Universidade Federal do Rio Grande do Norte
Centro de Tecnologia
Programa de Pós-Graduação em Engenharia
Elétrica e de Computação

Detecção e Diagnóstico de Falhas Não-supervisionados Baseados em Estimativa de Densidade Recursiva e Classificador Fuzzy Auto-evolutivo

Bruno Sielly Jales Costa

Orientador: Prof. Dr. Luiz Affonso H. Guedes de Oliveira

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica e de Computação da UFRN (área de concentração: Engenharia de Computação) como parte dos requisitos para obtenção do título de Doutor em Ciências.

Natal, RN, Julho de 2014

UFRN / Biblioteca Central Zila Mamede

Catalogação da publicação na fonte.

Costa, Bruno Sielly Jales.

Detecção e diagnóstico de falhas não-supervisionados baseados em estimativa de densidade recursiva e classificador fuzzy auto-evolutivo / Bruno Sielly Jales Costa - Natal, RN, 2014

102 f.: il.

Orientador: Prof. Dr. Luiz Affonso H. Guedes de Oliveira.

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

1. Localização de falhas (Engenharia) - Tese. 2. Detecção de falhas - Tese. 3. Diagnóstico de falhas - Tese. 4. Identificação de falhas - Tese. 4. Estimativa de densidade recursiva - Tese. 5. Classificadores evolutivos - Tese. 6. Sistemas fuzzy - Tese. 7. Aprendizagem autônoma - Tese. I. Oliveira, Luiz Affonso H. Guedes de. II. Universidade Federal do Rio Grande do Norte. III. Título.

RN/UF/BCZM 620.179.1

Detecção e Diagnóstico de Falhas Não-supervisionados Baseados em Estimativa de Densidade Recursiva e Classificador Fuzzy Auto-evolutivo

Bruno Sielly Jales Costa

Tese de Doutorado aprovada em 13 de maio de 2014 pela banca examinadora composta pelos seguintes membros:

Prof. Dr. Luiz Affonso H. Guedes de Oliveira (orientador) . DCA/UFRN

Prof. Dr. Adrião Duarte Dória Neto ..... DCA/UFRN

Prof. Dr. Daniel Aloise ..... DCA/UFRN

Prof. Dr. Plamen Parvanov Angelov ..... SCC/Lancaster University

Profª Drª Marley Maria Bernardes Rebuzzi Vellasco ..... DEE/PUC-RJ

Agradecimentos

Ao meu orientador, Prof. Luiz Affonso, pela paciência, esforço e disponibilidade no decorrer dos últimos oito anos.

Ao Prof. Plamen Angelov, por ter me acolhido tão bem em terras estrangeiras.

Ao Prof. Adrião Duarte, pelo auxílio prestado no desenvolvimento do trabalho.

À minha noiva e companheira Larissa, por ter sido o pilar que me sustentou até este dia.

Aos amigos Marcus, Francisco, Adriano, Pedro, Rodolfo, Berg, Agamenon, Clauber, Rodrigo, Karla, Ivanilson, Pablo, Pauleany, João Paulo, Breno, John, Matheus, Thiago e Rêmullo, que, de maneira positiva, influenciaram essa jornada em dado momento.

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

À minha família, pelo apoio durante essa jornada.

À CAPES, pelo apoio financeiro.

Resumo

Este trabalho propõe um algoritmo de dois estágios para detecção e identificação de falhas, em tempo real, em plantas industriais. A proposta baseia-se na análise de características selecionadas utilizando estimativa de densidade recursiva e um novo algoritmo evolutivo de classificação. Mais especificamente, a abordagem proposta para detecção é baseada no conceito de densidade no espaço de dados, o que difere da tradicional função densidade de probabilidade, porém, sendo uma medida bastante útil na detecção de anormalidades/*outliers*. Tal densidade pode ser expressa por uma função de Cauchy e calculada recursivamente, o que torna o algoritmo computacionalmente eficiente, em termos de processamento e memória, e, dessa maneira, apropriado para aplicações *on-line*. O estágio de identificação/diagnóstico é realizado por um classificador baseado em regras *fuzzy* capaz de se auto-desenvolver (evolutivo), chamado de AutoClass, e introduzido neste trabalho. Uma propriedade importante do AutoClass é que ele é capaz de aprender a partir "do zero". Tanto as regras *fuzzy*, quanto o número de classes para o algoritmo não necessitam de pré-especificação (o número de classes pode crescer, com os rótulos de classe sendo adicionados pelo processo de aprendizagem *on-line*), de maneira não-supervisionada. Nos casos em que uma base de regras inicial existe, AutoClass pode evoluir/desenvolver-se a partir dela, baseado nos dados adquiridos posteriormente. De modo a validar a proposta, o trabalho apresenta resultados experimentais de simulação e de aplicações industriais reais, onde o sinal de controle e erro são utilizados como características para os estágios de detecção e identificação, porém a abordagem é genérica, e o número de características selecionadas pode ser significativamente maior, devido à metodologia computationalmente eficiente adotada, uma vez que cálculos mais complexos e armazenamento de dados antigos não são necessários. Os resultados obtidos são significativamente melhores que os gerados pelas abordagens tradicionais utilizadas para comparação.

