | 671 | 795 | 738 | - | - |
| opb_timer | 245 | 224 | 196 | - | - |
| opb_intc | 133 | 90 | 72 | - | - |
| fsl_v20 | 7 | 22 | 44 | - | - |
| som_core | 3406 | 2858 | 2775 | - | 26 |
| **Total** | **5762** | **5724** | **6366** | **134** | **42** |
| **Disponível** | **27392** | **13696** | **27392** | **556** | **136** |
| **Utilização** | **21,04%** | **41,80%** | **23,25%** | **24%** | **30,90%** |

O módulo do *hardware* do SOM desenvolvido neste projeto foi acoplado ao barramento OPB e será chamado *som_core*. Fez-se necessário criar um *wrapper* para integrar o novo componente aos sinais e padrões adotados pelo barramento. Essa integração visa possibilitar o acesso ao dispositivo pelas instruções de programa que serão executadas pelo microprocessador.

O *wrapper* foi desenvolvido utilizando a ferramenta *Peripheral Wizard* do XPS (XILINX, 2002) (JESMAN ET AL., 2006), que visa auxiliar na construção de "esqueletos" para integração com padrões de outros componentes. Durante a construção do *wrapper*, adotou-se a vinculação das entradas e saídas do *som_core* em registradores mapeados em memória. Assim, as escritas e leituras nos registradores servem para controlar os comandos de operações do *som_core* e podem ser acessadas pelo *software* através de um endereço na memória.

Na Figura 4.22, é apresentada a parte operativa do *wrapper* do *som_core*, que integra o *hardware* do SOM ao barramento OPB. Na figura, são apresentados os registradores de dados, canais de comunicação, entradas e saídas do dispositivo.

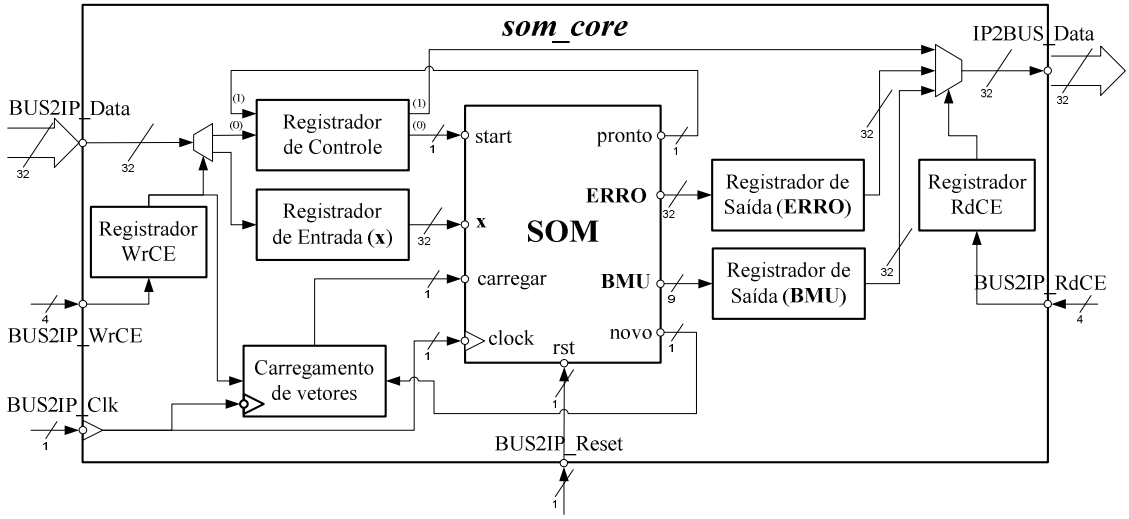



Figura 4.22: Parte operativa do *wrapper* para integração do *som_core* ao barramento OPB do sistema embarcado.

Foram utilizados quatro registradores de 32 bits mapeados em memória para controlar o dispositivo:

1. Registrador de Controle: registrador de leitura e escrita. Controla o sinal de START do SOM (para inicializar o funcionamento) e lê o sinal de PRONTO do SOM (quando o cálculo estiver concluído). Utiliza apenas dois bits.
2. Registrador de Entrada (**x**): registrador somente de escrita. Escreve o sinal de entrada, armazenando os dados apresentados no barramento OPB no registrador e apresenta o sinal na entrada **x** do SOM. Utiliza 32bits.
3. Registrador de Saída (**BMU**): registrador somente de leitura. Lê os dados coletados na saída **BMU** do SOM. Utiliza 32bits.
4. Registrador de Saída (**ERRO**): registrador somente de leitura. Lê os dados coletados na saída **ERRO** do SOM. Utiliza 32bits.

Os sinais de RESET e CLOCK do SOM são vinculados aos demais sinais de controle do barramento OPB.

As escritas e leituras nos registradores que visam controlar o SOM são realizadas a partir do barramento OPB. São utilizados os canais de comunicação BUS2IP_Data (entradas) e IP2BUS_ Data (saídas) para ler e escrever dados nos registradores de 32bits. O canal de seleção dos registradores BUS2IP_WrCE (escrita) e BUS2IP_RdCE (leitura) é usado para selecionar qual registrador será manipulado.

Na Tabela 4.6, são apresentados os resultados de comparação da área ocupada entre o *hardware* do SOM (Tabela 4.2) e o *som_core*. Os resultados foram obtidos pela síntese do protótipo do sistema embarcado completo, com base na Figura 4.21. Ocorreu um aumento na área ocupada (*overhead*) pelo *som_core* em relação ao SOM. Isso é devido à adição de componentes extras ao *wrapper* para integrar o componente ao barramento OPB, possibilitando a interação com os demais componentes do sistema embarcado.

Tabela 4.6: Resultados de área do *hardware* do SOM obtidos na síntese.

| **Parâmetro** | **SOM** | **SOM_CORE** | **Overhead** |
|---|---|---|---|
| Flip-flops | 2597 | 3406 | 31,15% |
| Slices | 2158 | 2858 | 32,44% |
| LUTS | 2664 | 2775 | 4,16% |
| IOBS | 79 | - | - |
| BRAM | 29 | 26 | - 11,65% |

Com a integração do *hardware* do SOM ao barramento OPB, foi criado o *som*_core. Este possibilita por meio do *software* embarcado acessar o dispositivo de *hardware*. Para isso, deve ser desenvolvido um *driver* que abstraia e oculte os comandos e as instruções necessárias para controlar, escrever e ler os registradores do novo dispositivo. Esse *driver* pode ou não ser integrado ao SOE.

De acordo com a metodologia de fluxo de projeto em FPGA, deveriam ser feitas simulações temporais e análise de consumo de potência. Entretanto, devido à dificuldade e ao grande custo computacional necessário, a etapa de simulação temporal não foi realizada. O principal motivo foi a utilização de componentes já validados e testados na plataforma, não necessitando de uma nova simulação de todos eles. Como

consequência, a análise de consumo de potência também não foi realizada, pois utiliza dados resultantes da simulação temporal.

Após a conclusão do sistema embarcado na ferramenta XPS, pode-se gerar os dados de *bitstream* e programar o FPGA para funcionamento. Agora resta a etapa de configuração do *software* embarcado (SOE e aplicações) para ser carregado na memória do sistema.

### 4.5.2 *Software* do Sistema Embarcado

O *software* utilizado para aplicações em sistemas embarcados é fortemente influenciado pelas restrições exercidas pela plataforma de *hardware* utilizada. De acordo com (CARRO; WAGNER, 2003), o projeto de *software* embarcado deve atender aos seguintes requisitos:

- Limite da memória disponível: dependendo da aplicação e da plataforma de *hardware*, pode estar disponível memória na ordem de apenas *kilobytes* (Kb) para programa e dados. Essa é uma restrição muito forte, pois, caso comparado com a disponibilidade de memória "infinita" por parte de sistemas *desktop*, a construção de um *software* embarcado pode ser muito difícil. Para alcançar essa redução de memória, o *software* deve apenas desempenhar as tarefas mínimas exigidas pela aplicação.
- Limite do desempenho do processador: devido a limitações impostas durante a construção do processador, como reduzir o consumo de potência e ocupar menos área, o processador pode apresentar um desempenho muito inferior, se comparado a um *desktop*. Desse modo, tenta-se exigir um nível maior de paralelismo no sistema, como por exemplo, em aplicações clássicas de sistemas embarcados como processamento de imagens, automação industrial, telecomunicações, etc.
- Limite no consumo de potência: está diretamente ligado aos dois requisitos anteriores. A quantidade e o desempenho da memória disponível tem forte impacto na potência, já que memórias mais rápidas consomem mais energia. Um processador com frequência reduzida consome menos energia, mas pode não satisfazer o desempenho exigido pela aplicação. Dependendo da aplicação, o consumo de potência pode ser o requisito mais importante a ser tratado, por exemplo, no caso de aparelhos portáteis como celulares e *laptops*.
- Maior custo de projeto do *software*: os requisitos anteriores influenciam nos recursos de projeto disponíveis (ferramentas, linguagens, plataformas, etc.) que serão utilizados durante o desenvolvimento. As ferramentas não são totalmente integradas em uma metodologia de engenharia de *software*, como nos sistemas convencionais. Além disso, as ferramentas para modelagem, compiladores, *debugers*, programação e as plataformas são muito distintas das de *softwares* convencionais. Geralmente, o reuso de *softwares* convencionais migrados para embarcados não é trivial e demanda um esforço extra. Em razão disso, as aplicações embarcadas resultam em mais custos e uma demanda maior de tempo durante o projeto do *software*.

Devido à complexidade do projeto de *software* embarcado, foi desenvolvida uma plataforma de *software* que objetiva criar uma camada de abstração dos conceitos de

*hardware* e facilitar a interação do *software* com o *hardware*. Isso pode ser feito por meio de uma camada de alto nível que tem por função tratar cada aplicação como sendo uma tarefa a ser executada pelo *hardware*.

Esse *software* utilizado como camada de abstração é um Sistema Operacional Embarcado (SOE) (SILBERSCHATZ, 2008) (CARRO; WAGNER, 2003). Ele é classificado como um *software* embarcado específico que tem alta interação com o *hardware*. Além disso, é composto por um conjunto de *softwares* implementados em diferentes níveis de abstração, desde a linguagem de máquina até a de mais alto nível.

Por ser, geralmente, dedicado a aplicações embarcadas, o SOE também está sujeito aos mesmos requisitos naturais exigidos pelos sistemas embarcados.

Conforme (SILBERSCHATZ, 2008)(CARRO; WAGNER, 2003), um SOE deve suportar a execução de múltiplos programas ou tarefas, apresentando, aparentemente, uma execução concorrente, em que cada programa em execução é abstraído para uma tarefa sob o domínio do sistema operacional. A seguir estão algumas das vantagens do uso de um SOE:

- Multitarefa e comunicação entre processos permitem que as aplicações sejam particionadas em um conjunto menor e mais gerenciável.
- Com particionamento do *software,* proporciona um teste fácil, redução do tempo de projeto e reuso de código.
- No código da aplicação, podem-se utilizar diversas plataformas de *software* para programação, como linguagens C/C++, POSIX, JAVA, etc.
- Temporização e sequenciamento podem ser removidos do código da aplicação e passam a ser responsabilidade do sistema operacional, incluindo mais transparência ao desenvolvedor.
- Garantia no atendimento de *deadlines* em processos críticos da aplicação.

Segundo (SILBERSCHATZ, 2008)(CARRO; WAGNER, 2003), para fornecer as funcionalidades de um SOE, o sistema operacional deve prover alguns serviços para as aplicações, entre eles:

- Gerência de processos: inclui aspectos como a criação, carga e o controle da execução de processos (escalonamento).
- Comunicação entre processos: deve fornecer serviços de sincronização, detecção, tratamento de *deadlocks* e mecanismos de trocas de mensagens. Isso visa esconder detalhes de mais baixo nível da infraestrutura de comunicação da aplicação.
- Gerência de memória: inclui a criação, remoção e proteção de dados manipulados pelo sistema.
- Gerência de E/S: é responsável pelo controle de comunicação com periféricos e o tratamento de interrupções do *hardware*. Também deve oferecer *drivers* para periféricos, escondendo detalhes das interfaces e da infraestrutura do *hardware* da aplicação.

Um SOE, além de oferecer os serviços já vistos, deve também atender requisitos temporais associados à execução de processos, tais como *deadline*, execução de processos periódicos, escalonamento previsível e baixa latência. Esses requisitos

dependem da aplicação exigir tratamentos de recursos de tempo real (SILBERSCHATZ, 2008).

A principal consequência das restrições temporais está no escalonamento de tarefas, pois em um SOE, as tarefas têm prioridades associadas para garantir o atendimento das restrições temporais, além de serem preemptivas. A questão das prioridades é crucial em sistemas temporais, caso uma prioridade não seja atendida em tempo hábil, pode ocorrer o vencimento do *deadline* e, dependendo da aplicação, causar danos sérios na aplicação.

Os sistemas embarcados estão inseridos nas mais diversas aplicações do cotidiano humano, como aviões, carros, eletrodomésticos, celulares, instrumentos médicos, etc. Esses sistemas são caracterizados por serem limitados fisicamente pela memória reduzida, consumo de potência, interfaces não-amigáveis, desempenho reduzido, etc. Em geral, todos esses sistemas utilizam algum SOE, que além de ser limitado fisicamente pelo *hardware*, é responsável por toda a execução dos *softwares* da aplicação.

Neste trabalho, foi adotado o SOE, conhecido como PetaLinux[9], para ser utilizado como plataforma de *software* no sistema embarcado para o Sistema de Manutenção Inteligente.

#### *4.5.2.1 Sistema Operacional PetaLinux*

O PetaLinux (WILLIAMS; PETALOGIX, 2008) é um sistema operacional para aplicações embarcadas (SOE) que trata restrições como memória disponível e desempenho de processamento.

O PetaLinux é um sistema operacional para processadores de pequeno desempenho e para executar sobre a arquitetura do processador MicroBlaze em FPGA. Os requisitos mínimos de *hardware* exigidos para execução do PetaLinux são:

- Processador MicroBlaze em FPGA.
- Timer.
- Controlador de interrupção.
- Dispositivo de entrada e saída padrão (RS-232).
- Memória RAM.
- Memória FLASH (opcional).

PetaLinux tem sua origem no sistema operacional GNU/Linux, sendo uma customização do sistema operacional para sistemas embarcados *uCLinux*[10]. A principal característica desse projeto está na execução sobre microprocessadores sem Unidade de Gerenciamento de Memória (MMU – *Memory Management Unit*) e o tamanho reduzido da imagem do *kernel*.

Como o PetaLinux é baseado nos padrões de operação do GNU/Linux e originou-se do *uCLinux*, o *kernel* do sistema continua sendo licenciado como solução de *software* livre e de código-fonte aberto. Contudo, é importante notar que os *scripts* para

---

[9] Site oficial do projeto PetaLinux - www.petalogix.com

[10] Site oficial do projeto *uCLinux* - www.uclinux.org

configuração e compilação do *kernel* são proprietários da empresa que administra o PetaLinux, não permitindo a modificação destes.

Além de ser um sistema operacional, o PetaLinux é uma plataforma de projeto de *software* para sistemas embarcados, integrando um SOE a uma plataforma de *hardware* baseada em FPGA. Como em um FPGA o *hardware* pode ser customizado ao gosto do projetista, o PetaLinux pode customizar o sistema operacional para as necessidades de cada sistema embarcado projetado.

Desse modo, o PetaLinux é integrado na ferramenta XPS, visto na seção anterior, para o projeto do *hardware* do sistema embarcado. Essa integração permite configurar os parâmetros do sistema operacional de acordo com os requisitos do sistema embarcado projetado.

De acordo com o tutorial (WILLIAMS; PETALOGIX, 2008), o XPS é uma ferramenta voltada para o *hardware* em torno do FPGA, ele apresenta como saída para o PetaLinux um arquivo com as configurações do *hardware* e do *software* (sistema operacional). Esse arquivo deve ser utilizado durante a compilação da imagem do *kernel*, dentro do ambiente do PetaLinux, uma vez disponibilizada informações, como quais componentes de *hardware* são utilizados, seus endereços de mapeamento, memória disponível, configuração de *timer* e *clock*, configuração de entrada e saída padrão, entre outros.

A plataforma do PetaLinux deve ser instalada em ambiente GNU/Linux, para o desenvolvimento das aplicações, da configuração, compilação da imagem do *kernel*, dos *links* para bibliotecas do *kernel* e bibliotecas-padrão. Ademais, fornece as ferramentas clássicas de programação, como compilador, *debuger* e editor de código-fonte.

No PetaLinux, é fornecido um ambiente de desenvolvimento *cross compiler*, que possibilita compilar *softwares* de uma paltaforma-alvo diferente daquela utilizada para compilação. A partir de um PC rodando GNU/Linux, pode-se compilar *softwares* para MicroBlaze. Isso é importante para compilar a imagem do *kernel* e também para o desenvolvimento das aplicações.

Após uma breve apresentação da plataforma do PetaLinux, agora o foco são as particularidades do sistema operacional adotado (YAGHMOUR, 2008) (NIKKANEN, 2003) (NAGARAJAN; ASOKAN, 2007). O PetaLinux tem herdado muitas características do GNU/Linux convencional, como a pilha TCP/IP e os diversos protocolos de comunicação. Também suporta diversos sistemas de arquivos, além dos especiais para sistemas embarcados. O PetaLinux foi projetado para manter compatibilidade com as aplicações desenvolvidas nos padrões do GNU/Linux.

Para manter o nível de compatibilidade das aplicações do PetaLinux com o GNU/Linux convencional, é utilizada uma versão reduzida das bibliotecas-padrão do C (conhecida por *libc*) utilizada pelo GNU/Linux convencional. Desse modo, foi utilizada no PetaLinux a biblioteca *uClibc* (com origem do *uClinux*), que mantém compatibilidade com o padrão de arquitetura do GNU/Linux e foi projetada para ocupar menos espaço e suportar processadores sem MMU. Com isso, o PetaLinux proporciona aos desenvolvedores reutilizarem aplicações já desenvolvidas, seguindo o mesmo padrão POSIX do GNU/Linux.

As duas principais diferenças do PetaLinux comparando-o com o GNU/Linux convencional é a ausência de MMU e o tamanho da imagem do *kernel*. Todavia, a

principal vantagem que o PetaLinux oferece em relação ao GNU/Linux é o tamanho da imagem do *kernel* (YAGHMOUR, 2008).

A imagem do *kernel* é um arquivo que armazena todas as instruções do *kernel* compilado em um formato compactado. Por exemplo, um *kernel* compilado somente com as opções necessárias para uma arquitetura de um processador, sistema de arquivos e dispositivos periféricos, pode ocupar aproximadamente 400 KBytes de memória. Após o *boot*, a imagem final do *kernel* (incluindo as aplicações) já carregada na memória, pode ocupar aproximadamente 2MB da memória.

Após a compilação e gerada a imagem do *kernel*, o arquivo da imagem deve ser carregado e gravado na memória do sistema embarcado para inicializar o processo de *boot*.

Para o estudo de caso adotado no presente trabalho, a imagem do *kernel* é carregada e gravada diretamente no espaço da memória RAM. O método mais simples para efetuar *boot* é executar diretamente a imagem do *kernel* no espaço de memória. Para isso, é preciso endereçar o fluxo de execução do processador para a primeira instrução do *kernel*. Pode-se correr o risco de todas as configurações de inicialização do *hardware* não serem realizadas antes da execução do *kernel*. Esse método pode ser utilizado quando uma memória não-volátil (FLASH), para armazenar a imagem do *kernel*, assim como um *bootloader*, estiverem indisponíveis.

Outro método de *boot* pode ser por meio de um pequeno programa, conhecido por *bootloader*, que é uma opção mais segura e flexível. Este é o primeiro programa a ser executado pelo processador e trata com antecedência as configurações de inicialização do *hardware*, além de carregar a imagem do *kernel* para a memória, a partir de uma memória não-volátil, *link* de Internet, porta serial, etc. Existem diversos programas utilizados como *bootloader*, cada um com suas vantagens e desvantagens. Esses programas exigem um espaço de armazenamento muito pequeno e possibilitam uma segurança maior durante a inicialização do sistema operacional (YAGHMOUR, 2008).

Como neste trabalho foi adotado utilizar o método mais simples de *boot*, um inconveniente foi detectado. Em cada instante que o *hardware* é energizado, uma nova recarga da imagem do *kernel* para memória RAM se torna necessário. Então, nesse caso, foi utilizada a ferramenta XPS para desempenhar essa tarefa.

Durante o projeto do sistema embarcado, foi utilizado um módulo para *debug* do processador (*debug_module*), que possibilita ler e escrever todos os registradores, executar instruções diretamente no processador e acessar a memória. Por meio dessa ferramenta, a imagem do *kernel* é carregada para a posição de memória configurada como início do sistema operacional. Após, o registrador PC (*Program Counter*) do processador é modificado para apontar para o endereço onde estão as instruções de inicio da imagem do *kernel*. Desse modo, o *boot* do sistema operacional PetaLinux neste trabalho é executado.

Como o *boot* é o primeiro programa executado pelo processador, é responsável por descompactar a imagem do *kernel*, carregá-la para a memória e iniciar a execução. Durante o *boot*, são realizadas uma série de tarefas importantes de preparo do *hardware* para a execução correta do sistema operacional.

A inicialização do *kernel* no PetaLinux segue o mesmo padrão de funcionamento do *boot* do GNU/Linux convencional. Durante o carregamento do sistema, a execução visa detectar e inicializar todos os dispositivos de *hardware*. Serão configuradas as rotinas

de tratamento de interrupções e carga dos *drivers* de dispositivos de *hardware* (YAGHMOUR, 2008).

Após a configuração do *hardware* é carregado o módulo de execução dos *scripts* de inicialização (programas de configuração do *software*), que são executados apenas uma vez durante o *boot*. A última etapa do *boot* é executar os *scripts* para carga de aplicações nativas do sistema, que são programas importantes para colocar o dispositivo em ambiente de trabalho, por exemplo, montar as partições de sistemas de arquivos, configurar a interface de rede e executar as aplicações do usuário.

Após a execução de todo o processo de *boot*, o sistema está pronto para ser utilizado como um ambiente de trabalho para a aplicação. Entretanto, para que isso se consolide, algumas etapas precisam ser esclarecidas, como o sistema de arquivos é utilizado e como são desenvolvidas as aplicações para o sistema.

O PetaLinux suporta diversos sistemas de arquivos padrões em ambientes GNU/Linux, além de incluir os especiais para sistemas embarcados. Geralmente, em um sistema embarcado, é utilizado uma combinação de diferentes unidades de armazenamento, com memória FLASH, RAM e discos rígidos. Isso exige utilizar um sistema de arquivos específico para cada unidade de armazenamento.