Palavras-chave: Detecção de falhas, diagnóstico de falhas, identificação de falhas, estimativa de densidade recursiva, classificadores evolutivos, sistemas *fuzzy*, aprendizagem autônoma.

Abstract

In this work, we propose a two-stage algorithm for real-time fault detection and identification of industrial plants. Our proposal is based on the analysis of selected features using recursive density estimation and a new evolving classifier algorithm. More specifically, the proposed approach for the detection stage is based on the concept of density in the data space, which is not the same as probability density function, but is a very useful measure for abnormality/outliers detection. This density can be expressed by a Cauchy function and can be calculated recursively, which makes it memory and computational power effcient and, therefore, suitable for on-line applications. The identification/diagnosis stage is based on a self-developing (evolving) fuzzy rule-based classifier system proposed in this work, called AutoClass. An important property of AutoClass is that it can start learning "from scratch". Not only do the fuzzy rules not need to be prespecified, but neither do the number of classes for AutoClass (the number may grow, with new class labels being added by the on-line learning process), in a fully unsupervised manner. In the event that an initial rule base exists, AutoClass can evolve/develop it further based on the newly arrived faulty state data. In order to validate our proposal, we present experimental results from a level control didactic process, where control and error signals are used as features for the fault detection and identification systems, but the approach is generic and the number of features can be significant due to the computationally lean methodology, since covariance or more complex calculations, as well as storage of old data, are not required. The obtained results are significantly better than the traditional approaches used for comparison.

Keywords: Fault detection, fault diagnosis, fault identification, recursive density estimation, evolving classifiers,fuzzy, systems, autonomous learning.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Durante as últimas quatro décadas os sistemas *fuzzy* têm sido utilizados com sucesso na resolução de um amplo escopo de problemas, em diferentes aplicações industriais. Dentre as diferentes áreas de estudo, devem ser mencionados os trabalhos de Kruse et al. (1994) na área de ciência da computação, Pedrycz & Gomide (2007) na área de engenharia industrial, Lughofer (2011) na área de mineração de dados, Abonyi (2003) na área de controle de processos, Kerre & Nachtegael (2000) na área de processamento de imagens e Nelles (2001) na área de identificação de sistemas.

Uma das principais vantagens de um sistema *fuzzy*, quando comparado a outras técnicas de manipulação de informações "imprecisas", tais como as redes neurais, é que a sua base de conhecimento, que é composta por regras de inferência, é bastante simples de examinar e entender. Esse formato de regras também facilita a manutenção e atualização da estrutura do sistema. O uso de modelos *fuzzy* para expressar o comportamento de um sistema real de uma maneira inteligível é uma tarefa de grande importância, uma vez que a principal "filosofia da teoria de conjuntos *fuzzy* é servir de ponte entre o entendimento humano e o processamento da máquina" (Casillas et al., 2003). Sobre esse assunto, a interpretabilidade de sistemas *fuzzy* foi objeto de estudo de diversos autores, tais como Casillas et al. (2003), Lughofer (2013), Gacto et al. (2011), e Zhou & Gan (2008).

A integração da teoria de conjuntos *fuzzy* com técnicas de inteligência artificial passaram a possibilitar o auto-desenvolvimento de sistemas baseados em regras, a partir das informações coletadas de um processo e das respostas obtidas para as entradas determinadas. Surgiram, daí, os modelos adaptativos e evolutivos, capazes de não só aprenderem o comportamento, mas também acompanharem as mudanças na dinâmica do processo (Graham & Newell, 1989).

Detecção e diagnóstico de falhas (DDF - do inglês, *fault detection and diagnosis*) é, sem dúvidas, uma das tarefas dentre o escopo de aplicações industriais onde a teoria de sistemas *fuzzy* encaixa-se melhor (Mendonça et al., 2006), (Dash et al., 2003). Como exemplos concretos de aplicações de sistemas *fuzzy* no contexto de DDF, podemos mencionar os trabalhos de Serdio et al. (2014) e Angelov et al. (2006), onde sistemas *fuzzy* são listados entre as abordagens de melhor performance na detecção de falhas baseada em resíduo, e Lemos et al. (2013) e Laukonen et al. (1995), que apresentam abordagens *fuzzy* para uso em no contexto de DDF baseada em mineração de dados.

Enquanto a DDF ainda é amplamente executada por operadores humanos, o núcleo da tarefa consiste, a grosso modo, de uma sequência de "passos de inferência" baseados nos dados coletados, como poderemos ver no decorrer deste trabalho. A inferência *fuzzy* pode ser aplicada em todos os passos e de diversas maneiras diferentes na tarefa de DDF.