Durante a execução dos *scripts* para carga de aplicações nativas pelo *boot* do sistema, é criado e montado um sistema de arquivos clássicos em ambientes GNU/Linux, o RAMFS. O RAMFS (YAGHMOUR, 2008). é um sistema de arquivos para memória RAM, que cria uma partição de dados contíguos no espaço de memória após o endereço final do *kernel*. A partição é criada utilizando o sistema de arquivos do tipo *ext2*, muito comum em ambientes GNU/Linux. O *ext2* é um sistema de arquivos simples, eficiente, com alto desempenho e tem suporte a *links* entre os arquivos.

A partição RAMFS geralmente é utilizada para armazenar arquivos de *logs* (históricos) produzidos pelas aplicações e como espaço de arquivos para o usuário. A configuração da partição é feita antes da compilação do *kernel*, quando é ajustado o tamanho da partição. Como neste estudo de caso não está disponível uma memória FLASH, apenas uma memória RAM, a partição RAMFS foi adotada como sistema de arquivos-padrão neste trabalho.

Com a partição de usuário disponível para uso, pode-se desenvolver e executar os *softwares* de aplicação do usuário e utilizar normalmente a partição como uma unidade de armazenamento.

O desenvolvimento das aplicações do usuário deve ser feito antes da compilação do *kernel*, pois são inseridas dentro da imagem gerada.

A plataforma do PetaLinux (WILLIAMS; PETALOGIX, 2008) disponibiliza um ambiente de *building* completo para desenvolvimento de *softwares* para aplicações, que podem utilizar todos os recursos disponíveis pelas bibliotecas (*uclibc*) e pelo sistema operacional, como processamento multitarefas, reutilização de outros programas ou partes de código, etc.

No entanto, é preciso atenção especial ao desenvolver as aplicações, devido à ausência de MMU no PetaLinux. Essa característica influencia diretamente o trabalho do programador.

Conforme (SILBERSCHATZ, 2008) o MMU, quando disponível, é utilizado pelo sistema operacional para realizar o mapeamento em tempo de execução dos endereços

lógicos (endereço gerado por instruções) para físicos (endereços na memória física). Assim, na associação de endereços em tempo de execução, os endereços lógicos e físicos são diferentes. O MMU faz essa conversão transparente para o programador.

Como neste trabalho foi adotado o processador MicroBlaze (XILINX, 2003), que não tem MMU, a associação de endereços é feita em tempo de compilação. Desse modo, a associação de endereços lógicos e físicos tem valores idênticos.

Quando o módulo de MMU está disponível, este fornece um nível de proteção para as aplicações executadas e cria um modelo de memória virtual (SILBERSCHATZ, 2008). Provê uma técnica para proteger o sistema operacional contra aplicações do usuário e protege as próprias aplicações dos usuários umas das outras. Isso melhora o gerenciamento dos recursos de memória, criando a ideia de a memória ser “ilimitada e única”, além de o gerenciamento do uso da memória ser totalmente transparente ao programador.

Quando o módulo de MMU está indisponível, os recursos de memória se tornam escassos e o gerenciamento se torna muito mais custoso e de alta responsabilidade por parte do programador (SILBERSCHATZ, 2008). Então, quando se trabalha sem a MMU, significa que todos os programas da memória estão literalmente mapeados em uma sequência física no espaço de memória. Isso é chamado de modelo de memória *flat*, em que os dados são gravados em um espaço linear e acessados diretamente pelo endereço. Este é um método simples de mapeamento, comparando com os modelos de paginação ou segmentados.

Nesse caso, não existe proteção por parte do sistema operacional ao espaço de memória. Por exemplo, um ponteiro para um endereço inválido da memória acessado por uma aplicação do usuário pode causar um erro no endereçamento e corromper o fluxo de execução do processador ou paralisar a execução. Ou ainda uma aplicação pode acessar espaço de memória de outra aplicação e corromper os dados.

Esses exemplos mostram a importância que o programador tem com o teste correto do código-fonte implementado antes de validar aplicação. Esses problemas podem acontecer porque o sistema operacional permite que a aplicação possa acessar qualquer endereço no espaço de memória e modificá-lo, o que pode prejudicar o fluxo de execução do sistema.

Como neste estudo a MMU não está disponível. Em função desse requisito, deve-se mudar o paradigma de programação das aplicações, pois exige modificações e adaptações a algumas chamadas ao sistema e, claro, o programador se torna responsável pelo gerenciamento do uso da memória.

Quando for reutilizar *softwares* que se originaram em sistemas com MMU, as partes do código-fonte, que utilizam alocação dinâmica de memória, devem obrigatoriamente garantir que todos os espaços de memória alocados sejam desalocados e, dependendo do caso, até mesmo essas partes do código devem ser reprogramadas.

Nas próximas seções, serão abordados o desenvolvimento do *software* da aplicação do Sistema de Manutenção Inteligente utilizando a plataforma do PetaLinux.

#### *4.5.2.2 Integração do som_core ao PetaLinux*

No sistema embarcado para o Sistema de Manutenção Inteligente visto na Figura 4.21, o componente *som_core* precisa ser integrado ao sistema operacional PetaLinux.

Essa integração é feita pelo desenvolvimento de um *device driver* (YAGHMOUR, 2008), que tem por função tornar transparente, para o programador, os acessos de controle do *hardware*. De acordo com o componente *som_core* apresentado na Figura 4.22, são utilizados quatro registradores mapeados em memória (XILINX, 2005d). Estes controlam e armazenam os resultados de operação do dispositivo.

O *device driver* tem que operar o componente *som_core* por meio de escrita e leitura nos registradores. As seguintes operações precisam ser realizadas para controlar o componente:

1. START: a *entrada* de dados é feita mediante *escrita* no Registrador de Controle, que comandará o sinal de START do SOM (para inicializar o funcionamento do componente). É utilizado o bit 0 (menos significativo).
2. PRONTO: a *saída* de dados é feita mediante *leitura* do Registrador de Controle, que armazenará o sinal de PRONTO do SOM (fim de operação e resultado disponível). É utilizado o bit 1 (menos significativo).
3. X: a *entrada* de dados (elementos dos vetores de entrada) é feita mediante *escrita* no Registrador de Entrada (**x**), que armazenará o dado a ser apresentado ao sinal **x** do SOM (carregar vetores de entrada).
4. BMU: a *saída* de dados, como resultado do SOM, é feita mediante *leitura* do Registrador de Saída (**BMU**) do SOM (índice do neurônio vencedor).
5. ERRO: a *saída* de dados, como resultado do SOM, é feita mediante *leitura* do Registrador de Saída (**ERRO**) do SOM (erro de quantização).
6. RESET: a *entrada* de dados é feita mediante a execução de comando no barramento OPB, que enviará o sinal de RESET para o SOM (para reinicializar o componente).

Do ponto de vista do programador, o *device driver* deve fornecer funções em *software*, desenvolvidos em linguagem C, para que possa desempenhar as tarefas de controle das operações do componente. Isso possibilita ao programador trabalhar com os sinais de tratamento do dispositivo de um modo mais transparente e em alto nível.

Na Tabela 4.7, são apresentadas as funções do *device driver* desenvolvidas em linguagem C, que devem ser utilizadas para programação de aplicações dentro do ambiente do PetaLinux. Nota-se que todas as funções têm uma entrada em comum, o *endereço-base* com a posição do componente do SOM na memória.

Tabela 4.7: Funções implementadas em linguagem C do *device driver*.

| **Função** | **Entrada** | **Saída** | **Definição** |
|---|---|---|---|
| SOM_CORE_start | Valor do sinal para start | Valor no registrador, bit (0) | Programar o sinal de início de funcionamento do dispositivo |
| SOM_CORE_reset | - | - | Programar o sinal para reinicializar o dispositivo |
| SOM_CORE_readStatus | - | Valor no registrador, bit(1) | Ler sinal de fim de funcionamento e dados prontos para a leitura |
| SOM_CORE_writeX | Valor de um elemento do | - | Inserir elemento do vetor de entrada no dispositivo |

| | | | |
|---|---|---|---|
| | vetor x | | |
| SOM_CORE_compute | Ponteiro de memória para o início do vetor de entrada x | - | Realiza todo o cálculo do algoritmo do SOM, inserindo o vetor de entrada no dispositivo, e aguarda o fim dos cálculos até o resultado |
| SOM_CORE_readBMU | - | Valor do BMU | Ler o valor na saída do BMU |
| SOM_CORE_readError | - | Valor do ERRO | Ler o valor na saída do ERRO |

Dentre as funções, a única que não realiza exclusivamente escrita ou leitura dos registradores é a função SOM_CORE_compute. Essa função realiza a carga do vetor de entrada do SOM e controla a operação de separar cada elemento do vetor de entrada. O algoritmo implementado em linguagem C é apresentado a seguir:

```
char SOM_CORE_compute(Xuint32 BaseAddress, Xuint32 *x){
     int i,k;
     Xuint32 ready;
     //NEURONS - constante que representa a quantidade de neurônios do SOM
     //SIZE - constante de dimensão do vetor de entrada
     for (k=0; k<NEURONS; k++){ //laço para varrer todos os neurônios
          for (i=0; i<SIZE; i++){ //laço para varrer elementos do vetor
              SOM_CORE_writeX(BaseAddress, x[i]); //apresenta vetor ao SOM
          }
     }
     //Verificar o sinal de finalização do cálculo, aguardando hardware
     ready = SOM_CORE_readStatus(BaseAddress);
     while (!ready){
          ready = SOM_CORE_readStatus(BaseAddress);
     }
     return 0;
}
```

Por meio dessas funções do *device driver*, que integram o *som_core* ao sistema operacional PetaLinux, é possível fazer o desenvolvimento de aplicações. Isso será visto na próxima seção.

#### *4.5.2.3 Projeto da aplicação*

Para o desenvolvimento da aplicação, estão disponíveis todos os recursos que um ambiente GNU/Linux oferece, proporcionando maior flexibilidade ao programador.

Neste trabalho, a aplicação não tem apenas a finalidade de implementar um estudo de caso para o Sistema de Manutenção Inteligente, mas também apresentar um caso concreto de como utilizar o *device driver* dentro de uma aplicação no PetaLinux.

No contexto do Sistema de Manutenção Inteligente, visou-se implementar uma aplicação que utilize o *som_core* para detectar e diagnosticar falhas. A aplicação pode realizar ao mesmo tempo a detecção e o diagnóstico de falhas, mas depende dos dados utilizados para o treinamento do SOM.

Devido à característica do *som_core*, a base de treinamento da rede neural pode ser somente para detecção ou diagnóstico. Assim, a aplicação terá como saída apenas um dos casos. Portanto, para a aplicação realizar a detecção, o *som_core* deve ser treinado com dados que visam a detecção, e para diagnóstico, treinado com dados que visam o diagnóstico.

Uma sugestão para realizar as duas operações ao mesmo tempo, seria existirem no *som_core* duas redes neurais, uma para detecção e outra para diagnóstico. Nesse caso, a aplicação poderia extrair as duas informações juntamente.

Do ponto de vista da aplicação, o que diferencia a detecção do diagnóstico são apenas as saídas utilizadas do *som_core*. A implementação da aplicação pode ser única, pois o custo computacional para detecção ou classificação é igual, já que apenas são lidos os registradores no *som_core* (BMU ou ERRO).

Neste estudo de caso, foi adotada a leitura de dois registradores de saída. O mesmo código de programa pode ser aplicado tanto para a detecção quanto para o diagnóstico, o que muda entre eles é apenas o treinamento do SOM.

A aplicação foi implementada em linguagem C. Foram utilizados alguns recursos tradicionais do sistema operacional, como acesso a arquivos, função de conversão numérica e ponteiros de memória (sem uso de alocação dinâmica). A seguir, segue um trecho do código-fonte implementado:

```
#include <stdio.h>
#include <stdlib.h>
#include "xparameters.h"
#include <linux-2.6.x/drivers/xilinx_common/xio.h>
#include "som_core.h" //Definição dos endereços de memória do som_core
const int SIZE = 20; //dimensão do vetor de entrada
const int NEURONS = 90; //quantidade de neurônios do SOM
int main( ){
     FILE *f, *fbmu, *ferro;
     char *fim;
     float *erro; //armazenar erro de quantização do som_core
     float vetor[SIZE];
     Xuint32 result; //armazena as saídas do som_core
     //Realizar reset do som_core
     SOM_CORE_reset(XPAR_SOM_CORE_0_BASEADDR);
     //Ler dados dos vetores de entrada e criar arquivos de saída
     f = fopen("/home/teste.txt", "rt");
     ferro = fopen("/tmp/resultado_erro.txt", "wt");
     fbmu = fopen("/tmp/resultado_bmu.txt", "wt");
     //Ler um vetor de entrada por vez e armazenar na memória
     fim = getVetorEntrada(vetor,f);
     while (fim != NULL){
          //Calcular BMU e ERRO para um vetor de entrada
          SOM_CORE_start(XPAR_SOM_CORE_0_BASEADDR, 1);
          SOM_CORE_start(XPAR_SOM_CORE_0_BASEADDR, 0);
          //Inicia o algoritmo para calcular o algoritmo do SOM
          SOM_CORE_compute(XPAR_SOM_CORE_0_BASEADDR, vetor);
          //Ler registrador de saída do BMU e armazenar em variável
          result = SOM_CORE_readBMU(XPAR_SOM_CORE_0_BASEADDR);
          fprintf(fbmu, "%d \n\r", result);
          //Ler registrador de saída do ERRO e armazenar em variável
          result = SOM_CORE_readError(XPAR_SOM_CORE_0_BASEADDR);
          erro = &result; //cast para transformar em número real
          fprintf(ferro, "%e \n\r", *erro);
          //Realizar reset do som_core para próximo vetor de entrada
          SOM_CORE_reset(XPAR_SOM_CORE_0_BASEADDR);
          //Carregar próximo vetor de entrada
          fim = getVetorEntrada(vetor,f);
     }
     fclose(f); fclose(ferro); fclose(fbmu);
     return 0;
}
```

Essa implementação é apenas um exemplo para um caso de teste da aplicação que faz uso do componente *som_core*. O componente foi implementado de forma que, antes de entrar em cada laço de iteração, um vetor de entrada completo é lido. Após, um ponteiro para o início desses dados é repassado para a função de carga de vetores ao SOM. Os resultados são salvos e repete-se novamente esse ciclo até não existirem mais vetores de entrada.

Essa aplicação funciona de forma a simular dados de entrada por meio de um arquivo de teste (`teste.txt`) que contém os vetores de entrada. Os resultados da aplicação também foram armazenados em um arquivo (`resultado_erro.txt` e `resultado_bmu.txt`) para posterior transferência a um PC e para fazer as devidas análises em MATLAB.

Para um protótipo de uma possível aplicação real (testar esse sistema embarcado em um atuador elétrico real), seria necessário incrementar o sistema embarcado. Esses novos recursos seriam: utilizar entradas e saídas analógicas para leitura dos sensores, implementar a etapa de processamento de sinais e implementar uma interface homem-computador para a apresentação de resultados.

## 4.6 Resumo do Capítulo

Neste capítulo, foram abordados os passos necessários para a construção do protótipo de um sistema embarcado para um Sistema de Manutenção Inteligente de válvulas elétricas. Esse protótipo visa implementar técnicas computacionais de redes neurais, que tem como objetivo tolerar falhas na aplicação-alvo.

As técnicas de tolerância a falhas definidas para o trabalho foram: a detecção, o diagnóstico e a predição de falhas. Estas foram utilizadas para aumentar a confiabilidade do sistema industrial.

O objetivo do protótipo é desempenhar a função de um sistema de monitoramento *on-line* do equipamento industrial. Por meio da aplicação de técnicas computacionais, esse sistema pode ler os sinais coletados, processar e apresentar informações na saída como: descobrir com antecedência a ocorrência de uma falha, identificando onde ela ocorrerá e informar à equipe de manutenção.

Para possibilitar esse monitoramento *on-line* do equipamento industrial, é necessário que o protótipo seja implementado em uma plataforma de sistema embarcado. Desse modo, a ferramenta de monitoramento pode ser acoplada ao chão de fábrica, diretamente no equipamento, para realizar as tarefas de monitoramento.

Como visto neste capítulo, foi definida uma estrutura para o sistema de manutenção inteligente, do ponto de vista de tolerância a falhas, para projeto do sistema embarcado. Essa estrutura apresenta as ferramentas para detecção, diagnóstico e predição de falhas, que utilizam o SOM como técnica computacional. Além disso, foram apresentados os detalhes de cada algoritmo para desempenhar cada uma das tarefas.

Em seguida, essa estrutura foi implementada em *software*, utilizando a ferramenta MATLAB. Isso foi necessário para realizar simulações do funcionamento das partes da estrutura do Sistema de Manutenção Inteligente, pois por meio desses experimentos foi possível validar a estrutura e comparar com os resultados do sistema embarcado.

Após validação do sistema em *software*, foi iniciado o projeto do protótipo em *hardware*. Foi adotada a plataforma de FPGA da Xilinx, utilizando a placa de

desenvolvimento para experimentos fabricada pela Digilent, XUP Virtex-2 PRO. Essa plataforma de *hardware* apresenta muitos recursos computacionais disponíveis para explorar o espaço de projeto do sistema embarcado.

É importante destacar que a prototipação das técnicas de tolerância a falhas na plataforma de FPGA visa implantar nela a rede neural SOM. As técnicas de tolerância a falhas implementadas em FPGA foram a detecção e o diagnóstico de falhas. A etapa de predição foi apenas validada em *software,* utilizando o MATLAB, e será prototipada em *hardware,* em trabalhos futuros.

Foi desenvolvido um protótipo em VHDL da rede neural SOM. O algoritmo do SOM foi detalhado e criado um porte para o *hardware*. Para desempenhar as tarefas de detecção e diagnóstico, foi implementado apenas o algoritmo de teste em VHDL e utilizado um SOM já treinado por *software*.

Durante o projeto do algoritmo de recuperação, foi definida a parte operativa dos componentes de *hardware* e a Máquina de Estados para controlar o dispositivo. Este foi testado e validado na plataforma de FPGA.

Durante o desenvolvimento do *hardware* em VHDL, foram vencidas muitas barreiras de projeto, como a memória disponível, o desempenho e a dinâmica de funcionamento do algoritmo. Tais barreiras se originam em características particulares do algoritmo do SOM, como o custo computacional de treinamento, a suposição de memória “ilimitada” e por ser um algoritmo para avaliação de resultados puramente empíricos.

Mesmo com essas dificuldades no projeto, o protótipo do SOM em *hardware* foi aprovado em todas as etapas da metodologia de projeto em FPGA. Foram realizadas simulações de nível comportamental, de nível temporal (com atrasos) e de consumo de potência. Para realizar tais simulações, foram utilizadas as ferramentas disponíveis pela Xilinx, o *ISE* e o *ModelSim*. Por fim, o projeto foi aprovado em todas as simulações, estando pronto para ser implantado em um sistema embarcado mais complexo.

O sistema embarcado foi projetado utilizando uma plataforma de desenvolvimento da Xilinx e uma placa de desenvolvimento fabricada pela Digilent. Essa plataforma disponibiliza processadores, memórias, barramentos e demais recursos computacionais para prototipação.

Ademais, foi definida uma arquitetura mínima do sistema embarcado necessário para implementar o Sistema de Manutenção Inteligente. Este é composto por microprocessador, memória, barramento, periféricos e o *hardware* do SOM (*som_core*).

Devido aos requisitos de um sistema embarcado (limite de memória, limite de desempenho e consumo de potência), decidiu-se projetar um hardware mínimo que possibilite a execução de um sistema operacional embarcado (SOE).

Foi definido utilizar o microprocessador MicroBlaze e demais periféricos compatíveis (barramento OPB, timer, comunicação, etc). Todos esses recursos são fornecidos e estão disponíveis na plataforma de FPGA. Assim, o sistema computacional embarcado projetado, deve fornecer os recursos necessários para executar o sistema operacional.

Com a definição do sistema embarcado, foi adicionado ao projeto o componente de *hardware* do SOM. Para que este opere corretamente, foi preciso desenvolver um *wrapper* que faça a conversão dos sinais do componente para o padrão de comunicação

do barramento. Esse *wrapper* foi desenvolvido e acoplado ao sistema, além de ter sido validado e testado.

Com a síntese de todo o sistema embarcado para o FPGA, foi também realizada uma avaliação da área ocupada pelo sistema como um todo e feita uma comparação particular entre a área ocupada do *hardware* do SOM com o *wrapper*. O *wrapper* levou um aumento de área da ordem de 28%, que é referente aos componentes de integração ao barramento.

Com o *hardware* do sistema embarcado pronto, foram iniciados os trabalhos de instalação, customização e compilação do sistema operacional adotado, o PetaLinux. Esse sistema operacional é uma versão customizada especialmente para executar sobre a plataforma FPGA, com microprocessador MicroBlaze, e *drivers* para os periféricos da placa de desenvolvimento.

Foram apresentados conceitos básicos de *software* para sistemas embarcados e também sobre o sistema operacional PetaLinux. Como este é derivado do GNU/Linux, eles têm muitos conceitos em comum, mas a principal diferença está na memória ocupada pelo *kernel*. Além disso, as características internas e particularidades do PetaLinux que afetam o desenvolvimento de aplicações também foram apresentadas. Para o desenvolvimento de *softwares*, devem ser vencidas algumas limitações devido ao sistema não ter MMU, o que transfere muitas responsabilidades de segurança e confiabilidade do *software* para o programador.

Por fim, foi apresentado o desenvolvimento do *device driver*, necessário para acessar o *hardware* do SOM pelo *software*, além da apresentação de um exemplo de aplicação em *software* que utiliza o componente.

# 5 EXPERIMENTOS E RESULTADOS

## 5.1 Introdução

Neste capítulo, serão apresentados os experimentos e resultados utilizando as ferramentas apresentadas e desenvolvidas no capitulo anterior.

O objetivo dos experimentos é comprovar, por meio de resultados, que o Sistema de Manutenção Inteligente funciona para o caso de teste. Além disso, também se deve avaliar o protótipo do *hardware* comparando-o com os resultados obtidos pelas simulações em *software*.

Primeiro, será apresentada a metodologia para realizar os experimentos. Esta pode ser aplicada para qualquer estudo de caso, já que é apresentada em termos gerais.

Em segundo, será apresentado o estudo de caso adotado neste trabalho. Serão abordados os problemas no transporte de derivados de petróleo, na indústria petrolífera e como este trabalho ajuda a resolver tais problemas. Com base neste estudo de caso, serão desenvolvidos os experimentos.

Na terceira etapa, será apresentado o modelo matemático que é utilizado para simular o comportamento físico do processo industrial. A abordagem é a de conhecimento *a priori*, em que por meio de um conhecimento especializado sobre as relações físicas do processo é possível especificar o modelo. Este serve como base para aquisição de dados quantitativos, como valores ou sinais de características do comportamento do equipamento.

Em quarto, será apresentado como realizar o treinamento das redes neurais, utilizando o simulador do modelo matemático como gerador de sinais ou valores. Esses sinais são tratados como dados de histórico, em que grandes quantidades desses dados devem estar disponíveis. Eles serão aplicados a uma técnica de processamento de sinais para extrair determinadas características importantes. A partir disso, serão utilizados como vetores de entrada na rede neural tanto para treinamento quanto para testes.

Após, serão apresentadas as aplicações desenvolvidas para avaliar os resultados para detecção, classificação e predição de falhas. Nestas, serão avaliados apenas os resultados adquiridos por meio das simulações em *software* por intermédio da ferramenta MATLAB.

Por último, serão realizados experimentos de detecção e diagnóstico utilizando o protótipo do sistema embarcado. Este será comparado aos resultados obtidos da

simulação em *software*, em MATLAB, para avaliar se os resultados do sistema embarcado estão corretos.

Nas próximas seções, serão apresentadas as etapas dos experimentos para o sistema embarcado de manutenção inteligente.

## 5.2 Metodologia dos Experimentos

O propósito dos experimentos é aplicar as técnicas e ferramentas vistas no Capítulo 4, utilizando o conjunto de dados ou sinais de entrada para testar e simular o Sistema de Manutenção Inteligente.

Para realizar os experimentos, assume-se de antemão que esteja disponível um conjunto de dados ou sinais de entrada, que serão utilizados para *treinamento* e *testes*. É importante que esses dados reflitam as informações de um período histórico do processo em análise.

Caso não estejam disponíveis os dados de entrada, pode-se utilizar uma ferramenta auxiliar para produzi-los. Nos experimentos, aqui apresentados, foi utilizada uma ferramenta para simular o comportamento do processo.

Os experimentos serão feitos com base inteiramente na proposta de um Sistema de Manutenção Inteligente (SMI) apresentada na Seção 4.2. Nesse sistema, são utilizadas redes neurais do tipo Mapas Auto-organizáveis (SOM) para aprender as características comportamentais do processo a ser monitorado.

Os experimentos foram divididos de dois modos:

- *Experimentos em software*: será utilizada a ferramenta implementada em MATLAB e apresentada na Seção 4.3.
- *Experimentos em hardware*: será utilizado o protótipo do sistema embarcado desenvolvido e apresentado na Seção 4.5.

Os experimentos em *software* têm a finalidade de avaliar os resultados do ponto de vista do SOM. Isso será realizado pela utilização de sinais que sejam realistas e representativos do processo industrial. Serão feitos experimentos para avaliar as tarefas de detecção, diagnóstico e predição de falhas.

As fases a serem realizadas no experimento em *software* são:

1. *Aquisição de dados*: um banco de dados deve estar disponível a fim de refletir o histórico de comportamento do processo. Este deve fornecer dois conjuntos de dados: um para *treinamento* (história do processo) e um para *teste*.
2. *Treinamento*: utilizar os dados de *treinamento* para treinar as redes neurais, pois cada tarefa do SMI (detecção, diagnóstico e predição) utiliza uma rede neural em particular. Por isso, necessita-se separar os dados mais adequados para cada caso conforme algoritmo apresentado na Seção 4.2.
3. *Teste*: utilizar os dados de teste para avaliar a generalização da rede neural. Para cada tarefa do SMI, devem-se separar os dados de teste mais adequados para cada caso conforme apresentado na seção 4.1.
4. *Resultados*: para cada caso do SMI extrair os seguintes resultados:

- *Detecção*: obter da rede neural o gráfico do erro de quantização para cada vetor de teste.
- *Diagnóstico*: obter da rede neural a visualização por *U-Matrix* revelando o mapeamento e as taxas de acertos para cada vetor de teste.
- *Predição*: extrair da rede neural a visualização por *U-Matrix* com a projeção da trajetória percorrida pelos neurônios vencedores. Também extrair o gráfico da relação entre os vetores vencedores e o tempo de chegada da amostra.

Os experimentos em *hardware* visam avaliar e validar o sistema embarcado do ponto de vista do SOM. Isso será realizado pelos mesmos sinais utilizados para o caso de experimentos em *software*. Além disso, serão feitos experimentos para avaliar as tarefas de detecção e diagnóstico de falhas.

Como o SOM foi implementado no sistema embarcado, algumas etapas anteriores não estão disponíveis, como a aquisição de sinais pela entrada analógica e a etapa de processamento de sinais. Devido a isso, esses experimentos têm a finalidade apenas de simular o funcionamento do SOM na prática pelo uso de dados estáticos salvos na memória. Estes fazem o papel de simular a aquisição de dados para testes e executar o sistema embarcado na placa de desenvolvimento.

As fases a serem realizadas no experimento em *hardware* são:

1. *Aquisição de dados*: neste experimento não será tratada a aquisição de dados em tempo real. Os dados para treinamento e testes serão iguais aos utilizados pelo experimento em *software* e devem estar disponíveis. Esses dados serão gravados de forma estática no sistema embarcado para realizar a simulação.
2. *Treinamento*: será utilizado o mesmo SOM treinado pelo experimento em *software* e apenas para as tarefas de detecção e diagnóstico. Os vetores de pesos sinápticos dos neurônios devem ser separados e salvos, em separado, em um arquivo chamado de *mapa*.
3. *Conversão*: converter os valores do *mapa* para a representação numérica binária, seguindo o padrão de Ponto Flutuante IEEE 754.
4. *Síntese*: o arquivo do *mapa* em binário é utilizado durante a síntese do sistema embarcado. O *mapa* é utilizado como entrada de dados para memória interna (BRAM) do componente conforme apresentado na Seção 4.4.1.2.
5. *Programa de teste*: os dados de teste para avaliar os resultados do SOM, são inseridos no programa da aplicação na imagem do *kernel* do PetaLinux. Estes são salvos em um arquivo-texto na memória do sistema para simular a aquisição de dados para testes.
6. *Resultados*: dependendo da tarefa para qual o SOM foi treinado, extrair os seguintes resultados em cada caso:
   - *Detecção*: obter os resultados do erro de quantização coletados na saída ERRO do componente *som_core* e armazenados em um arquivo de saída. Esse arquivo deve ser

transferido para um PC, a fim de processar os dados de resultados e gerar o gráfico do erro de quantização.

- *Diagnóstico*: obter os resultados dos neurônios vencedores coletados na saída BMU do componente *som_core* e armazenados em um arquivo de saída. Esse arquivo deve ser transferido para o PC para processar os dados de resultados. Ademais, gerar a representação de visualização por *U-Matrix* com destaque para o mapeamento e calcular as taxas de acertos para cada vetor de teste.

Nota-se uma fase em comum entre os experimentos em *software* e *hardware*: a fase de treinamento. Esta foi definida ao ser executada no PC, utilizando o MATLAB, devido à reutilização da implementação do SOM Toolbox (ver seção 4.3). O reuso desse *software* já trata de todos os requisitos intrínsecos do algoritmo de treinamento do SOM e, claro, evita transferir o custo computacional de treinamento para o sistema embarcado.

Nas próximas etapas, serão apresentados os experimentos para cada um dos modos (*software* e *hardware*) e em cada um delas serão apresentadas as tarefas do ponto de vista do Sistema de Manutenção Inteligente.

## 5.3 Estudo de Caso

O estudo de caso adotado nos experimentos vem de um setor muito importante para a indústria petrolífera: o transporte e distribuição de derivados de petróleo.

O processo de transporte de derivados de uma plataforma de extração até uma refinaria é realizado por uma rede dutoviária ou gasodutos. Estes são tubulações especiais para transportar os derivados de um local de origem para um destino.

O gasoduto pode percorrer milhares de quilômetros e também interligar equipamentos do processo dentro da refinaria. Em todo o seu percurso podem ser encontradas centenas ou até milhares de válvulas, que são utilizadas para controlar o fluxo do processo durante o transporte.

As válvulas são equipamentos industriais que tem a finalidade de controlar o fluxo de fluídos, como derivados de petróleo, água, esgoto, etc. A válvula efetua o trabalho de abrir ou fechar o obturador de fluxo. Este estudo de caso se concentra na válvula.

Quando uma válvula falha de forma indesejada, esta pode interromper ou abrir total ou parcialmente o obturador, que pode causar grandes prejuízos no processo produtivo. Por exemplo, a interrupção de um gasoduto pode cessar o processo de refino e gerar prejuízos financeiros altíssimos, em função do tempo de parada. Outro exemplo, uma falha em um obturador pode interromper o funcionamento de uma adutora e o fornecimento de água para uma cidade por um determinado tempo.

Então, a válvula foi escolhida como estudo de caso por ser um equipamento importante no processo de transporte de derivados de petróleo, segundo a equipe de manutenção da empresa Transpetro, subsidiária da Petrobras.

Neste estudo de caso, foi adotada uma válvula movida eletricamente, em que o obturador é acionado por um atuador motorizado. O modelo escolhido é fabricado pela

empresa Coester Automação S.A.[11], e o modelo do atuador (motor) é o CSR25 que é acoplado a uma válvula do tipo gaveta.

O atuador é responsável por movimentar a haste da válvula, abrindo ou fechando o obturador. Na Figura 5.1, podem ser vistas as partes que compõem a válvula adotada nos experimentos.

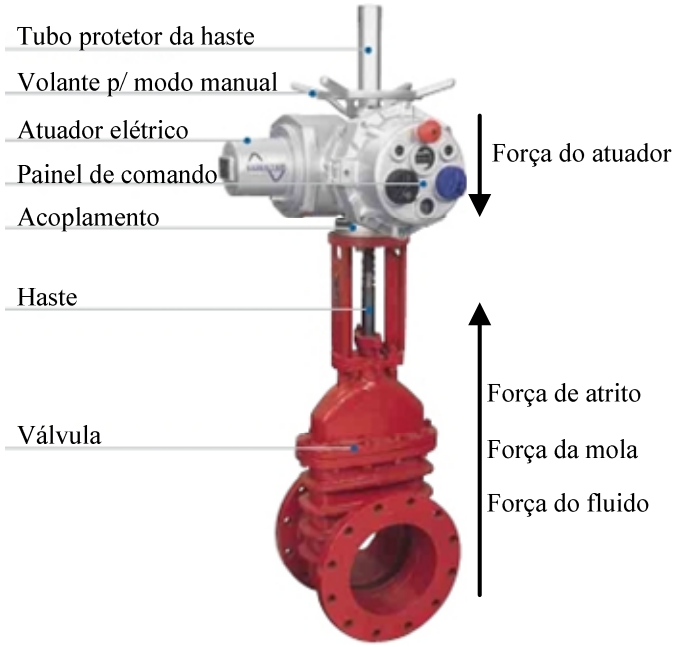



Figura 5.1: Visão geral da válvula e do atuador adotado nos experimentos.

No conjunto válvula e atuador, é possível monitorar o torque mecânico exercido pelo motor, durante a realização do movimento de abertura e/ou fechamento da válvula, e a posição que se encontra o obturador da válvula. Nesses experimentos serão adotadas ambas variáveis, *torque* e *posição*.

Já que os dados de histórico para um caso real de utilização do equipamento não estão disponíveis, então deve-se utilizar uma ferramenta para simulação do comportamento físico do conjunto atuador e válvula. Na próxima seção, será abordada a ferramenta de simulação e injeção de falhas.

## 5.4 Modelo para Simulação

Para desenvolver uma ferramenta de simulação de todos os parâmetros e comportamentos esperados para o conjunto atuador e válvula, necessita-se de um grande esforço de projeto. Para isso, é preciso conhecer as características e o comportamento físico de cada componente que forma o conjunto e as relações entre cada uma das partes.

A construção de modelos é baseada no conhecimento *a priori* dos princípios físicos que governam o comportamento do sistema. Na ferramenta de simulação, com base em modelagem física, o comportamento do sistema e os valores de saída são modelados como um sistema matemático.

---

[11] Site da empresa Coester Automação S.A. - http://www.coester.com.br/

Então, o modelo é inteiramente baseado no conhecimento detalhado das relações físicas e das características dos componentes que integram o sistema. No final obtêm-se um conjunto de equações matemáticas (equações diferenciais, sistemas algébricos, etc.), que será chamado de *modelo matemático*.

O modelo matemático é implementado como uma ferramenta de simulação, em que serão feitos os ajustes dos parâmetros e das variáveis que governam o modelo. Essa ferramenta apresentará uma saída de dados com os resultados esperados.

A construção do modelo matemático, para simulações deste estudo de caso, foi realizada por (GONÇALVES ET AL., 2007) (GONÇALVES ET AL., 2008) no Grupo de Pesquisa em Manutenção Inteligente, da Universidade Federal do Rio Grande do Sul. Em Anexo nesta dissertação está o artigo apresentando maiores detalhes a respeito das equações que governam a construção do modelo matemático.

Durante as pesquisas para elaboração do modelo matemático, foram estudadas, com detalhes, as relações físicas de construção entre as partes que formam o atuador e a válvula. O atuador é composto por um conjunto elétrico e mecânico. As principais partes (ver Figura 5.2) são:

- *Motor elétrico*: motor de indução trifásico do tipo gaiola de esquilo.
- *Redução mecânica*: sistema de transmissão mecânica por engrenagens.
- *Acoplamento*: interface do atuador com a haste rotativa da válvula.
- *Sensor de torque*: célula de carga eletrônica.
- *Sensor de posição e movimento*: potenciômetro.

A válvula é composta por um conjunto mecânico. As principais partes (ver parte inferior na Figura 5.1) são:

- *Acoplamento*: interface rotativa da haste com o atuador.
- *Haste*: sistema rotativo para movimentar obturador.
- *Obturador*: dispositivo que bloqueia o fluxo no canal de vazão.
- *Canal de vazão*: local onde passa o fluido, sendo a interface da válvula com a tubulação.

Como pode-se notar, a construção do conjunto atuador e válvula é feita para interligar todas essas partes entre si. Para isso, é necessário estudar todas as questões físicas dessa relação e construir um modelo matemático para simulá-las.

Conforme (GONÇALVES ET AL., 2008), durante o funcionamento do conjunto, existe um forte relacionamento entre as partes do conjunto: atuador, válvula, tubulação e fluido: uma relação entre a determinação da posição do obturador em relação à vazão do fluido pela tubulação; uma relação entre a transmissão do torque do motor e a redução mecânica, para movimentar o obturador até uma posição indicada que controlará a vazão pela tubulação; etc.

Durante o processo de transmissão do torque até o obturador, uma série de forças devem ser tratadas pelo modelo matemático. Essas forças influenciam diretamente no movimento de abertura ou fechamento do obturador da válvula. Na Figura 5.1 ao lado direito, são apresentadas as direções dessas forças.

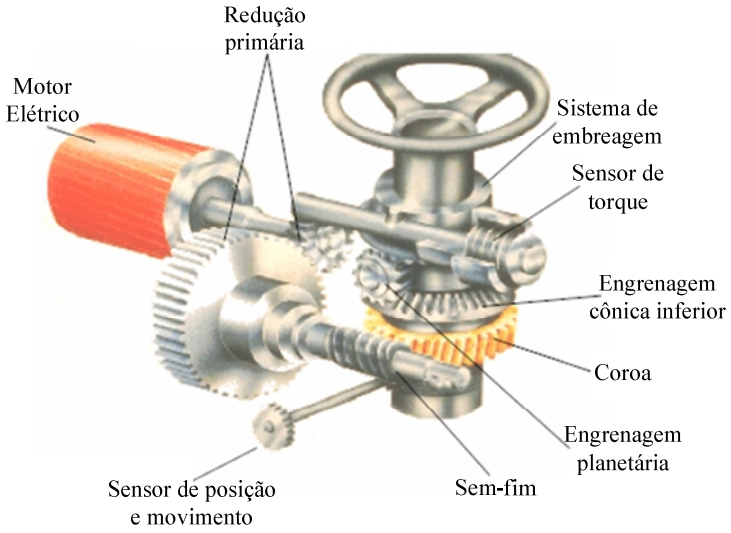



Figura 5.2: Principais partes do atuador elétrico.

A força exercida pelo motor elétrico do atuador deve vencer as demais contrárias. A força de atrito corresponde ao atrito existente entre a haste e a gaxeta (peça utilizada para vedação). A força da mola corresponde ao movimento do obturador até sua posição inicial, na ausência de energia elétrica. A força do fluído passa pela tubulação exercendo pressão sobre o obturador.

As saídas do modelo matemático devem ser as mesmas do sistema de controle do atuador, com sinal de torque e posição do obturador. A saída de torque do modelo é para simular os dados oriundos do *sensor de torque* (medida do torque de saída exercida pelo motor elétrico). A saída de posição corresponde a simular o *sensor de posição e movimento* (medida do percentual de movimento do obturador no canal de vazão).

### 5.4.1 Modelo Matemático

Durante o processo de modelagem matemática deste estudo de caso, (GONÇALVES ET AL., 2008) o dividiu em duas etapas fundamentais: definição e modelagem. O modelo matemático foi implementado de forma a ser um simulador computacional e suportar recursos para injeção de falhas. Para maiores detalhes consultar o Anexo.

A etapa de definição consiste na identificação dos fatores que influenciam na dinâmica de funcionamento e no comportamento do sistema. Além disso, implica em escolher adequadamente os princípios físicos, aplicar considerações e simplificações, além da escolha das variáveis que descrevem o sistema. O modelo matemático é constituído por um conjunto de equações diferenciais e algébricas não-lineares.

Para a modelagem, foram consideradas as seguintes partes do conjunto atuador, válvula e tubulação, que descrevem o comportamento do sistema:

- Motor elétrico.
- O sistema de redução mecânica (engrenagens).
- O fluxo do fluido que passa pela tubulação e válvula.
- A posição, velocidade e aceleração da haste.
- As forças envolvidas no movimento do motor e obturador.

A partir das considerações e separando cada caso em particular, foi desenvolvido um conjunto de equações para cada uma das partes isoladas do sistema. As equações são interligadas por variáveis comuns em cada parte, construindo um sistema de equações.

As equações e os métodos numéricos utilizados para solucionar as equações diferenciais e algébricas que descrevem as relações e os comportamentos físicos do conjunto atuador e válvula podem ser vistos com mais detalhes em (GONÇALVES ET AL., 2008).

É importante notar que esse modelo matemático ainda encontra-se em fase de validação. Necessita-se ainda definir alguns parâmetros importantes, que devem ser fornecidos pelo fabricante do equipamento, e validar utilizando experimentos para um caso real.

Na Figura 5.3, é apresentada a ferramenta de simulação desenvolvida por (GONÇALVES ET AL., 2008). É um simulador de dinâmica de sistemas não-lineares, utilizado para simular o modelo matemático do estudo de caso. Foi desenvolvido em MATLAB. Gera o conjunto de sinais necessários para realizar os experimentos para o Sistema de Manutenção Inteligente.

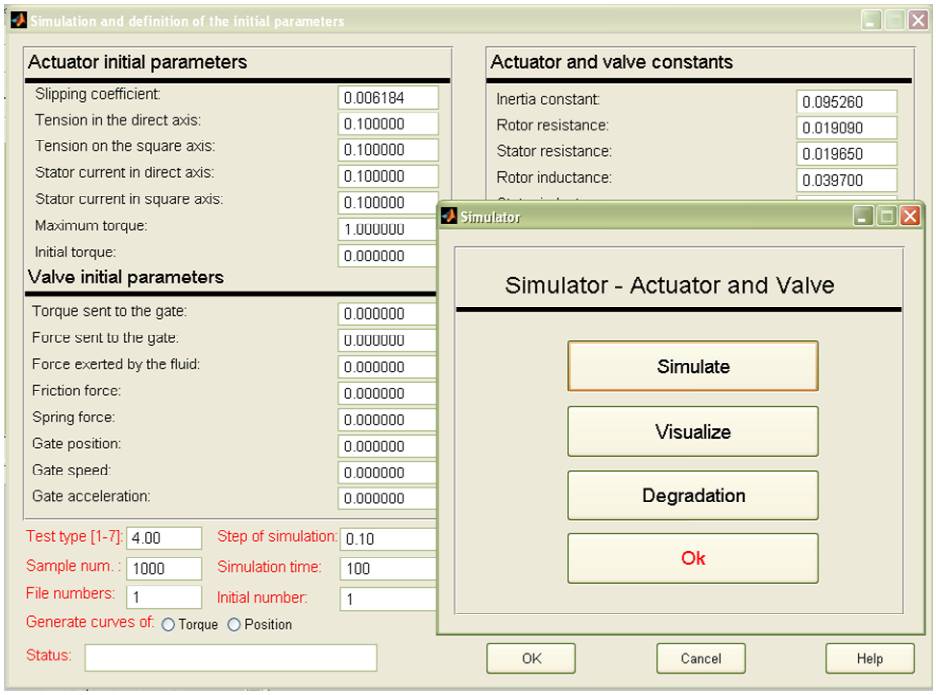



Figura 5.3: Ferramenta de simulação para experimentos no conjunto atuador e válvula.

## 5.5 Injeção de Falhas

Nesta seção, será apresentada a aplicação da ferramenta de simulação para experimentos de injeção de falhas no atuador elétrico, a fim de gerar conjunto de dados com históricos de comportamento do atuador. Esses dados são importantes para o treinamento e os testes das redes neurais do Sistema de Manutenção Inteligente.

Serão simuladas duas falhas que ocorrem no conjunto atuador e válvula. Essas falhas são: obstrução do obturador por acúmulo de sedimentos ou por materiais sólidos transportados pela tubulação até a válvula e desgastes originados nas engrenagens, causando folgas ao longo do tempo.

A simulação dessas falhas é importante para gerar o conjunto de dados de histórico que deve representar três tipos de comportamentos do atuador: normal, degradação e falha.

Os dados de histórico poderiam ter sido coletados em um ensaio real, em que se tenta imitar as situações normais, a de degradação e de falha. Entretanto, em alguns processos industriais, muitas das situações de falhas não podem ser ensaiadas, uma vez que há a necessidade de uma incisão física para forçar uma anormalidade (como quebra ou desgaste de peças) (VENKATASUBRAMANIAN; RENGASWAMY; KAVURI; ET AL., 2003). Nesse caso, quando se causa degradação ou falha, têm-se custos elevados de tempo e dinheiro, e isso inviabiliza a realização dos ensaios.

Uma opção mais viável é simular as diversas situações normais, de degradação e de falha por meio de um simulador utilizando um PC. No entanto, esta é aplicável somente quando a ferramenta de simulação está disponível. Neste estudo de caso, esta opção está à disposição e é será aplicada neste trabalho.

Como o objetivo deste trabalho é aplicar técnicas de tolerância a falhas de detecção, diagnóstico e predição, necessita-se *a priori* do conjunto de dados de histórico de comportamento. Para gerar o conjunto de dados pelo simulador, foi utilizado, como saída, o sinal de *torque* do atuador e a *posição do obturador* da válvula.

Para fins de validação da técnica de tolerância a falhas proposta neste trabalho, foi simulada uma série de experimentos para a geração de amostras de dados, formando um conjunto de dados de histórico.

Também foi definido o modelo do atuador CSR25, fabricado pela empresa Coester, que tem um motor elétrico de 5HP, fabricado pela empresa WEG, e com torque nominal de 250Nm. Como o atuador tem uma proteção interna contra o aumento repentino do torque, então foi definido como valor de sobre-torque 275Nm e após isso ele é desligado. A válvula adotada contém um obturador do tipo gaveta.

Com esses parâmetros definidos, já é possível simular os comportamentos normal, de degradação e de falhas que podem ser observados no conjunto atuador e válvula.

Por exemplo, as causas de algumas das falhas que podem ser simuladas são o aumento gradual do torque ou a dificuldade em movimentar o obturador por estar mais rígido e bloqueado. Essas falhas influenciam diretamente o valor do torque e a posição do obturador. Tudo isso é feito por meio de ajustes em parâmetros no modelo matemático do simulador.

Para realizar os experimentos de abertura e o fechamento do obturador, foram simuladas falhas em dois parâmetros do modelo matemático. A injeção de falhas foi feita mediante modificação gradual dos parâmetros nas equações do modelo matemático. Deste modo pode-se simular a degradação nas partes do sistema até atingir o teto de sobre-torque em 275Nm, considerado como falha. Os dois parâmetros são:

- $K_2$: corresponde a falhas observadas no segundo sistema de redução cinemática. Trata-se da quebra da engrenagem sem-fim. Por ser uma peça frágil do sistema de redução, pode sofre degradação ou até mesmo romper-se quando receber um esforço mecânico adicional, causando um sobre-torque.
- $K_m$: corresponde à degradação da elasticidade da mola na válvula. Esta, ao longo do tempo, vai se degradando, perdendo suas características iniciais e alterando sua ação, modificando a posição do obturador.

A modificação gradual dos parâmetros utilizados para injeção de foi foram feita até alcançar o teto de sobre-torque conforme a Tabela 5.1:

Tabela 5.1: Configuração dos parâmetros para injeção de falhas.

| **Parâmetro** | **Faixa** | **Taxa** |
|---|---|---|
| $K_2$ | 11,00 – 12,00 | 0,0033 |
| $K_m$ | 4,215 – 5,215 | 0,0033 |

A modificação gradual desses parâmetros visa simular uma situação de degradação progressiva dos componentes até a ocorrência da falha. Por exemplo, quando o parâmetro $K_2$ está na faixa de 11,00 (250Nm) está em situação normal de operação, mas quando começa a desviar-se em direção a 12,00, inicia-se um processo de degradação do componente. Sempre que este atingir o teto 12,00 (275Nm), configura-se situação de sobre-torque, o componente já sofreu muito esforço extra e é considerado comportamento de falha. Observa-se que a ocorrência dessa falha é em razão do desgaste natural das engrenagens, simuladas nesse experimento. Na Figura 5.4, é apresentada a saída do simulador para este caso de teste.

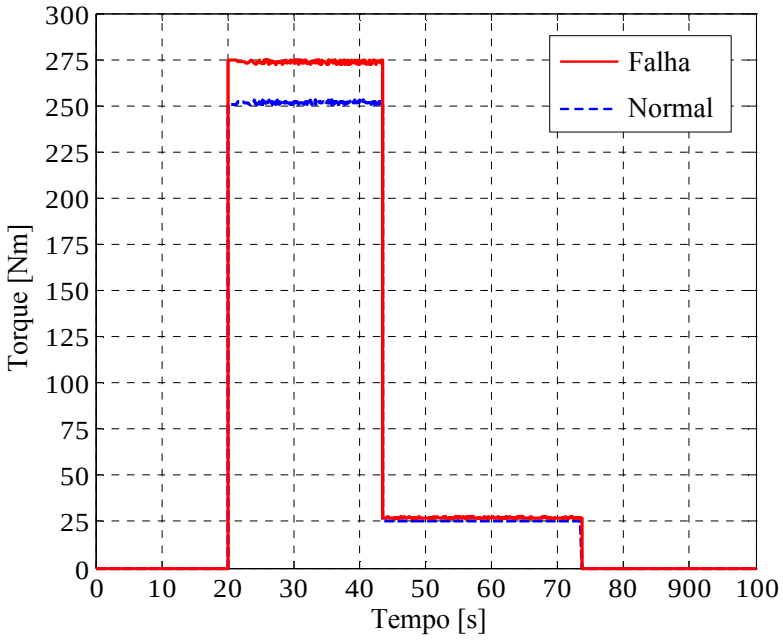



Figura 5.4: Saída do simulador para o sinal de torque e falha injetada no parâmetro $K_2$.

Pode-se também simular, por exemplo, a perda da elasticidade na mola devido ao desgaste natural da peça. Isso pode ser simulado pelo parâmetro $K_m$ sofrendo uma mudança gradual (alterando-se de 4,215 até 5,215). O efeito disso é que a mola se tornará mais rígida e por consequência será exigido mais torque do motor, que não conseguirá movimentar o obturador até sua posição inicial, deixando o canal de vazão parcialmente aberto ou fechado, por consequência, em estado falha. Na Figura 5.5, é apresentada a saída do simulador para este caso de teste.

Um grande exemplo para esses eventos é o sistema de transporte de água de adutoras ou tubulações de esgoto, que é muito comum no saneamento básico das cidades. Nesse sistema, pode-se observar uma situação de falha que é normalmente

causada pelo acúmulo de sedimentos (ou resíduos) na base da válvula. Esses resíduos podem ser areia, terra ou até pequenas partículas (como galhos ou pedras), que decantam no fundo das tubulações e podem se concentrar nas conexões das válvulas. Nesse caso, para que a válvula se feche por completo (sem vazamentos), é necessário que o atuador forneça um torque maior, que, muitas vezes, pode atingir o sobre-torque e ir degradando as engrenagens ao longo do tempo devido ao grande esforço exercido.

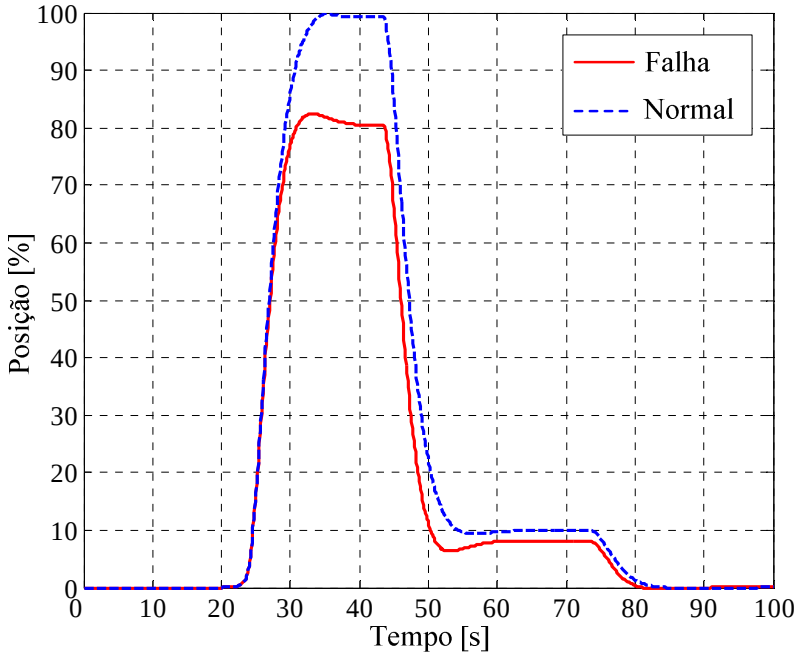



Figura 5.5: Saída do simulador para o sinal de posição do obturador e falha injetada no parâmetro $K_m$.

Com a simulação das falhas para os parâmetros $K_2$ e $K_m$ já é possível gerar todos os conjuntos de dados de histórico necessários para os experimentos. Foram gerados amostras de dados para treinamento e para testes.

Para treinar as redes neurais com os dados gerados, primeiro é preciso aplicá-los a uma ferramenta de processamento de sinais. Os dados foram inseridos na ferramenta *Watchdog Agent* (DJURDJANOVIC ET AL., 2003) que aplica uma fusão nos sinais (torque e posição) e extrai as características deste sinal, a partir da análise da energia, obtido pela Transformada *Wavelet Packet* (QIU, HAI ET AL., 2006). Após o pré-processamento, os dados estão prontos para os experimentos no Sistema de Manutenção Inteligente.

## 5.6 Treinamento das Redes Neurais

O treinamento é a parte mais importante do preparo do SOM para aprender sobre o comportamento do sistema. Nesta seção, será apresentado como utilizar os dados gerados pelo simulador de injeção de falhas para treinamento das redes neurais que serão utilizadas durante os experimentos em *software* e *hardware*.

### 5.6.1 Treinamento para experimento em *software*

Nesta seção, será apresentado como treinar as redes neurais para realizar os experimentos em *software*, utilizando os dados gerados pelo simulador.

Todos os experimentos em *software* são feitos com base em simulações executadas no PC. Essas simulações incluem as etapas de treinamentos e testes.

Para auxiliar nos experimentos, foi adotado o ambiente MATLAB 7 para executar o SOM Toolbox (VESANTO, J. ET AL., 1999) (VESANTO, J. ET AL., 2000). Além disso, foi adotada uma ferramenta para experimentos de tolerância a falhas, baseado no SOM Toolbox, que será utilizada para treinamentos, testes e coleta de resultados.

Essa ferramenta faz o carregamento de dados de entrada para treinamento e geração de resultados extraídos do SOM. É focada nas etapas de detecção, diagnóstico e predição de falhas, como nos algoritmos apresentados nas seções 4.2.1 a 4.2.4.

Em virtude de o SOM conter diversos parâmetros para configuração do algoritmo de treinamento, foram adotadas as mesmas configurações utilizadas para treinamento do protótipo de *hardware* do SOM, conforme visto na seção 4.4.2, e os parâmetros adotados conforme Tabela 4.1.

Na Figura 5.6, é apresentado o modelo de treinamento para esses experimentos.

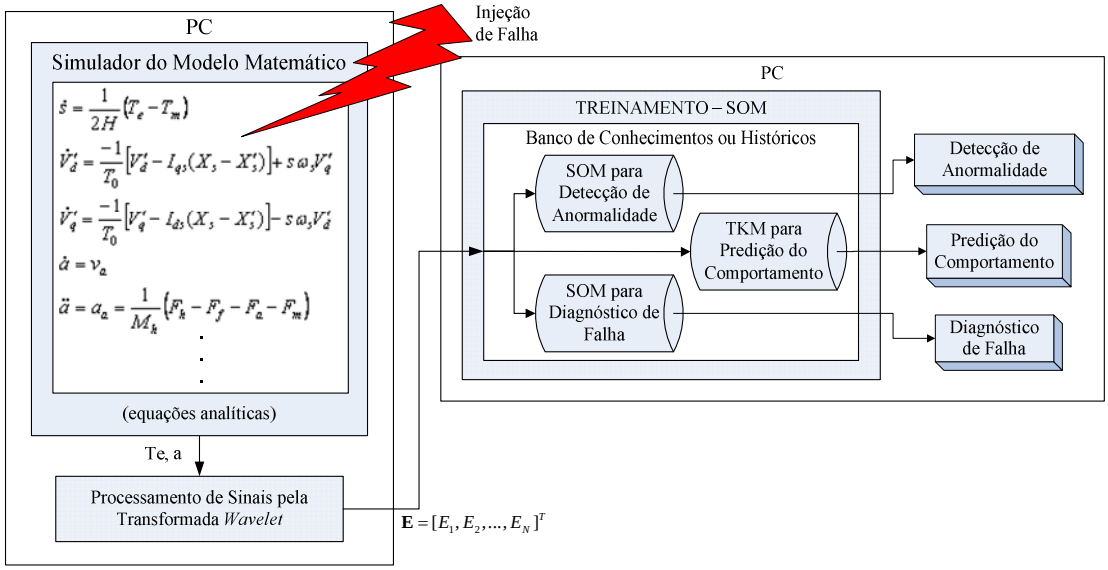



Figura 5.6: Modelo de treinamento para experimentos em *software*.

O modelo de treinamento é baseado nas saídas geradas pelo Simulador do Modelo Matemático (GONÇALVES ET AL., 2008). O simulador realiza a tarefa de injetar falhas em partes específicas no modelo do atuador (limitando-se ao domínio das falhas mais críticas), possibilitando gerar um conjunto de dados rico em informações de comportamento do atuador. Esses dados são utilizados para treinar o SOM, a fim de aprender o comportamento das partes onde as falhas foram injetadas.

Como são utilizadas três redes neurais, cada amostra de dados necessários para o treinamento deve ser gerada pelo Simulador do Modelo Matemático. Estes dados seguem o modelo apresentado na seção 4.2, por meio das Figuras 4.2 e 4.3.

Os sinais gerados são apresentados à ferramenta de Processamento de Sinais, que retorna, como saída, um vetor com a energia do sinal $\boldsymbol{E}=[E_1, E_2, ..., E_N]$, onde $E_N$ é a

energia de cada banda de frequência. Esses dados são aprendidos e armazenados pelas redes neurais do Banco de Conhecimentos ou Históricos.

Após o treinamento, os algoritmos de Detecção de Anormalidade, Diagnóstico de Falha e Predição do Comportamento utilizarão as redes neurais já treinadas para aplicarem as técnicas de tolerância a falhas no atuador elétrico.

A partir disso, os dados de testes já podem ser apresentados para simular o funcionamento do Sistema de Manutenção Inteligente.

### 5.6.2 Treinamento para experimento no protótipo em *hardware*

Nesta seção, será apresentado como treinar as redes neurais para realizar os experimentos no sistema embarcado (*hardware*) usando as amostras geradas pelo simulador. Espera-se que, com esses resultados, os experimentos atinjam os mesmos objetivos alcançados em *software*, somente para os casos de detecção e diagnóstico de falhas.

O treinamento para os experimentos em *hardware* reutiliza as mesmas redes neurais treinadas nos experimentos em *software*. Isso é devido ao *hardware* do SOM não suportar o treinamento das redes neurais internas.

Para auxiliar nos experimentos, é necessário utilizar as ferramentas ISE e XPS, da Xilinx, para programação do sistema embarcado e da placa de desenvolvimento do protótipo.

Na Figura 5.7, é apresentado o modelo de treinamento para os experimentos.

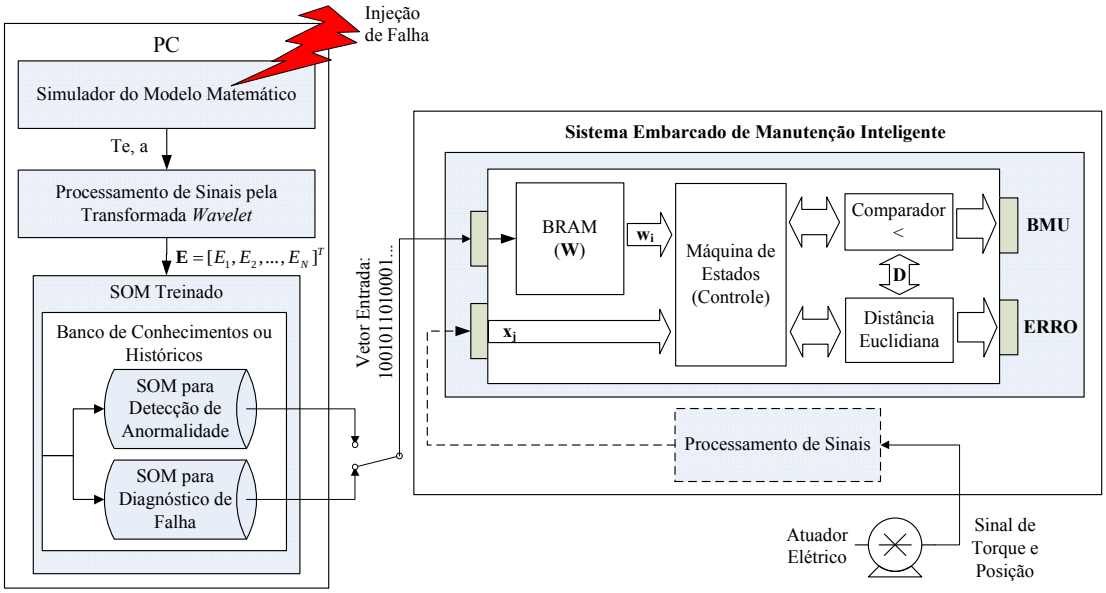



Figura 5.7: Modelo de treinamento para experimentos no sistema embarcado.

O modelo de treinamento também é baseado nas saídas geradas pelo Simulador do Modelo Matemático, que, nesse caso, reutilizará as redes neurais já treinadas pelo experimento em *software*. Como o *hardware* do SOM funciona para apenas um dos casos, deve-se selecionar qual rede neural utilizar, isto é, ou para detecção, ou para diagnóstico em cada um dos experimentos.

Os pesos sinápticos dos neurônios do SOM devem ser exportados, de forma que possam ser armazenados na memória interna (BRAM) do sistema embarcado. Desse modo, estes são convertidos por meio de um *script* do padrão numérico decimal para o binário, em que cada elemento do vetor de pesos sinápticos é convertido para a representação binária, seguindo a formatação numérica de Ponto Flutuante IEEE 754.

A etapa de transferência dos vetores do SOM já treinado, para o sistema embarcado, é realizada antes da síntese pela ferramenta ISE. Durante a síntese, é criado o componente BRAM, que armazena os dados referentes ao SOM treinado.

A próxima etapa é sintetizar todo o sistema embarcado conforme visto na seção 4.5.1. Neste, é criada toda a plataforma necessária do Sistema Embarcado de Manutenção inteligente já com o SOM adequado para cada caso.

Após a preparação do sistema embarcado, os algoritmos de Detecção de Anormalidade ou Diagnóstico de Falha utilizarão o SOM treinado para aplicar as técnicas de tolerância a falhas no atuador elétrico. A partir daí a execução de tarefas de testes é realizada por uma aplicação em *software* executada na plataforma.

Com o sistema embarcado definido, o *software* de aplicação com a etapa de teste do Sistema de Manutenção Inteligente precisa ser carregado. Essa aplicação fará a leitura dos sinais de torque coletados no atuador elétrico, realizará o processamento de sinais e preparará os vetores de entrada. Esses vetores são apresentados na entrada do *hardware* do SOM, que realizará a tarefa de detecção ou diagnóstico das falhas em tempo real.

Para gerar os resultados do experimento, as saídas do *hardware* do SOM são armazenadas na memória disponível no sistema embarcado. Esses dados são enviados para o PC, e serão processados pelo MATLAB para converter em gráficos de erro de quantização ou U-Matrix. Essa parte é necessária, pois o sistema embarcado ainda não tem uma IHC (Interface Humano-Computador).

## 5.7 Resultados para experimentos em *software*

Os resultados visam apresentar alguns casos de teste, para comprovar a eficácia do SOM em detectar, diagnosticar e prever as falhas no atuador elétrico.

Eles são baseados nos algoritmos apresentados na seção 4.2, em que para cada uma das etapas obtém-se um resultado diferente. Para a Detecção de Anormalidades, ver algoritmo na seção 4.2.2, para Diagnóstico de Falhas, ver algoritmo na seção 4.2.3 e para Predição de Falhas, ver algoritmo na seção 4.2.4.

Os experimentos foram realizados analisando as falhas $K_2$ (referente a falhas na engrenagem do sem-fim) e $K_m$ (referente à degradação na mola) e utilizando os dados de testes gerados pelo Simulador do Modelo Matemático.

### 5.7.1 Detecção de Anormalidades

Os gráficos de resultados para o Erro de Quantização ($E_q$), utilizado para detectar anormalidades, são apresentados nas Figuras 5.8 e 5.9. Foram simulados o funcionamento, no primeiro caso, para o componente $K_2$ e, no segundo caso, para o componente $K_m$.

Nestas figuras, o Erro de Quantização ($E_q$) foi apresentado em relação aos Ciclos de Operação, ou seja, corresponde à aquisição de um conjunto de amostras para uma determinada situação de comportamento ao longo do tempo.

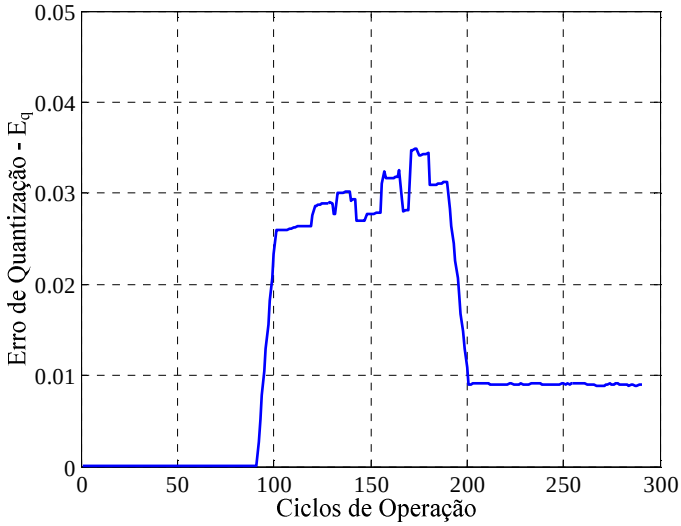



Figura 5.8: Resultado de Detecção de Anormalidades para falha em $K_2$.

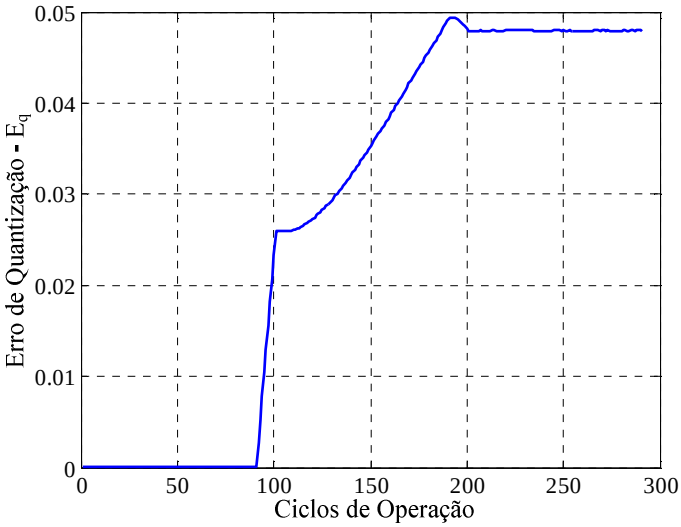



Figura 5.9: Resultado de Detecção de Anormalidades para falha em $K_m$.

As amostras utilizadas para teste foram definidas na seguinte ordem:

- Amostras de 1 – 100: comportamento normal.
- Amostras de 101 – 200: comportamento em degradação.
- Amostras de 201 – 300: comportamento de falha.

Durante as 100 primeiras amostras, por serem simuladas, sabe-se, a princípio, que são referentes ao comportamento normal. Desse modo, o algoritmo de Detecção de Anormalidades conseguiu identificar corretamente as anormalidades testadas no sistema que ocorreram a partir da 101ª amostra.

### 5.7.2 Diagnóstico de Falhas

O resultado do diagnóstico de falhas é visualizado pela U-Matrix, que revela a classificação das amostras de acordo com a classe a que pertence. Na Figura 5.10, é

apresentada a U-Matrix do SOM treinado, revelando as regiões de classificação para cada modo de comportamento que possibilita o diagnóstico destes.

Foram simulados comportamentos de teste para os componentes $K_2$ e $K_m$. As amostras utilizadas para teste em cada componente foram definidas na seguinte ordem:

- Amostras de 1 – 50: comportamento normal.
- Amostras de 51 – 100: comportamento em degradação.
- Amostras de 101 – 150: comportamento de falha.

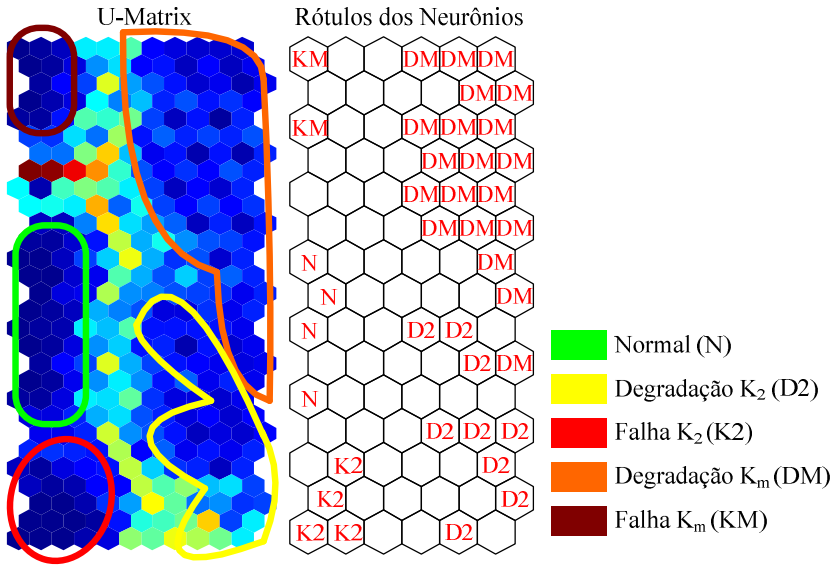



Figura 5.10: Visualização do SOM treinado para diagnóstico dos componentes $K_2$.e $K_m$.

Na Figura 5.11, são apresentados os resultados de diagnóstico. Nesse caso, o SOM revela a classificação dos modos de comportamento dos dados de teste, projetando-os dentro dos agrupamentos a qual pertence.

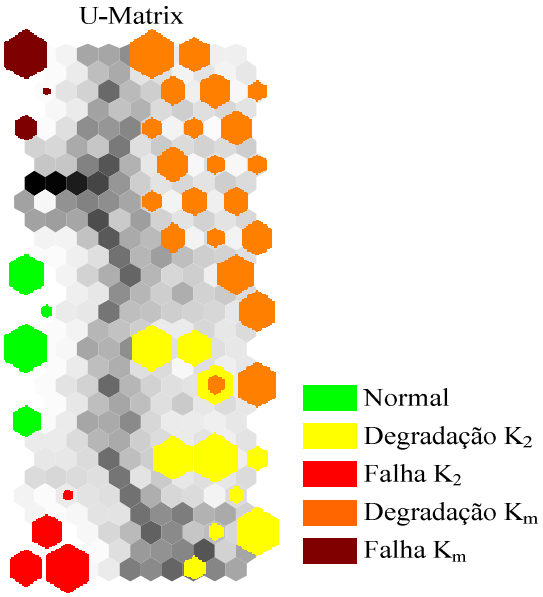



Figura 5.11: Resultado de diagnóstico de falhas para os dados de testes nos componentes $K_2$.e $K_m$.

Nota-se pela análise visual do resultado que, em geral, a classificação dos modos de comportamento para $K_2$ e $K_m$ foram acertadas corretamente pelo SOM. Desse modo, o SOM foi capaz de identificar em qual modo de comportamento pertence o sinal de entrada.

Entretanto, durante o diagnóstico, o SOM também está sujeito à ocorrência de erros na classificação. Isso pode ocorrer quando são utilizados poucos dados para treinamento, não refletindo todos os casos necessários para generalizar o problema. Outra variável que pode influenciar a classificação é a presença de ruído nos sinais.

Quando se utiliza apenas os rótulos dos neurônios (Figura 5.10 da direita) como técnica para classificar, em alguns casos, a classificação pode apresentar erros. No entanto, quando levam em consideração as regiões de comportamentos comuns (Figura 5.10 da esquerda), os erros diminuem.

No experimento da Figura 5.11, foram utilizadas 150 amostras em cada componente. Quando se utiliza os rótulos do SOM para classificação, ocorre 1 erro para $K_2$ (menor que 1%) e 4 erros para $K_m$ (menor que 3%).

Em ambos os experimentos, quando se utiliza as regiões de comportamentos para classificação, todos os casos acertam a classificação em 100%. Isso ocorre porque alguns neurônios vizinhos não recebem um rótulo, mas observando a formação das regiões, pode-se notar pela similaridade dos neurônios que estes também classificam os dados para a mesma classe.

### 5.7.3 Predição e Monitoramento de Falhas

O resultado para predição e monitoramento de falhas é apresentado pela projeção de uma trajetória na U-Matrix. Por meio dela, é revelado visualmente o caminho percorrido pelos neurônios vencedores, proporcionando monitorar todo o ciclo de vida do sistema em análise.

Na Figura 5.12, é apresentada a U-Matrix para um TKM treinado, que é utilizado para monitorar a trajetória dos modos de comportamento ao longo do tempo decorrido.

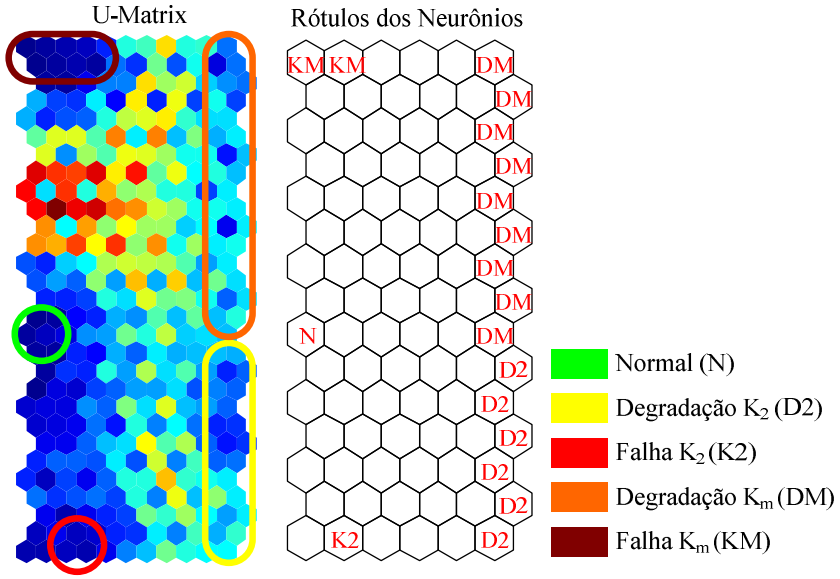



Figura 5.12: Visualização do TKM treinado para monitorar os componentes $K_2$.e $K_m$.

Foram utilizados os mesmos dados de teste do experimento da seção anterior para simular o comportamento dos componentes $K_2$ e $K_m$, ao longo do tempo. A partir de agora, os ciclos de operação serão ordenados na ordem de chegada das amostras e serão chamados de série temporal.

Na série temporal utilizada para teste de cada componente, a ordem de tempo adotada é referente a um conjunto de dados adquiridos que representa uma determinada situação de comportamento ocorrida no tempo. A série foi definida na seguinte ordem:

- Amostras de 1 – 50: comportamento normal.
- Amostras de 51 – 100: comportamento em degradação.
- Amostras de 101 – 150: comportamento de falha.

Os parâmetros de configuração do TKM foram ajustados do seguinte modo: as dimensões do mapa seguem o mesmo padrão do SOM utilizado anteriormente, o histórico de ativações ($a_i$) foi ajustado para *5* amostras e a profundidade da memória ($\lambda$) para *0,07*. Estes valores dos parâmetros foram definidos empiricamente.

Nas Figuras 5.13 e 5.14, são apresentados os resultados para predição e monitoramento do comportamento para os componentes $K_2$ e $K_m$.

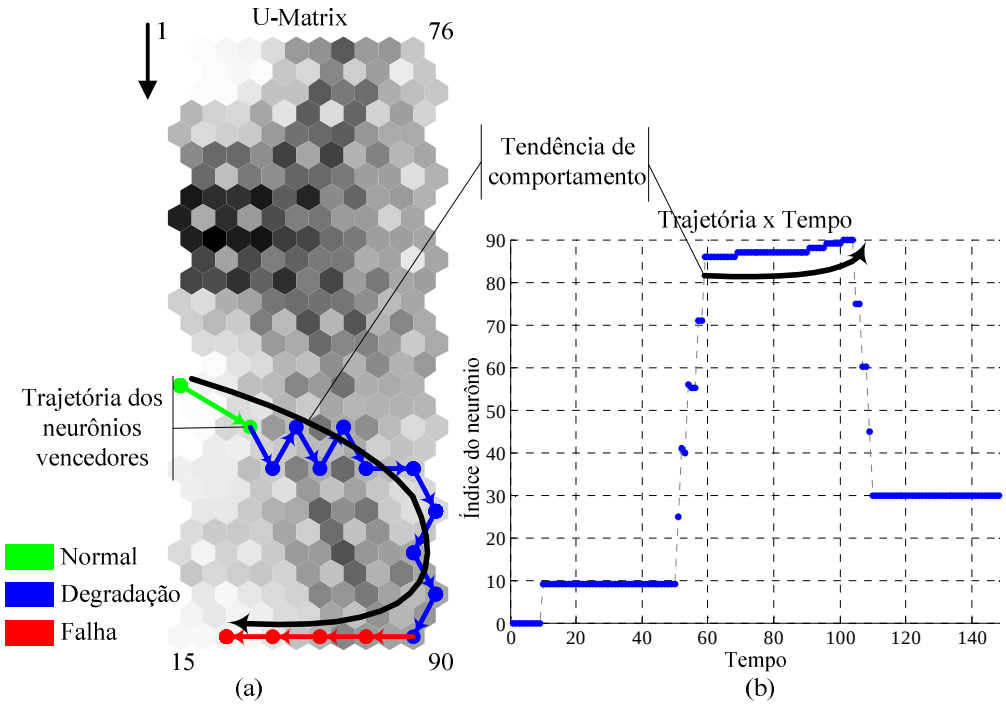



Figura 5.13: Visualização da trajetória para TKM e a direção da tendência para caso de teste **$K_2$**, sendo (a) Caminho do trajeto percorrido pelos neurônios vencedores e (b) Relação dos neurônios vencedores para cada instante de tempo.

Nesses resultados, nota-se a correlação espaço-temporal entre os dados da série de teste. A trajetória percorrida revela a direção da tendência nos comportamentos observados e que estão sendo mapeados nos agrupamentos. Também é possível identificar um padrão no comportamento da trajetória no período de tempo.

Além disso, no gráfico da relação dos neurônios vencedores com o tempo, pode-se observar o instante em que ocorreram as mudanças de comportamento e identificar uma tendência na classificação da série temporal. Quando ocorre uma mudança de um

neurônio vencedor para outro do mesmo agrupamento, significa que a série de testes já sofreu modificações em relação a seus valores de histórico.

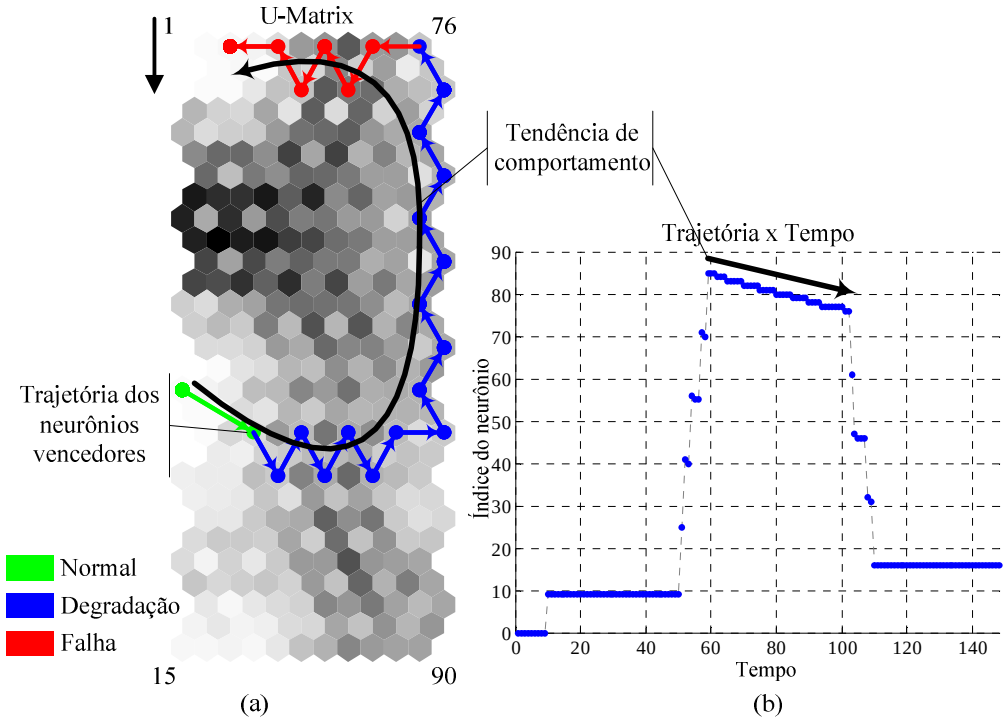



Figura 5.14: Visualização da trajetória para TKM e a direção da tendência para caso de teste $\mathbf{K_m}$, sendo (a) Caminho do trajeto percorrido pelos neurônios vencedores e (b) Relação dos neurônios vencedores para cada instante de tempo.

A informação importante para a predição de comportamento que pode ser obtida dos resultados vistos, nas Figuras 5.13 e 5.14, são os instantes finais antes da mudança para o comportamento em falha.

Para o caso do componente $K_2$ (Figuras 5.13), o instante de tempo em que se revela a tendência para falha é somente a partir da amostra *91*, quando já estiver muito próximo da manifestação da falha, mas, ainda sim, existe um tempo de tolerância até a manifestação da falha.

Para o caso do componente $K_m$ (Figuras 5.14), a tendência até a falha é mais definida do que em $K_2$, pois os neurônios vencedores seguem um padrão constante de mudança de comportamento. A partir da amostra *85*, começa a se concretizar a transição para um comportamento de falha, em que a partir da amostra *100* ocorre realmente a manifestação da falha.

O importante é identificar em qual neurônio ocorre a mudança de um estado de degradação para a falha, deste modo pode-se calcular o tempo restante até a manifestação da falha.

Para ambos os casos, pode-se emitir um sinal de alarme para a equipe de manutenção quando a série temporal de testes for classificada para alguns desses neurônios vencedores. Esses neurônios podem ser considerados como “críticos”, pois estão situados em regiões onde ocorre a transição para outra situação de comportamento, ou seja, uma situação de possível falha.

## 5.8 Resultados para Experimento no Protótipo em *Hardware*

Para o experimento de Detecção de Anormalidade e Diagnóstico de Falhas em *hardware* foi utilizado um protótipo implementado em uma placa de desenvolvimento (XILINX, 2005b) e, para gerar resultados, foram executados os *softwares* de testes sobre o sistema operacional do sistema embarcado (WILLIAMS; PETALOGIX, 2008).

Desse modo, os experimentos objetivam utilizar o *hardware* do SOM, que armazena na memória interna do *som_core* a rede neural treinada. Os resultados foram coletados na placa de experimentos e enviados para o PC, onde são processados pelo MATLAB, a fim de gerar os gráficos do erro de quantização e apresentação da U-Matrix para avaliações.

Também foram realizados experimentos comparativos entre os resultados obtidos pelo *hardware* do SOM em relação aos experimentos anteriores em *software*. Isso é importante para validar o correto funcionamento do projeto do sistema embarcado.

Para o experimento de Detecção de Anormalidades, foram utilizados os mesmos dados para testes do experimento em *software,* além de utilizar a mesma rede neural treinada. Os resultados foram praticamente os mesmos quando comparados aos do *software*.

Para avaliar o quanto foram próximos os resultados obtidos em *hardware* e *software*, foi calculada a diferença entre os dois e calculado o valor de máxima e mínima diferença. Na Tabela 5.2, são apresentados os resultados da diferença:

Tabela 5.2: Diferença da máxima amostra entre os resultados de *hardware* e *software* para Detecção de Anormalidades.

| **Parâmetro** | **Diferença** |
|---|---|
| $K_2$ | $5{,}9332.10^{-7}$ |
| $K_m$ | $5{,}9332.10^{-7}$ |

Devido aos resultados serem muito próximos entre si, a diferença máxima entre os erros de quantização é $1.10^{-7}$. Isso comprova que os resultados obtidos pelo *hardware* do SOM estão corretos em relação às simulações em *software*, apresentando valores praticamente iguais.

Para o experimento de Diagnóstico de Falhas, também foram utilizados os mesmos dados dos testes do experimento em *software*, além de utilizar a mesma rede neural treinada. Os resultados foram os mesmos quando comparados aos do *software*.

Foram comparadas as amostras de cada ciclo de operação, do *hardware* em relação ao *software*, verificando se os neurônios vencedores (BMU) obtidos foram os mesmos. Em 100% dos casos, o *hardware* do SOM classificou corretamente as amostras de teste, apresentando os mesmos resultados do *software*.

Em função de os resultados serem iguais entre o *hardware* e *software*, não serão apresentados aqui os resultados de visualização obtidos pelo sistema embarcado (gráficos para Erro de Quantização e a U-Matrix).

Como no sistema embarcado foi instalado um sistema operacional, este tem suporte a uma interface de entrada e saída padrão, que pode ser visualizada pelo *software* HiperTerminal (programa do Windows que interpreta a interfaces de comunicação com

o sistema operacional embarcado), por meio da porta RS-232 conectado ao sistema embarcado.

Na Figura 5.15, é apresentado um *screenshot*, que mostra a ferramenta XPS da Xilinx para programar o FPGA e as mensagens, durante o *boot* do PetaLinux, vistas na tela do HyperTerminal.

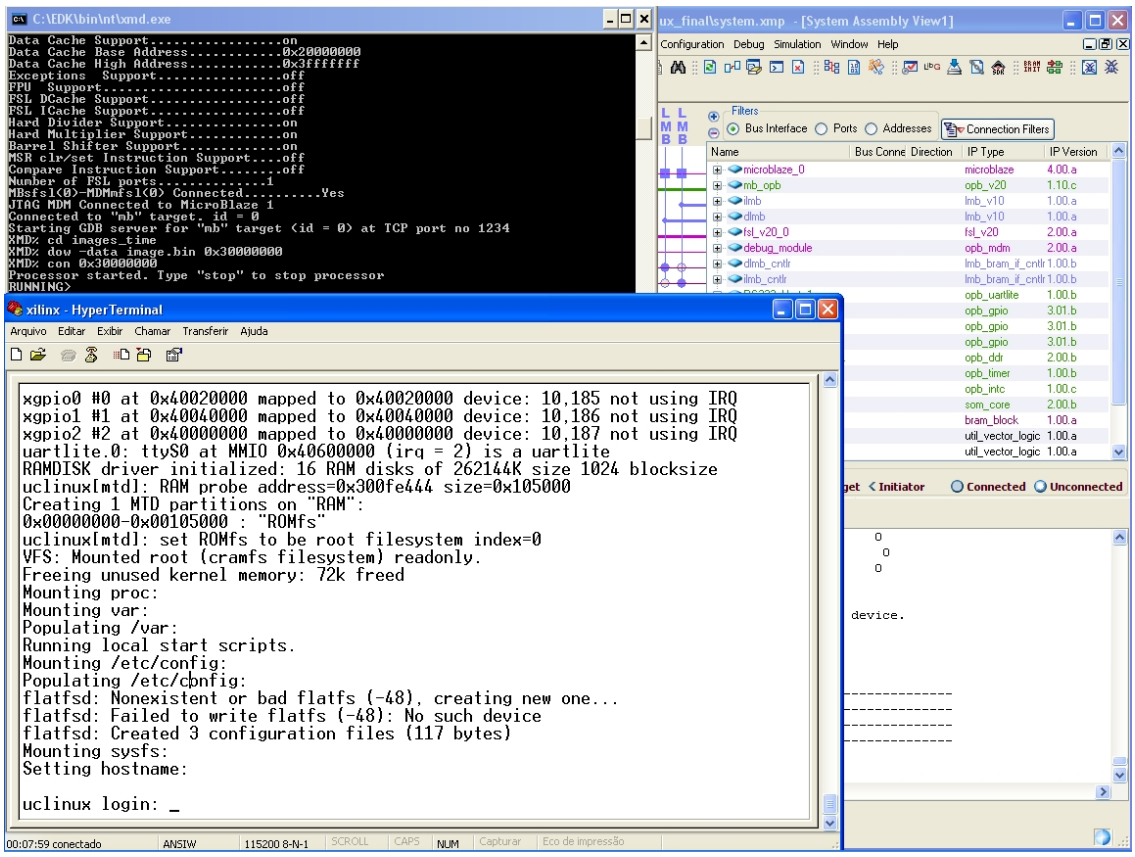



Figura 5.15: *Screenshot* da tela do PC mostrando a ferramenta XPS da Xilinx e as mensagens de *boot* no *HyperTerminal*.

Na Figura 5.16, é apresentada a execução de um programa de teste na plataforma do sistema embarcado. Esse programa realiza execuções infinitas, apresentando entradas de dados para teste no dispositivo do SOM até ser interrompido pelo usuário. Nessa figura, também são apresentados os resultados coletados no dispositivo, mostrando o índice do neurônio vencedor (BMU) e o valor do erro de quantização (ERRO).

Na Figura 5.17, é apresentada a execução do programa utilizado para gerar os resultados avaliados neste experimento. Esse programa realiza a carga de todos os vetores de entrada (obtidos pelo simulador), apresenta-os ao dispositivo do *hardware* do SOM e armazena todos os resultados (BMU e ERRO) em um arquivo texto na memória do sistema. Nessa figura, é apresentada a execução dos dois programas, um para detecção e outro para diagnóstico, e também apresenta o tempo de execução para cada caso.

## 5.9 Resumo do Capítulo

Neste capítulo, foram abordadas as etapas necessárias para realização dos experimentos para a implementação do SOM aplicado no Sistema de Manutenção

Inteligente de atuadores elétricos. Os casos de teste desses experimentos visam simular o funcionamento para a abordagem em *software* e *hardware*.

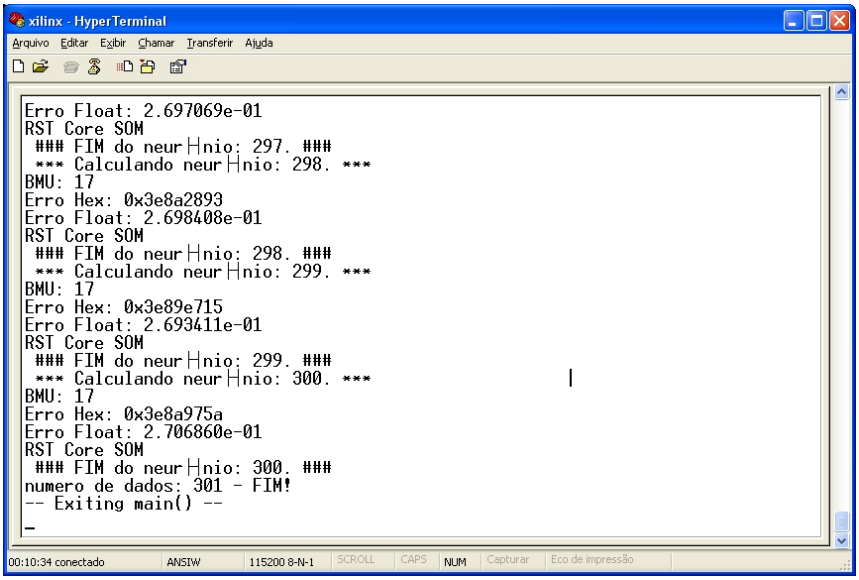



Figura 5.16: *Screenshot* do HyperTerminal mostrando a execução de um programa de teste e valores de saídas do dispositivo coletado do *som_core*.

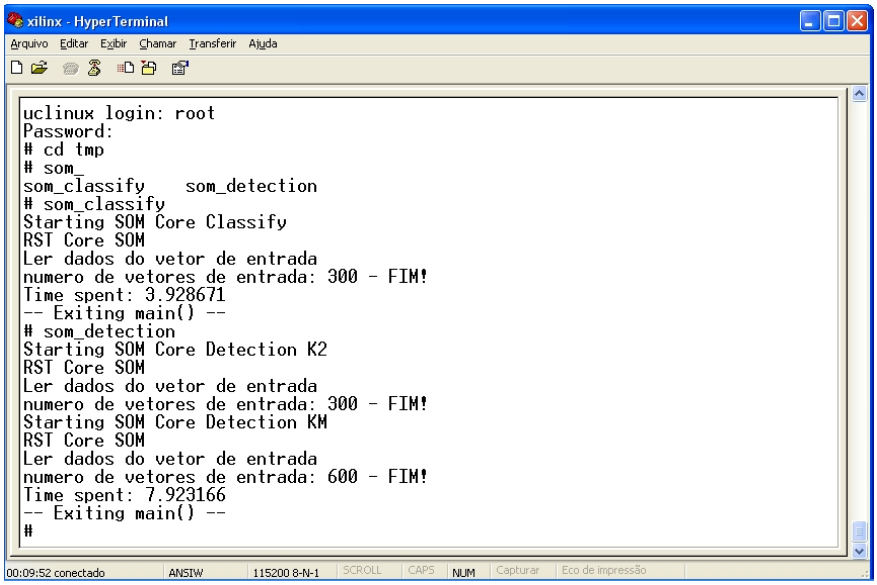



Figura 5.17: *Screenshot* do *HyperTerminal* mostrando a execução do programa utilizado para gerar os resultados avaliados neste experimento.

O objetivo fundamental do capítulo foi comprovar que o SOM é uma ferramenta importante na implementação do Sistema de Manutenção Inteligente, o que foi provado pelos resultados obtidos.

Foi definida uma metodologia para a realização dos experimentos em que são apresentados os passos necessários para simular tanto em *software* como em *hardware*.

Um estudo de caso foi definido e apresentado para ser o ambiente de simulação para o Sistema de Manutenção Inteligente, cuja aplicação situa-se nas válvulas e nos atuadores utilizados em tubulações para transporte de derivados de petróleo. Foram apresentadas as principais características de funcionamento do conjunto atuador e

válvula. Foram também definidas as variáveis de interesse a serem utilizadas nos experimentos, o *torque* exercido pelo motor do atuador e a *posição* do obturador.

Já que não existe um banco de dados com informações históricas de funcionamento do atuador e válvula, foi necessário utilizar uma ferramenta para simular as condições de operação e gerar amostras de dados para treinamentos e testes. A ferramenta foi desenvolvida pelo Grupo de Pesquisa de Manutenção Inteligente da UFRGS.

Como a ferramenta de simulação está disponível, foi apresentada a técnica para injeção de falhas no conjunto atuador e válvula, que consiste em mudar gradualmente os parâmetros no modelo matemático, a fim de simular mudanças no comportamento físico do atuador. Nesse caso, foi definido simular falhas no sistema de engrenagens (sem-fim) e na abertura e no fechamento do obturador (mola).

Como as variáveis definidas para coleta de sinais do atuador são o torque exercido pelo motor e a posição do obturador, foram definidos valores para representar a condição normal de operação, e valores-teto que representem condição de falha. Com esses sinais disponíveis, torna-se possível a simulação de comportamentos normais, de degradação (mudança gradual de normal até a falha) e de falhas, que foram aplicados como entradas no SOM.

Com os dados históricos prontos, foi apresentada a metodologia para o treinamento do SOM nos casos de simulações em *software* e *hardware*. A diferença entre ambos está no experimento em *hardware*, em que utiliza a mesma rede neural treinada em *software*, mas, nesse caso, os pesos sinápticos do SOM devem ser transformados para o formato numérico binário do sistema embarcado.

Com todas as amostras de dados e sistemas para testes prontos, foram realizados, em primeiro lugar, os experimentos em *software*. Estes são importantes para gerar resultados úteis para posterior comparação com relação aos resultados obtidos em *hardware*. Também foram executados experimentos para Detecção de Anormalidades (gráfico do erro de quantização), Diagnóstico de Falhas (visualização da U-Matrix) e Predição e Monitoramento de Falhas (visualização da U-Matrix com o trajeto dos neurônios vencedores).

Após obter os resultados de *software*, foram executados experimentos na plataforma de sistemas embarcados em *hardware*. Nesse caso, foram executados os experimentos para Detecção de Anormalidades e Diagnóstico de Falhas.

Com os resultados obtidos do sistema embarcado, comprovou-se que a implementação do *hardware* está correta em relação aos resultados do *software*, comprovando a correta implementação do SOM.

Os resultados obtidos para Detecção de Anormalidades apresentaram diferenças muito pequenas e insignificantes quando comparados com do *software*. Os resultados obtidos para o Diagnóstico de Falhas foram os mesmos resultados do *software*.

Além disso, também foram apresentadas algumas imagens da saída obtida por meio da entrada e saída padrão do sistema embarcado, observados no programa *HyperTerminal*. Nas saídas, são apresentados as chamadas para os programas de teste e os resultados obtidos pela leitura nos registradores do *hardware* do SOM.

A principal vantagem desse protótipo é o fato de levar menos tempo para a execução e utilizar menos recursos computacionais, pois, nesse caso, não exige a utilização de um

PC industrial em cada equipamento em que se deseja implantar o Sistema de Manutenção Inteligente. Este protótipo pode ser aprimorado, para isso seria necessário:

- Implementar tratamento para os casos de *overflow* e *underflow* nos cálculos feitos pelos componentes de operações matemáticas (soma, adição, multiplicação e raiz quadrada).
- Substituir os componentes que realizam cálculos matemáticos em ponto flutuante por ponto fixo, para acelerar o tempo de execução dos cálculos e reduzir a área ocupada pelo projeto.
- Substituir o algoritmo que calcula a distância entre dois vetores, a Distância Euclidiana, por um algoritmo de menor custo com a Distância de Manhattan (HIKAWA, 2005), pois grande parte do custo da Distância Euclidiana está vinculada ao cálculo da raiz quadrada.
- Implementar tratamento de falhas, durante o funcionamento do *hardware* do SOM, já que podem ocorrer alguns casos de falhas, como a falta de algum dado na memória, *overflow*, fluxo de execução entrar em *loop* infinito ou erro nos parâmetros de configuração.
- Implementar uma interface para utilizar o recurso de interrupção do sistema embarcado, para por exemplo, informar ao sistema operacional o fim dos cálculos ou solicitar um novo vetor de entrada.
- Retirar a memória interna BRAM e passar a armazenar o SOM treinado na memória SRAM do sistema embarcado. Desse modo, não precisaria resintetizar todo o sistema embarcado cada vez que mudar os dados de treinamento, o que possibilita utilizar um mesmo algoritmo do SOM para várias propostas de treinamentos. Na situação atual do trabalho, isso não é possível.
- O *hardware* do SOM deve acessar os vetores de entrada diretamente na memória SRAM do sistema embarcado, evitando o gargalo que pode ocorrer durante a carga dos vetores pelo programa de teste em *software*, pois é submetido ao escalonamento de tarefas pelo sistema operacional.

# 6 CONCLUSÕES E TRABALHOS FUTUROS

Esta dissertação teve como objetivo maior o projeto de um protótipo de um sistema embarcado que implemente técnicas de manutenção inteligente, utilizando os Mapas Auto-Organizáveis (SOM) para aplicar técnicas de tolerância a falhas em atuadores elétricos na rede dutoviária da Transpetro. Cinco foram as áreas envolvidas nesta pesquisa: (1) manutenção inteligente, (2) testes e tolerância a falhas, (3) redes neurais artificiais, em especial o SOM, (4) projeto de circuitos digitais em FPGA e (5) projeto de sistemas embarcados em *hardware* e *software*. Tomando por base essa divisão, o texto foi escrito de forma a distribuir os assuntos visando facilitar o entendimento e encadeando os principais conceitos relacionados com cada uma das áreas citadas. Em todos os capítulos, procurou-se apresentar pontos que fossem fundamentais para o projeto proposto e para os trabalhos futuros.

O Capítulo 1 fez uma breve explanação do contexto geral do trabalho, apresentando o contexto do problema, a motivação, os objetivos, as contribuições e a organização do restante da dissertação.

No Capítulo 2, foi tratada a apresentação do problema da manutenção na indústria, os conceitos e a metodologia de manutenção inteligente. O problema abordado como estudo de caso nessa dissertação é o de falhas em atuadores elétricos de válvulas em dutos da Transpetro.

Pode-se concluir do Capítulo 2 que a estratégia de manutenção inteligente pode oferecer novos benefícios para o planejamento da manutenção nas empresas, a fim de antecipar a intervenção antes da ocorrência da falha. Para o estudo de caso, é imprescindível o planejamento de reparos nos diversos atuadores distribuídos ao longo de toda a rede dutoviária.

No Capítulo 3, foram apresentados conceitos, fundamentos e técnicas de tolerância a falhas para detecção, diagnóstico, predição e monitoramento de falhas, focando no uso das redes neurais. Além disso, foi definida a utilização do SOM e TKM como ferramentas computacionais para implementar as tarefas de tolerância a falhas, em que foram apresentados os princípios teóricos e trabalhos relacionados.

É possível inferir, a partir do Capítulo 3, a ideia de uma arquitetura geral para o desenvolvimento de um Sistema de Detecção, Diagnóstico e Predição de Falhas e as ferramentas de redes neurais como técnicas computacionais para implementá-lo. Concluí-se que o SOM tem as características essenciais para desenvolver esse sistema, pois é geralmente aplicado à classificação de padrões (formação de agrupamentos), trata

de grande quantidade de dados, a aprendizagem não supervisionada preserva as relações estatísticas dos dados de entrada, e a ferramenta se presta à visualização de processos dinâmicos, além, claro, da vasta aplicação dessa rede neural na literatura para solução de problemas semelhantes ao do presente trabalho.

Os capítulos anteriores mostraram o estado da arte necessário para compreender como se implementar um sistema de manutenção inteligente. A partir dos conceitos e das definições extraídos desses capítulos, foram propostos os próximos capítulos que estão relacionados com a construção do protótipo do sistema embarcado para manutenção inteligente.

No Capítulo 4, foi abordada a construção do protótipo do sistema embarcado para o Sistema de Manutenção Inteligente, proposto para monitorar falhas em atuadores elétricos. Também foi definida a estrutura do sistema, mostrando como aplicar e utilizar as redes neurais para realizar a detecção, diagnóstico e predição de falhas. Além disso, foram implementados *softwares* para experimentos de simulação em MATLAB e, a seguir, apresentada a construção do protótipo do SOM em *hardware*, em plataforma FPGA. É importante ressaltar que foram prototipadas apenas em *hardware* as técnicas de *detecção* e *diagnóstico* de falhas.

Para construção do protótipo, os algoritmos do SOM foram implementados VHDL, sintetizados em FPGA e apresentados resultados de área ocupada, desempenho e consumo de potência do circuito digital. Após, foi definida a arquitetura do sistema embarcado, que visa utilizar o componente de *hardware* do SOM a ser acoplado em um sistema computacional, a fim de realizar experimentos mistos de *software* e *hardware*. Por fim, foi implantando neste, um sistema operacional embarcado baseado em GNU/Linux para executar experimentos de casos de testes utilizando *softwares* embarcados. Esse protótipo tem o objetivo de, no futuro, ser implantado no atuador elétrico.

A partir do Capítulo 4 é possível concluir que o protótipo implementou a rede neural SOM em FPGA e tornou disponível uma plataforma para experimentos de manutenção inteligente. Essa contribuição é necessária para realizar experimentos de detecção e diagnóstico de falhas, simulando situações reais de monitoramento *on-line* da condição de operação de atuadores elétricos. Outro aspecto a ressaltar é o espaço disponível no FPGA, pois possibilita embarcar as outras etapas da metodologia de manutenção inteligente, como o processamento de sinais e a tomada de decisão.

A prototipação em um sistema embarcado é importante para fornecer subsídios para atuação local, já que o sistema passa a ser autônomo e pode vir a ser auto-reconfigurar com a incidência de falhas. Outro ponto importante é que evita-se o envio de muitas informações para operadores humanos, como nos sistemas supervisórios, que, muitas vezes, não os tratam de forma adequada. Nesse protótipo embarcado, o sistema envia uma mensagem aos gerentes de manutenção, informando a detecção de anormalidade, diagnóstico da causa e uma predição até a manifestação da falha, o que auxilia na tomada de decisão para planejar as ações de manutenção corretiva ou preventiva.

No Capítulo 5, são apresentados os resultados experimentais para o sistema de manutenção inteligente divididos em duas etapas: simulação em *software* e execução do protótipo em *hardware*. Além disso, foi detalhado o estudo de caso dos atuadores elétricos em válvulas de dutos da Transpetro.

Em geral, para a realização dos experimentos, os dados de torque do atuador e posição do obturador da válvula foram simulados por um modelo matemático implementado em MATLAB. Contudo, condições de comportamento do atuador e válvula (normal, degradação e falha) foram reproduzidas em uma ferramenta de injeção de falhas no modelo matemático. Os experimentos foram divididos em detecção e diagnóstico. Para cada caso, foi apresentada a estrutura de injeção de falhas para treinar sua rede neural. Finalmente, foram realizados os experimentos para avaliar a capacidade das redes neurais em detectar, diagnosticar e prever falhas no sistema de manutenção inteligente, validando as técnicas para o estudo de caso.

A partir dos experimentos realizados no Capítulo 5, validou-se o uso do SOM para detecção e diagnóstico e do TKM para predição (com ressalvas). Nos experimentos tanto em *software* quanto no protótipo em *hardware*, foram avaliadas simulações com dados oriundos do modelo matemático. De acordo com os resultados obtidos em *software*, comprovou-se o uso do SOM para desempenhar as tarefas de detecção de anormalidades e diagnóstico de falhas e do uso do TKM para predição de comportamentos futuros (necessita de uma pesquisa mais minuciosa). Além disso, a validação dessas técnicas possibilitou avançar para a prototipação, pois os algoritmos funcionam para desempenhar o especificado e foram utilizados como padrão comparativo para validação do protótipo.

Já nos experimentos no protótipo em *hardware*, os resultados foram comparados àqueles obtidos pelos experimentos em *software*. A partir disso, comprovou-se o correto projeto do protótipo do SOM em *hardware* para desempenhar as tarefas de detecção e diagnóstico, bem como o exato funcionamento do protótipo do sistema embarcado implementado em placa de desenvolvimento.

Com a validação da técnica do SOM para a detecção de anormalidades, diagnóstico de falhas e do TKM para predição do comportamento, assim como a prototipação do SOM para detecção e diagnóstico em sistema embarcado, fica disponível uma plataforma para novos experimentos sobre o atuador elétrico.

O presente trabalho atingiu os objetivos esperados, propondo uma solução para o problema de detectar, diagnosticar e prever falhas em sistemas ou processos industriais e, em especial, uma proposta inicial de um protótipo em sistema embarcado para o atuador elétrico. É claro que esse sistema proposto tem limitações e ainda precisa passar por novos testes e melhorias, sobretudo a parte de predição. Para tornar o sistema completo, é necessário implementar as demais etapas de um sistema de manutenção inteligente, como a aquisição de sinais, processamento digital de sinais, tomada de decisão e reconfiguração, que serão abordados em trabalhos futuros.

Durante o desenvolvimento do presente trabalho, surgiram inúmeras possibilidades para ampliação da proposta. Devido à complexidade e abrangência dos objetivos traçados e das áreas da ciência a serem exploradas, muitas da ideia permanecem como possíveis trabalhos futuros. A seguir, são discutidos os principais desdobramentos e direções para o encaminhamento de pesquisas futuras:

- Realizar mais treinamentos e testes no SOM, aumentando-se o domínio de falhas do atuador a serem exploradas. Os casos de teste simples adotados e realizados comprovam o potencial do SOM. No entanto, os experimentos certamente não foram exaustivos e seria interessante abranger um número superior de falhas e avaliar o comportamento do SOM na generalização dessas informações. Além disso, aumentar a riqueza dos sinais de

treinamento, com informações de ruído, vibração, temperatura, análise de corrente, etc.

- Pesquisar de forma mais aprofundada o algoritmo do TKM para predição do comportamento, realizar mais experimentos com outros dados de entrada, bem como gerar amostras de teste de formas diferentes para avaliar a generalização da predição. Além, claro, de explorar outras alternativas para predição, pois as pesquisas sobre representação temporal existentes ainda são pouco exploradas.
- Aprimorar o protótipo do sistema embarcado conforme descrito no final da seção 5.9.
- Pesquisar e implementar a etapa de *tomada de decisão*, que visa processar as informações obtidas pelos algoritmos de tolerância a falhas, a fim de fornecer decisões para a equipe de manutenção planejar intervenções de reparos nos equipamentos, antecipando as falhas.
- Pesquisar e implementar a etapa de *reconfiguração* local no equipamento, assim como implantá-la no sistema embarcado. Essa etapa é fundamental para tornar o sistema de manutenção inteligente, de forma proativa, para que seja integrado com a tomada de decisão.
- Implantar no sistema embarcado os algoritmos da etapa de processamento digital de sinais.
- Pesquisar e implantar, no sistema embarcado, a etapa de aquisição de dados. Seria interessante que esses dados fossem fornecidos por um sistema supervisório externo para formar uma base de dados com históricos.
- Realizar um estudo e experimentos em um contexto geral de todo o sistema de manutenção inteligente, de forma a avaliar as características presentes no trabalho, de acordo com conceitos apresentados na Seção 3.5.5.
- Aplicar a proposta vista no presente trabalho em outros estudos de casos, por exemplo, rolamentos, motores em geral, robôs, etc.

# REFERÊNCIAS


AGARWAL, M.; PAUL, B. C.; ZHANG, M.; MITRA, S. Circuit Failure and Its Application to Transistor Aging. **25th IEEE VLSI Test Symposium**, v. 1, n. 1, p. 277–286. doi: Berkeley, 2007.

ALMEIDA, M. T. Manutenção Preditiva: Confiabilidade e Qualidade. **MTA Engenharia de Vibrações** . Recuperado Março 3, 2009, de http://www.mtaev.com.br/download/mnt1.pdf, 2007.

AVIZIENIS, A.; LAPRIE, J.; RANDELL, B.; LANDWEHR, C. Basic concepts and taxonomy of dependable and secure computing. **Dependable and Secure Computing, IEEE Transactions on**, v. 1, n. 1, p. 11-33. doi: 10.1109/TDSC.2004.2, 2004.

BARRETO, G.; ARAUJO, A.; RITTER, H. Time in self-organizing maps: An overview of models. **International Journal of Computer Research**, v. 10, n. 2, p. 139-179, 2001.

BARRETO, G. A. **Redes neurais não-supervisionadas para processamento de sequências temporais**. Dissertação de Mestrado. Mestrado em Engenharia Elétrica., USP, 1998.

BARRETO, G. A. **Redes neurais nao-supervisionadas temporais para identificação e controle de sistemas dinâmicos**. Tese de doutorado. Doutorado em Engenharia Elétrica., USP, 2002.

BARRETO, G. A.; MOTA, J. C. M.; SOUZA, L. G. M.; ET AL. A New Approach to Fault Detection and Diagnosis in Cellular Systems Using Competitive Learning. **Proceedings of the VII Brazilian Symposium on Neural Networks (SBRN04)**, 2004.

BASTOS, E. N. F. **Uma rede neural Auto-Organizável construtiva para aprendizado perpétuo de padrões espaço-temporais**. Dissertação de Mestrado, Mestrado em Ciência da Computação, UFRGS, 2007.

BENGTSSON, M. Condition Based Maintenance System Technology – Where is Development Heading? **Proceedings of the 17th European Maintenance Congress, EUROMAINTENANCE 2004**, v. 17, n. 1, p. 82-91. doi: Barcelona, 2004.

CARRO, L. **Projeto de Prototipação de Sistemas Digitais**. v. 1, p.234. Porto Alegre: Editora da UFRGS, 2001.

CARRO, L.; WAGNER, F. Sistemas Computacionais Embarcados. **Capítulo 2 das Jornadas de Atualização em Informática, XXII JAI 2003 - Sociedade Brasileira de Computação**, v. 1, p. 45-94, 2003.

CHAPPEL, G. J.; TAYLOR, J. G. The temporal kohonen map. **IEEE Transactions on Neural Networks**, v. 6, 1993.

DÍAZ, I.; DOMÍNGUEZ, M.; CUADRADO, A. A.; FUERTES, J. J. A new approach to exploratory analysis of system dynamics using SOM. Applications to industrial processes. **Expert Systems with Applications**, v. 34, n. 4, p. 2953-2965. doi: 10.1016/j.eswa.2007.05.031, 2008.

DJURDJANOVIC, D.; LEE, J.; NI, J. Watchdog Agent—an infotronics-based prognostics approach for product performance degradation assessment and prediction. **Advanced Engineering Informatics**, v. 17, n. 3-4, p. 109-125. doi: 10.1016/j.aei.2004.07.005, 2003.

DOMÍNGUEZ, M.; FUERTES, J.; REGUERA, P.; DÍAZ, I.; CUADRADO, A. Internet-based remote supervision of industrial processes using self-organizing maps. **Engineering Applications of Artificial Intelligence**, v. 20, n. 6, p. 757-765. doi: 10.1016/j.engappai.2006.11.017, 2007.

ELMAN, J. L. Finding structure in time. **Cognitive Science**, v. 14, n. 2, p. 179-211, 1990.

ENDO, M.; UENO, M.; TANABE, T.; YAMAMOTO, M. Clustering method using self-organizing map. In: Neural Networks for Signal Processing X, 2000. Proceedings of the 2000 IEEE Signal Processing Society Workshop. **Anais...** . v. 1, p.261-270 vol.1. doi: 10.1109/NNSP.2000.889417, 2000.

ENDRENYI, J.; ABORESHEID, S.; ALLAN, R.; ET AL. The present status of maintenance strategies and the impact of maintenance on reliability. **IEEE Transactions on Power Systems**, v. 16, n. 4, p. 638-646. doi: 10.1109/59.962408, 2001.

GÉRMEN, E.; ECE, D. G.; GEREK, Ö. N. Self Organizing Map (SOM) Approach for Classification of Mechanical Faults in Induction Motors. **Lecture Notes in Computer Science**, v. 4507, n. 1, p. 855-861. doi: Eskisehir, 2007.

GOH, K.; TJAHJONO, B.; BAINES, T.; SUBRAMANIAM, S. A Review of Research in Manufacturing Prognostics. In: **Anais...** . p.417-422. doi: 10.1109/INDIN.2006.275836, 2006.

GONÇALVES, L. F.; BOSA, J. L.; HENRIQUES, R. V. B.; LUBASZEWSKI, M. S. Design of an embedded system for the proactive maintenance of electrical valves. In: Proceedings of the 22nd Annual Symposium on Integrated Circuits and System Design: Chip on the Dunes. **Anais...** . p.1-6. Natal, Brazil: ACM. doi: 10.1145/1601896.1601906, 2009.

GONÇALVES, L. F.; BOSA, J. L.; LUBASZEWSKI, M. S.; PEREIRA, C. E.; HENRIQUES, R. V. B. Um método de classificação de falhas em atuadores elétricos

baseado em mapas auto-organizáveis. **Anais do 8 Congresso Brasileiro de Automática - CBA2008**. doi: Juiz de Fora, 2008.

GONÇALVES, L. F.; LAZZARETTI, E. P.; LUBASZEWSKI, M. S.; ET AL. Desenvolvimento de um Sistema de Manutenção Inteligente para Válvulas Eletrônicas. **Anais do XIX Congresso Internacional de Engenharia Mecânica - COBEM**, p. 8, 2007.

HAYKIN, S. **Redes neurais: princípios e prática**. 2 ed. São Paulo: Bookman, 2001.

HIKAWA, H. FPGA implementation of self organizing map with digital phase locked loops. **Neural Networks**, v. 18, n. 5-6, p. 514-522. doi: 10.1016/j.neunet.2005.06.012, 2005.

HUANG, R.; XI, L.; LI, X.; ET AL. Residual life predictions for ball bearings based on self-organizing map and back propagation neural network methods. **Mechanical Systems and Signal Processing**, v. 21, n. 1, p. 193-207. doi: 10.1016/j.ymssp.2005.11.008, 2007.

ISERMANN, R. Supervision, fault-detection and fault-diagnosis methods — An introduction. **Control Engineering Practice**, v. 5, n. 5, p. 639-652. doi: 10.1016/S0967-0661(97)00046-4, 1997.

JÄMSÄ-JOUNELA, S. -.; VERMASVUORI, M.; ENDÉN, P.; HAAVISTO, S. A process monitoring system based on the Kohonen self-organizing maps. **Control Engineering Practice**, v. 11, n. 1, p. 83-92. doi: 10.1016/S0967-0661(02)00141-7, 2003.

JARDINE, A. K.; LIN, D.; BANJEVIC, D. A review on machinery diagnostics and prognostics implementing condition-based maintenance. **Mechanical Systems and Signal Processing**, v. 20, n. 7, p. 1483-1510. doi: 10.1016/j.ymssp.2005.09.012, 2006.

JESMAN, R.; VALLINA, F. M.; SANIIE, J. MicroBlaze Tutorial creating a simple embedded system and adding custom peripherals using Xilinx EDK Software tools. . Embedded Computing and Signal Processing Laboratory, Illinois Institute of Technology. Recuperado de http://ecasp.ece.iit.edu/mbtutorial.pdf, 2006.

KANGAS, J. Time-delayed self-organizing maps. In: **Anais...** . v. 2, p.331-336. doi: 10.1109/IJCNN.1990.137735, 1990.

KARDEC, A.; NASCIF, J. Manutenção: função estratégica. **Rio de Janeiro: Qualitymark**, 2001.

KATIPAMULA, S.; BRAMBLEY, M. R. Methods for Fault Detection, Diagnostics, and Prognostics for Building Systems—A Review, Part I. **International Journal of HVAC&R Research**, v. 11, n. 1, p. 3-25, 2005.

KOHONEN, T. The self-organizing map. **Proceedings of the IEEE**, v. 78, n. 9, p. 1464-1480. doi: 10.1109/5.58325, 1990.

KOHONEN, T. **Self-organizing maps**. Springer, 2001.

KOHONEN, T.; OJA, E.; SIMULA, O.; VISA, A.; KANGAS, J. Engineering applications of the self-organizing map. **Proceedings of the IEEE**, v. 84, n. 10, p. 1358-1384. doi: 10.1109/5.537105, 1996.

KOTHAMASU, R.; HUANG, S.; VERDUIN, W. System health monitoring and prognostics — a review of current paradigms and practices. **The International Journal of Advanced Manufacturing Technology**, v. 28, n. 9, p. 1012-1024. doi: 10.1007/s00170-004-2131-6, 2006.

LEBOLD, M.; THURSTON, M. Open standards for condition-based maintenance and prognostic systems. In: **Anais...** . p.6–9, 2001.

LEE, J.; QIU, H.; NI, J.; DJURDJANOVI, D. Infotronics technologies and predictive tools for next-generation maintenance systems. In: **Anais...** . Salvador, Brasil, 2004.

LEE, J. E-manufacturing—fundamental, tools, and transformation. **Robotics and Computer-Integrated Manufacturing**, v. 19, n. 6, p. 501-507. doi: 10.1016/S0736-5845(03)00060-7, 2003.

LEE, J.; NI, J.; DJURDJANOVIC, D.; QIU, H.; LIAO, H. Intelligent prognostics tools and e-maintenance. **Computers in Industry**, v. 57, n. 6, p. 476-489. doi: 10.1016/j.compind.2006.02.014, 2006.

MARÇAL, R. F. **Um método para detectar falhas incipientes em máquinas rotativas baseado em análise de vibração e lógica fuzzy**. Doutorado, Programa de Pós-Graduação em Engenharia de Minas, Metalúrgica, e de Materiais - PPGEM, Universidade Federal do Rio Grande do Sul - UFRGS, 2000.

MARCORIN, W. R.; LIMA, C. R. C. Análise dos Custos de Manutenção e de Não-manutenção de Equipamentos Produtivos. **Revista de Ciência & Tecnologia**, v. 11, n. 22, p. 35-42, 2003.

MIRSHAWKA, V.; OLMEDO, N. Manutenção-Combate aos Custos da Não-Eficácia-A Vez do Brasil. **São Paulo: MacGraw-Hill**, 1999.

MOSER, L. C. **Modelo de um neurônio diferenciador-integrador para representação temporal em arquiteturas conexionistas**. Dissertação de Mestrado, Mestrado em Ciência da Computação, UFRGS, 2004.

MULLER, A.; MARQUEZ, A.; IUNG, B. On the concept of e-maintenance: Review and current research. **Reliability Engineering & System Safety**, v. 93, n. 8, p. 1165-1187. doi: 10.1016/j.ress.2007.08.006, 2008.

MURTY, A.; NAIKAN, V. Availability and maintenance cost optimization of a production plant. **International Journal of Quality & Reliability Management**, v. 12, n. 2, p. 28 - 35. doi: 10.1108/02656719510080596, 1995.

NAGARAJAN, R.; ASOKAN, V. Getting started with uClinux on the MicroBlaze Processor. **Xilinx Application Notes**, p. 34, 2007.

NIKKANEN, K. **uClinux as an Embedded Solution**. Bachelor's Thesis, Turku Polytechnic, 2003.

NUNES, E. L. **Manutenção centrada em confiabilidade (MCC) - Análise da implantação em uma sistemática de manutenção preventiva consolidada.** Dissertação de Mestrado, Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal de Santa Catarina, 2001.

OTANI, M.; MACHADO, W. V. A proposta de desenvolvimento de gestão da manutenção industrial na busca da excelência ou classe mundial. **Revista Gestão Industrial**, v. 4, n. 2, 2008.

QIU, H.; LEE, J. Feature Fusion and Degradation Detection Using Self-Organizing Map. **International Conference on Machine Learning and Applications**, v. 1, n. 1, p. 107- 114. doi: Milwaukee, 2004.

QIU, H.; LEE, J.; LIN, J.; YU, G. Robust performance degradation assessment methods for enhanced rolling element bearing prognostics. **Advanced Engineering Informatics**, v. 17, n. 3-4, p. 127-140. doi: 10.1016/j.aei.2004.08.001, 2003.

QIU, H.; LEE, J.; LIN, J.; YU, G. Wavelet filter-based weak signature detection method and its application on rolling element bearing prognostics. **Journal of Sound and Vibration**, v. 289, n. 4-5, p. 1066-1090. doi: 10.1016/j.jsv.2005.03.007, 2006.

SAXENA, A.; SAAD, A. Fault diagnosis in rotating mechanical systems using self-organizing maps. **Artificial Neural Networks in Engineering (ANNIE04)**, 2004.

SHIKARI, B.; SADIWALA, C. M.; DWIVEDI, R. K. Automation in Condition Based Maintenance using vibration analysis. **Mechanical Engineering Department - Maulana Azad National Institute of Technology**, 2004.

SILBERSCHATZ, A. **Sistemas Operacionais Com Java**. 7 ed., v. 1, p.696. Campus, 2008.

SILVA, M. A. **Mapas Auto-Organizáveis na Análise Exploratória de Dados Geoespaciais Multivariados**. Dissertação de Mestrado, Mestrado em Computação Aplicada., INPE, 2004.

DE SOUZA, C. A.; DE FREITAS, C. M. Análise de causas de acidentes e ocorrências anormais, relacionados ao trabalho, em uma refinaria de petróleo, Rio de Janeiro. **Cad. Saúde Pública**, v. 19, n. 5, 2003.

SUDHA, N.; SRIKANTHAN, T.; MAILACHALAM, B. A VLSI architecture for 3-D self-organizing map based color quantization and its FPGA implementation. **J. Syst. Archit.**, v. 48, n. 11-12, p. 337-352, 2003.

THURSTON, M. An open standard for Web-based condition-based maintenance systems. In: **Anais...** . p.401-415. doi: 10.1109/AUTEST.2001.949021, 2001.

VACHKOV, G.; KOMATSU, K.; FUJII, S. Classification-based behavior model for detection of abnormal states in systems. In: **Anais...** . p.611-616, 2004.

VENKATASUBRAMANIAN, V.; RENGASWAMY, R.; KAVURI, S. N.; YIN, K. A review of process fault detection and diagnosis : : Part III: Process history based methods. **Computers & Chemical Engineering**, v. 27, n. 3, p. 327-346. doi: 10.1016/S0098-1354(02)00162-X, 2003.

VENKATASUBRAMANIAN, V.; RENGASWAMY, R.; YIN, K.; KAVURI, S. N. A review of process fault detection and diagnosis : : Part I: Quantitative model-based methods. **Computers & Chemical Engineering**, v. 27, n. 3, p. 293-311. doi: 10.1016/S0098-1354(02)00160-6, 2003.

VESANTO, J.; HIMBERG, J.; ALHONIEMI, E.; PARHANKANGAS, J. Self-organizing map in Matlab: the SOM toolbox. In: Proceedings of the Matlab DSP Conference. **Anais...** . v. 99, p.16–17, 1999.

VESANTO, J.; HIMBERG, J.; ALHONIEMI, E.; ET AL. Som toolbox for matlab. **Techn. Ber., Helsinki University of Technology**, 2000.

VESANTO, J. SOM-Based Data Visualization Methods. **INTELLIGENT DATA ANALYSIS**, v. 3, p. 111--126, 1999.

WEBER, T. S. Um roteiro para exploração dos conceitos básicos de tolerância a falhas. . Notas de Aula do Curso de Especialização em Redes e Sistemas Distribuidos, Instituto de Informática, UFRGS, 2002.

WEBER, T. S. Tolerância a falhas: conceitos e exemplos. **Intech Brasil**, v. 52, p. 32-42, 2003.

WILLIAMS, J.; PETALOGIX, I. PetaLinux Platform from Scratch. **Petalogix Tutorials**, 2008.

WONG, M.; JACK, L.; N, A.; I. Modified self-organising map for automated novelty detection applied to vibration signal monitoring. **Mechanical Systems and Signal Processing**, v. 20, n. 3, p. 593-610. doi: 10.1016/j.ymssp.2005.01.008, 2006.

XILINX, I. Designing Custom OPB Slave Peripherals for MicroBlaze. **Xilinx Tutorial Embedded Development**, p. 30, 2002.

XILINX, I. EDK MicroBlaze Tutorial. **Xilinx Tutorial Embedded Development**, 2003.

XILINX, I. XST User Guide 8.1i. **Xilinx Tutorial Embedded Development**, p. 570, 2005a.

XILINX, I. EDK Base system builder (BSB) support for XUPV2P board. **Xilinx University Program**, p. 23, 2005b.

XILINX, I. Embedded system tools reference manual. **Embedded Development Kit EDK 8.1i**, v. 1, p. 228, 2005c.

XILINX, I. Custom peripheral design guide. **Xilinx Tutorial Embedded Development**, p. 82, 2005d.

XILINX, I. Virtex-II Pro development system. **Xilinx University Program**, , n. 1, p. 142, 2008.

YAGHMOUR, K. **Building Embedded Linux Systems**. 2 ed., p.462. O'Reilly Media, Inc. Recuperado Novembro 17, 2009, de http://portal.acm.org/citation.cfm?id=1096085, 2008.

YANG, B. S.; HAN, T.; AN, J. L. ART–KOHONEN neural network for fault diagnosis of rotating machinery. **Mechanical Systems and Signal Processing**, v. 18, n. 3, p. 645-657. doi: 10.1016/S0888-3270(03)00073-6, 2004.

YANG, B.; HWANG, W.; KIM, D.; TAN, A. C. Condition classification of small reciprocating compressor for refrigerators using artificial neural networks and support vector machines. **Mechanical Systems and Signal Processing**, v. 19, n. 2, p. 371-390. doi: 10.1016/j.ymssp.2004.06.002, 2005.

ZHONG, F.; SHI, T.; HE, T. Fault diagnosis of motor bearing using self-organizing maps. In: **Anais...** . v. 3, p.2411-2414 Vol. 3. doi: 10.1109/ICEMS.2005.203004, 2005.

# ANEXO

Como anexo está o artigo publicado no Congresso Brasileiro de Automática em 2008, apresentando maiores detalhes das equações para construção do modelo matemático para o conjunto atuador e válvula.

# UM MÉTODO DE CLASSIFICAÇÃO DE FALHAS EM ATUADORES ELÉTRICOS BASEADO EM MAPAS AUTO-ORGANIZÁVEIS

Luiz F. Gonçalves*, Jefferson L. Bosa*, Marcelo S. Lubaszewski*, Carlos E. Pereira*, Renato V. B. Henriques*

*UFRGS - PPGEE - Av. Osvaldo Aranha, 103, Bom Fim, CEP:90031-190, Porto Alegre, RS, Brasil

Emails: luizfg@ece.ufrgs.br, jlbosa@inf.ufrgs.br, luba@ece.ufrgs.br, cpereira@ece.ufrgs.br, rventura@ece.ufrgs.br

**Abstract**— This paper presents some of the main activities developed for the implementation of an intelligent maintenance embedded system in electric actuators. The main idea is to develop an intelligent maintenance embedded system to evaluate and determine the degradation of the of actuators performance and achieve a forecast and a diagnosis of failures. For this, signal processing techniques and statistical methods, but specifically, the transform Wavelet Packet and the Logistic Regression method are used. Also, the developed a system of failure classification and prediction using self-organizing maps will be conducted. It is expected, with the deployment embedded of this system, to prevent damages in actuators and reduce costs from unexpected failures. The main reasons, objectives and steps for the implementation of this system will also be presented.

**Keywords**— Classification, Failures, Electric Actuators, Self-Organizing Maps.

**Resumo**— Este artigo aborda algumas das principais atividades desenvolvidas para a implantação de um sistema de manutenção inteligente embarcado em atuadores elétricos. A idéia principal é desenvolver um sistema de manutenção inteligente embarcado para avaliar e determinar a degradação do desempenho dos atuadores e realizar uma previsão e um diagnóstico de falhas. Para isto, utilizou-se de técnicas de processamento de sinais e métodos estatísticos, mas especificamente a transformada Wavelet Packet e o método de Regressão Logística. Além disso, foi feito o desenvolvimento de um sistema de classificação de falhas usando mapas auto-organizáveis. Espera-se, com a implantação embarcada desse sistema, evitar danos nos atuadores e reduzir os custos provenientes de falhas inesperadas. As principais etapas e objetivos para a implementação desse sistema também são apresentadas.

**Palavras-chave**— Classificação, Falhas, Atuadores Elétricos, Mapas Auto-Organizáveis.

## 1 Introdução

Os equipamentos ou processos industriais, a medida que são utilizados, ficam sujeitos a vários tipos de degradação: desgaste, sujeira, corrosão, rachaduras, e outras anomalias. Caso não sejam tomadas algumas medidas corretivas com a intenção de restaurar esses equipamentos, os mesmos começarão a apresentar algum defeito.

Permanecendo o defeito, não sendo realizada nenhuma ação corretiva, o equipamento ou processo poderá falhar, ficando indisponível para desempenhar sua função. A manutenção consiste de uma série de técnicas e medidas de prevenção, correção e predição de falhas.

Essas medidas são praticadas principalmente com a intenção de corrigir ou consertar os danos provocados pela degradação, promover uma maior sustentabilidade, manter os equipamentos em funcionamento, reduzir a probabilidade de falhas e a degradação de componentes.

Assim, a manutenção pode ser dividida, com relação às medidas tomadas perante as falhas, em quatro estratégias: corretiva, preventiva, preditiva e proativa (Lee et al., 2004).

A estratégia de manutenção corretiva apresenta uma maior ênfase no conserto dos equipamentos após a ocorrência de falhas. Portanto, essa técnica tradicional resulta em atrasos de operação, ociosidade do sistema e custos adicionais.

A manutenção preventiva, mais contemporânea, corresponde às ações previstas, preparadas ou programadas antes do provável aparecimento da falha (Djurdjanovic et al., 2003).

Entretanto, o principal inconveniente encontrado nesse tipo de estratégia são as constantes intervenções, muitas vezes desnecessárias em virtude das variações presentes nos materiais, peças, e freqüência de uso dos equipamentos.

Já a manutenção preditiva ocorre quando se aplica uma supervisão contínua dos parâmetros de controle e desempenho (Qiu et al., 2006). A principal desvantagem dessa estratégia é o custo de implementação que é, normalmente, bem elevado.

Em função dos recentes avanços da eletrônica, da computação, e dos sistemas embarcados a manutenção proativa, em especial, também conhecida como manutenção inteligente, vem ganhando força em todo o mundo.

A manutenção proativa é aquela que, além de monitorar o equipamento ou processo, diagnostica e quantifica a perda de desempenho do sistema ao longo do tempo.

A partir disso, torna-se possível que, em função do desgaste identificado (degradação), seja programada a troca das peças deterioradas em momentos de ociosidade do equipamento ou mesmo a reconfiguração automática do sistema de modo a continuar operando, até a troca das peças defeituosas.

Essas grandes transformações permitem um diagnóstico e análise muito maior das origens das falhas e dos seus efeitos, influenciando diretamente as atividades de manutenção, em especial, as preditivas e proativas (Djurdjanovic et al., 2003).

Assim, quando sensores com dispositivos inteligentes estão conectados em um barramento industrial, ou conectados diretamente nos equipamentos, e seus dados são analisados continuamente por sofisticados sistemas embarcados inteligentes é possível ir além da manutenção preditiva, evoluindo para uma predição inteligente (proativa), localizando com exatidão os componentes, peças, ou mecanismos degradados que estão propensos a falhar.

Neste contexto, alguns centros focados na manutenção preditiva vêm sendo criados. O Center for Intelligent Maintenance Systems (IMS Center), em especial, está desenvolvendo uma solução embarcada para avaliação, predição, e diagnóstico de falhas chamada Watchdog Agent (um PC industrial), além de uma interface implementada para o software Matlab, conhecida como Watchdog Agent Toolbox.

O presente trabalho insere-se no contexto da criação de um centro de manutenção inteligente no Brasil, sendo esse uma parceria entre o IMS Center, os Departamentos Nacionais e Regionais da Bahia e do Rio Grande do Sul do SENAI, e a Universidade Federal do Rio Grande do Sul.

Este trabalho apresenta a modelagem, as ferramentas, e resultados da avaliação de desempenho e de classificação de falhas em atuadores elétricos. Os atuadores são oriundos da empresa Coester Automação S.A., ver Fig. 2.

Atuadores, elétricos ou pneumáticos, são equipamentos que permitem motorização de válvulas, *dampers*, comportas e outros equipamentos similares. Os atuadores elétricos, em especial, sofrem diversos processos de degradação, tais como: corrosão, desgaste, e atrito, que irão provocar diferentes tipos de falhas, tais como a quebra de peças. A finalidade do atuador, considerada neste artigo, é realizar o controle do fluxo por meio de uma válvula gaveta.

O principal objetivo deste trabalho, além do aprendizado da metodologia, técnicas e ferramentas desenvolvidas pelo IMS, é a implantação de um sistema embarcado de manutenção inteligente em atuadores elétricos, em conjunto com a empresa Coester Automação S.A.

Espera-se, com a implantação desse sistema: quantificar a degradação do atuador elétrico; realizar uma predição da vida útil; e antecipar falhas inesperadas, por exemplo.

O sistema é composto de três blocos distintos: modelo matemático; ferramentas de processamento de sinais, de extração das características e de avaliação do desempenho; e classificação e predição de falhas, como visto na Fig. 1.

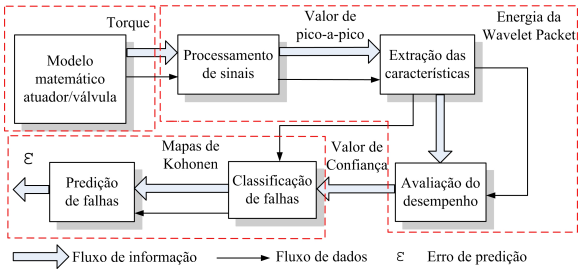



Figura 1: Principais etapas do sistema de classificação e predição de falhas.

As principais etapas para a implantação desse sistema de predição de falhas nos atuadores são:

1. Desenvolvimento de um modelo matemático;
2. Simulação das principais falhas;
3. Processamento de sinais;
4. Extração das características;
5. Avaliação da degradação do desempenho;
6. Classificação e predição de falhas;
7. Prototipação em um sistema embarcado.

Inicialmente, desenvolveu-se o modelo matemático do atuador/válvula. A seguir, foram realizadas diversas simulações de comportamento normal e de falha. Após, uma série de análises da degradação do desempenho dessas válvulas, usando as ferramentas desenvolvidas pelo IMS, foram efetuadas. Por fim, uma classificação das falhas e predição da vida útil dos atuadores, foi realizada.

A ferramenta de processamento de sinais e extração das características usada foi a transformada Wavelet Packet. A avaliação de desempenho foi feita através do método de Regressão Logística. Os Mapas Auto-organizáveis foram usados para classificação e predição de falhas.

Fez-se uso de um sistema embarcado de propósito mais ou menos geral, o Watchdog Agent, pois esse apresenta imunidade a ruído, baixo consumo, diversidade de interfaces, ferramentas matemáticas embutidas, opera em tempo real, e possui facilidade de comunicação com a central de controle, por exemplo.

O presente trabalho é organizado da seguinte forma: a seção 2 apresenta o modelo do conjunto atuador/válvula; na seção 3 são descritas as ferramentas de processamento de sinais e de extração das características utilizadas; na seção 4 os métodos de avaliação de desempenho são apresentados; na seção 5 os Mapas Auto-organizáveis são abordados; na seção 6 são visualizados os resultados das simulações; e, por fim, na seção 7 são apresentadas as conclusões finais e os trabalhos futuros, além dos agradecimentos e referências bibliográficas.

## 2 Modelagem do Atuador

O modelo de atuador CSR25 da Coester e uma válvula tipo gaveta estão sendo utilizados como estudo de caso neste trabalho.

Para avaliar corretamente o comportamento desse atuador, da válvula, e da tubulação nas mais variadas situações é necessário conhecer, se possível, o comportamento de cada componente do conjunto atuador, válvula e tubulação.

Além disso, é necessário obter um conjunto de equações diferenciais e algébricas que represente corretamente o comportamento do atuador, válvula, e tubulação.

### *2.1 Modelo de Atuador Elétrico*

O modelo CSR25 é composto por um conjunto elétrico e mecânico. Seu invólucro apresenta uma carcaça em ferro fundido nodular com tampas em alumínio.

As soluções da linha CSR se aplicam a válvulas tipo gaveta, globo e outros equipamentos similares.

Dentre as principais partes dos atuadores da Coester, Fig. 2, pode se destacar: motor elétrico, cadeia mecânica, acoplamento de saída, sensor de torque, sensor de posição e movimento.

### *2.2 Forças Envolvidas*

No comportamento do atuador elétrico existe uma parte estática que está associada com a determinação da abertura associada com a vazão para essa abertura (ou posição da haste) e outra dinâmica que corresponde a transmissão do torque para que ocorra o movimento do obturador da válvula e por conseguinte a vazão para o processo.

Na modelagem do sistema há uma série de forças que devem ser levadas em consideração, para que ocorra o movimento de abertura/fechamento do obturador da válvula, como visto na Fig. 2, onde a força do atuador é transmitida pelo motor.

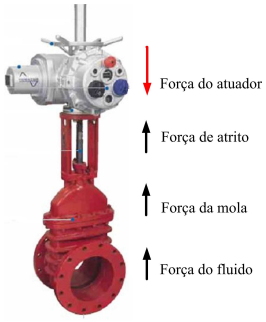



Figura 2: Diagrama de forças presentes no modelo de atuador/válvula/tubulação.

### *2.3 Sistema de Equações*

Foi feita a modelagem dos principais componentes do conjunto em um sistema de equações não-lineares.

Uma série de considerações físicas foram adotadas na modelagem para simplificar a representação e conseqüentemente o esforço computacional exigido.

Sendo assim, escolheu-se um modelo de terceira ordem para o motor de indução, pois esse representa bem tanto as condições de regime permanente como os transitórios.

Do ponto de vista da modelagem do sistema, composto por um motor assíncrono, sistema de engrenagens, válvula e tubulação, foi considerada a dinâmica do motor e da haste da válvula, enquanto o sistema de engrenagens e a tubulação foram descritos por relações estáticas, Equações (1), (2), e (3).

As equações diferenciais [1] que descrevem o motor assíncrono (tensões e escorregamento), a posição, velocidade e a aceleração da haste são dadas por:

$$\dot{s} = \frac{1}{2H}(T_e - T_m) \tag{1a}$$

$$\dot{V}_d^{'} = \frac{-1}{T_0}\left[V_d^{'} - I_{qs}(X_s - X_s^{'})\right] + s\omega_s V_q^{'} \tag{1b}$$

$$\dot{V}_q^{'} = \frac{-1}{T_0}\left[V_q^{'} + I_{ds}(X_s - X_s^{'})\right] - s\omega_s V_d^{'} \tag{1c}$$

$$\dot{a} = v_a \tag{1d}$$

$$\ddot{a} = a_a \tag{1e}$$

As equações algébricas que descrevem o motor assíncrono, a força exercida pelo fluido, a força de atrito, a força da mola, e a força e torque transmitido para a haste são [2]:

$$V_{ds} = V_d^{'} - R_s\, I_{ds} + X_s^{'}\, I_{qs} \tag{2a}$$

$$V_{qs} = V_q^{'} - R_s\, I_{qs} - X_s^{'}\, I_{ds} \tag{2b}$$

$$T_e = V_d^{'}\, I_{ds} + V_q^{'}\, I_{qs} \tag{2c}$$

$$T_h = -T_m\, K_R\, T_{mb} \tag{2d}$$

$$F_h = \frac{T_h}{R_h \cos\theta} \tag{2e}$$

$$F_f = \frac{W^2 A_v}{\rho\, N_R^2\, (100 - a)^2\, C_v^2} \tag{2f}$$

$$F_a = C_a\, v_a \tag{2g}$$

$$F_m = K_m\, a \tag{2h}$$

$$a_a = \frac{1}{M_h}\Big(F_h - F_f - F_a - F_m\Big) \tag{2i}$$

Ainda, há uma série de equações auxiliares dadas por:

[1] Modelo de $5^a$ ordem

[2] $\theta$=$90^o$

$$T_0 = \frac{L_r + L_m}{\omega_s R_r} \quad (3a)$$

$$X_s = L_s + L_m \quad (3b)$$

$$X_s^{'} = L_s + L_m - \frac{L_m^2}{L_r + L_m} \quad (3c)$$

$$K_R = K_1 K_2 K_3 \quad (3d)$$

### *2.4 Simulação das falhas*

Fez-se necessário, na fase de simulação, uma ferramenta que fosse aberta e que permitisse incluir distintos modelos de válvulas e atuadores.

As rotinas dessa ferramenta deveriam apresentar uma flexibilidade na alteração da estrutura e da modelagem do sistema, permitindo a simulação de qualquer sistema e a análise de vários parâmetros, atuadores e válvulas, bem como de diversas situações que possam ocorrer.

Assim, utilizou-se um simulador de dinâmica de sistemas não-lineares, adaptado de (Gonçalves, 2004), para simular o modelo.

Esse simulador foi desenvolvido com base no Matlab, pois esse apresenta as características mencionadas anteriormente.

O modelo proposto ainda não está validado. Contudo, já é possível realizar simulações de comportamento normal e das conseqüências das principais falhas observadas no conjunto, tais como um aumento gradual do torque devido a alterações nos parâmetros do modelo, e assim observar a curva de degradação.

Foram realizadas uma série de simulações onde, inicialmente, gerou-se uma série de curvas de torque para o modelo de atuador CSR25 cujo comportamento era normal, Fig. 3 (torque máximo igual a 250Nm ou 1pu).

Após, efetuaram-se outras simulações de comportamento de degradação e de falha. Nessas simulações, aumentou-se gradualmente certos parâmetros do modelo.

Como conseqüência, o valor de torque para abrir/fechar a válvula aumentou até atingir o valor de 275Nm (10% de sobretorque, ou 1.1 pu) definido como máximo valor de sobretorque admissível.

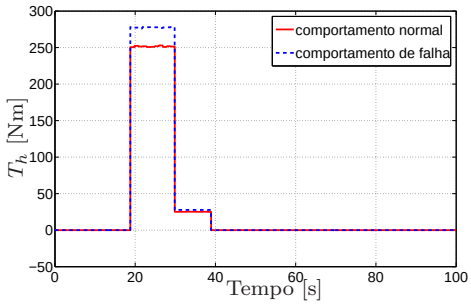



Figura 3: Curvas de torque da haste.

Foram realizados 2 tipos de simulações de comportamento de falha, ver Tabela 1.

Tabela 1: Faixa e taxa de variação de $K_2$ e $K_m$.

| Parâmetro | Faixa | Taxa |
|---|---|---|
| $K_2$ | 11.00 - 12.00 | 0.010 |
| $K_m$ | 4.215 - 5.215 | 0.010 |

Foram geradas 100 amostras para cada tipo de degradação (variando-se gradualmente os parâmetros $K_2$ e $K_m$ do modelo) e falha, e 100 para dados normais.

As falhas do tipo $K_2$ correspondem a falhas no segundo sistema de redução cinemática. Mais especificamente, trata-se da quebra do sem-fim.

A falha do tipo $Km$ corresponde a degradação da mola, equivalente a perda de elasticidade da mola, alterando a sua ação.

## 3 Ferramentas de Processamento de Sinais e Extração das Características

Os sinais de torque do atuador foram analisados utilizando a transformada Wavelet Packet (Qiu et al., 2006).

Os valores de densidade espectral, divididos em bandas, obtidas a partir da transformada Wavelet Packet foram obtidos a seguir.

### *3.1 Transformada Wavelet Packet*

A transformada Wavelet Packet é uma generalização do conceito da transformada Wavelet discreta. Nessa, o sinal também é dividido em coeficientes de aproximação e detalhamento sucessivamente.

Contudo, ao contrário da transformada Wavelet, os coeficientes de detalhamento também são sucessivamente divididos, decompondo o sinal em forma de árvore binária, chamada Wavelet Packet.

Essa técnica foi usada para extrair características relevantes de sinais, melhorando o desempenho dos classificadores ao obter características importantes por meio da decomposição do sinais em distintas bandas de freqüências, com diferentes resoluções.

### *3.2 Energia da Wavelet Packet*

A energia total de um sinal pode ser decomposto em um somatório de componentes de energia da WP que correspondem a diferentes bandas de freqüência.

Neste trabalho, fez-se se uso da energia das componentes da Wavelet Packet, $E$, para realizar a extração das características dos sinais de torque e a classificação de falhas, com $E = \mathbf{X} = (X_1, X_2, \ldots, X_{2N})$, onde $N$ é o número de bandas de freqüência, $N = 10$.

## 4 Avaliação do Desempenho

Os valores de Energia foram utilizados como entradas para o algoritmo do método de Regressão Logística, que foi usado para realizar a avaliação do desempenho do atuador.

### *4.1 Método de Regressão Logística*

O método de Regressão Logística faz parte de uma categoria de modelos estatísticos chamados de Modelos Generalizados Lineares (Djurdjanovic et al., 2003).

Esse método permite obter uma saída discreta, bem como uma classificação, em um grupo de conjunto de dados que pode ser contínuo, discreto, ou binário.

Geralmente a resposta possui dois estados como: presença/ausência ou normal/falha. A Regressão Logística tenta ajustar um mapeamento do espaço de dimensão $N$ para um espaço de saída de uma única dimensão.

O estado do sistema é medido através de um indicador de semelhança entre o comportamento normal (e de falha) e o comportamento recentemente observado de equipamentos e sistemas, conhecido como Valor de Confiança (VC).

A avaliação de desempenho, e o cálculo do VC, de peças e equipamentos feita pelo WA é realizada extraindo-se as características de degradação e de comportamento normal dos dispositivos a ele conectados.

Assim, pode-se dizer que VC é um indicador quantitativo da qualidade do sistema. Esse, é determinado a partir da análise dos sinais de desempenho observados durante o funcionamento e uso do sistema em questão.

O VC varia de zero a um, onde um valor mais elevado indica que o desempenho está mais perto do normal e um valor mais próximo do zero indica uma maior proximidade da ocorrência de algum tipo de falha, como visto na Fig. 4.

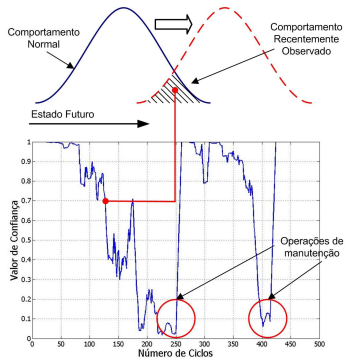



Figura 4: Gráfico do valor de confiança.

Conforme o equipamento degrada, os sinais atuais de desempenho do equipamento vão se diferenciando dos sinais de comportamento normal, reduzindo o VC.

O Valor de Confiança, quando se usa o método de Regressão Logística, é obtido a partir da seguinte expressão:

$$CV(\mathbf{X}) = \frac{1}{1 + e^{-(\lambda_0 + \lambda_1 X_1 + \ldots + \lambda_k X_k)}} \qquad (4)$$

Após os parâmetros do modelo serem obtidos a partir de amostras de treinamento, o CV do sistema pode ser calculado.

## 5 Classificação das falhas: Mapas Auto-organizáveis

Os Mapas Auto-organizáveis (MAO), ou mapas de Kohonen, pertencem a uma classe de redes neurais baseadas em um paradigma de aprendizagem não supervisionado e que utilizam técnicas de competição, cooperação e adaptação.

Nos MAO, existe uma interação e competição entre os neurônios dentro de certa vizinhança. Como resultado, apenas um neurônio é declarado como vencedor (Kohonen et al., 1995).

Formalmente, a principal função de um mapa auto-organizável é realizar um mapeamento de dados de entrada dispostos em um espaço $\Re^n$ em uma matriz bidimensional, formando uma rede, mantendo a topologia ordenada.

O vetor de entradas (energia das componentes da Wavelet Packet) $\mathbf{X}$ é selecionado aleatoriamente no espaço de entrada (onde $\mathbf{X} = [X_1, X_2, \ldots, X_{2N}]^T$) e o vetor peso sináptico do neurônio $j$ será representado por $\mathbf{W}_j$, $\mathbf{W}_j = [W_{j1}, W_{j2}, \ldots, W_{j2N}]^T$, com $j = 1, 2, \ldots, L$, onde $L$ é o número total de neurônios na grade.

O processamento interno dos algoritmos do MAO pode ser dividido, simplificadamente, em três etapas distintas:

- Inicialização: os pesos sinápticos da grade são inicializados.
- Treinamento: é realizada a aquisição do conhecimento pelo mapa;
- Recuperação: os dados de entrada são classificados no mapa.

A etapa de treinamento, pode ser dividida, simplificadamente, em três fases distintas:

1. Competição: nessa fase procura-se encontrar o melhor casamento (minimizando a distância euclidiana entre os vetores $\mathbf{X}$ e $\mathbf{W}_j$) do vetor de entrada com os vetores de pesos sinápticos.
2. Cooperação: o neurônio vencedor determina a localização espacial de uma vizinhança topológica de neurônios excitados, fornecendo assim a base para a cooperação entre os neurônios vizinhos.
3. Adaptação: o vetor de pesos sinápticos $\mathbf{W}_j(n)$, no tempo $n$, é atualizado de forma a se aproximar mais do vetor $\mathbf{X}$.

Na etapa de recuperação, são avaliadas as formações topológicas dos agrupamentos de neurônios. O método mais usado para realizar essa avaliação é a Matriz de Distâncias Unificada (MDU).

O resultado gerado a partir da MDU sobre o mapa é uma imagem em duas (ou três dimensões), onde o nível de intensidade de cada pixel corresponde a uma distância calculada.

Na imagem, a coloração dos pixels varia de acordo com a intensidade de cada componente da MDU. Regiões que apresentam baixos valores (vales) agrupam neurônios com padrões similares, e as regiões com valores altos (picos) correspondem a fronteiras entre os agrupamentos.

Assim, as condições de comportamento normal, de degradação, e de falha de um sistema podem ser visualizadas em distintos agrupamentos.

## 6 Resultados

Os resultados gráficos dos VC e dos MAO, para cada um dos tipos de falhas, são vistos na Fig. 5. Pode-se observar que os gráficos do VC obtidos a partir das ferramentas WP e RL, retratam fielmente as condições de comportamento normal, degradação, e falha simuladas.

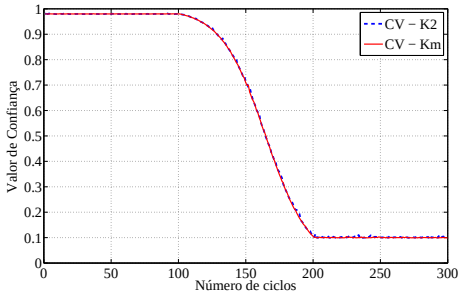



Figura 5: VC - falha do tipo $K_2$ e $K_m$.

Através do resultado da recuperação dos MAO, Fig. 6, pode-se observar as três regiões de comportamento: normal, degradação e falha.

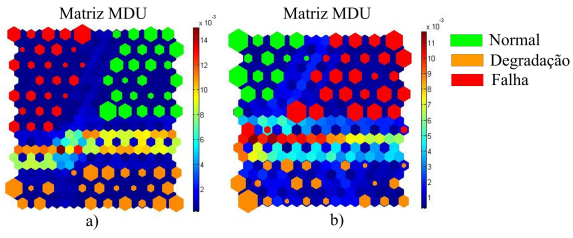



Figura 6: MAO - a) falha do tipo $K_2$ e b) $K_m$ .

Pode-se observar na Fig. 6 que os dados foram mapeados corretamente. São vistas, na figura, três regiões de comportamento bem distintas. Também é visível a concentração dos dados de degradação no vale inferior e dos dados normais e de falha na parte superior do mapa.

Além disso, os dados de falhas, $K_2$ e $K_m$, foram mapeados em diferentes lugares no mapa, o que nos permite fazer uma classificação quanto ao tipo de falha que ocorreu.

## 7 Conclusão

Este trabalho abordou algumas das atividades desenvolvidas para a implantação de um sistema de manutenção inteligente embarcado nos atuadores elétricos da empresa Coester Automação S.A.

Para tal, fez-se uso das técnicas de processamento de sinais e métodos estatísticos desenvolvidas pelo IMS, mas especificamente a transformada WP e o método de RL. Além disso, foi realizado o desenvolvimento de um sistema de classificação e predição de falhas utilizando-se MAO.

Os resultados gráficos dos VC e dos MAO, para dois tipos de falhas, também foram exibidos. Pode-se observar, através dos resultados, o comportamento normal, de degradação, e de falha corretamente.

Espera-se, no futuro, estimar o tempo de vida útil do atuador. Para isto, pretende-se utilizar algoritmos que tratam da análise temporal nos MAOs como o Temporal Kohonen Map ou o Recurrent Self-Organazing Map, por exemplo.

## Referências


Djurdjanovic, D., Lee, J. and Ni, J. (2003). Watchdog agent - an infotronics-based prognostic approach for product performance degradation, assessment and prediction, *Advanced Engineering Informatics* **17**: 109–125.

Gonçalves, L. F. (2004). *Contribuições para o estudo teórico e experimental de sistemas de geração distribuída*, Master's thesis, Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal de Rio Grande do Sul, Porto Alegre.

Kohonen, T., Hynninen, J., Kangas, J. and Laaksonen, J. (1995). *The self-organizing map program package*, 3.1 edn, Springer (Springer Series in Information Sciences), Rakentajanaukio, Finland.

Lee, J., Qiu, H., Ni, J. and Djurdjanovic, D. (2004). Infotronics technologies and predictive tools for next-generation maintenance systems, *International Federation of Automatic Control (IFAC)* .

Qiu, H., Lee, J., Lin, J. and Yu, G. (2006). Wavelet filter-based weak signature detection method and its application on rolling element bearing prognostics, *Journal of Sound and Vibration* **289**: 1066–1090.