Aplicações de técnicas de DDF em ambientes industriais estão crescendo consideravelmente, de modo a aprimorar a seguranca operacional, bem como reduzir os custos relacionados a paradas não programadas. A importância da pesquisa de DDF na área de engenharia de controle e automação está ligada ao fato de que a rápida detecção de uma falha em ocorrência, enquanto o sistema ainda está operando em uma região controlável, geralmente previne ou, no mínimo, reduz as perdas de produtividade e riscos à saúde (Venkatasubramanian et al., 2003c).

Diversos autores têm, recentemente, contribuído para o desenvolvimento da área de estudos em DDF, através de estudos exaustivos, compilações e revisões da literatura. Korbicz (2004) cobre os fundamentos da tarefa de DDF baseada em modelos, direcionados a engenheiros industriais, cientistas e acadêmicos focados em confiabilidade e demais problemas em processos industriais de segurança crítica. Chiang et al. (2001) apresenta o embasamento teórico e técnicas práticas para o monitoramento de processos baseados em dados, incluindo diversas abordagens baseadas na análise de componentes principais, análise de discriminantes lineares, mínimos quadrados parciais, análise de variáveis canônicas, estimativa de parâmetros, métodos baseados em observadores de estado, relações de paridade, análise causal, sistemas especialistas e reconhecimento de padrões. Isermann (2006) introduz no campo de sistemas de DDF diversos métodos de performance comprovada em variadas aplicações práticas, incluindo detecção de falhas baseada em modelo e em sinais, diagnóstico de falhas com classificação e métodos de inferência, estratégias de controle tolerante à falha, além de diversos resultados de simulações práticas e experimentais. Witczak (2013) apresenta uma seleção de métodos de DDF e estratégias de controle tolerante à falha para sistemas não-lineares, desde estimativa de estados até estratégias modernas de *soft computing*. Por último, mas não menos importante, Simani et al. (2002) foca na identificação de modelos orientados às abordagens analíticas de DDF, inclusindo estudos de caso utilizados para demonstrar a aplicação de cada técnica.

Com a crescente complexidade dos procedimentos e escopo das atividades industriais, a demanda por eficiência e qualidade do produto, o gerenciamento de eventos anormais (GEA) vem se tornando um campo de estudos bastante desafiador (Venkatasubramanian et al., 2003c). O operador humano desempenha um papel crucial nesse assunto, uma vez que, segundo estudos recentes, as pessoas responsáveis pelo GEA, muitas vezes, tomam decisões incorretas. Estatísticas industriais mostram que de 70% a 90% dos acidentes em ambiente industrial são causados por erros humanos (Wang & Guo, 2013). Além disso, existe muito mais por trás da necessidade da automação de processos de DDF; por exemplo, em diversos ambientes industriais, os esforços dos operadores no sentido de uma supervisão completa de todas as variáveis e estados são bastante elevados, o que resulta em altos custos para a empresa. Algumas vezes, uma supervisão manual é simplesmente impossivel, por

exemplo, em sistemas amplamente distribuídos (Chen et al., 2006).

No passado, a supervisão automática de processos era comumente realizada por simples *testes de limite* (do inglês, *threshold checks*) das váriáveis mais importantes do processo (por exemplo temperatura, pressão, força, nível, velocidade) (Isermann, 2005). Normalmente, alarmes eram disparados caso o valor limite de uma ou mais variáveis fosse excedido e, ou o operador ou o sistema de proteção automático, deveriam agir de modo minimizar os danos ao sistema. Embora muitas vezes esse procedimento seja suficiente para previnir danos maiores ao sistema, pelo fato de a falha ser detectada tardiamente, diagnósticos específicos não são possíveis. Nos trabalhos recentes na área de DDF, são apresentadas soluções inteligentes, capazes de fornecer um diagnóstico avançado, através do uso de modelos matemáticos, físicos, abordagens estatísticas e de inteligência artificial. Ainda segundo Isermann (2005), os objetivos dos métodos para DDF, dentre outros, devem incluir:

- detecção precoce de pequenas falhas com comportamento abrupto ou incipiente;
- diagnósticos de falhas em atuadores, sensores, processos e componentes;
- detecção de falhas em malha fechada;
- manutenção e reparo baseados nas condições do processo;
- base para gerenciamento de falhas;
- base para controle tolerante a falhas e sistemas reconfiguráveis.

Sabendo-se que a tarefa de GEA é responsabilidade primária do operador, um sistema de DDF deve trabalhar no auxílio à tomada de decisões. Uma vez que a quantidade de informações (variáveis monitoradas, dados em tempo real, dados históricos) a ser processada em um processo industrial é muito grande, as abordagens mais atuais de DDF tentam tirar do escopo de atividades do operador o máximo de tarefas possível, atribuindo a ele as decisões finais, baseadas na análise e processamento realizados pelo sistema automatizado. Torna-se, então, interessante a minimização de entradas/parâmetros/configurações por parte do operador, e a configuração automática do sistema a partir dos próprios dados coletados no processo é um objetivo a ser alcançado por um sistema de DDF. Além do mais, um sistema *data-driven* torna-se vantajoso para lidar com as mudanças naturais na dinâmica/ambiente do sistema no decorrer do tempo.

Nos próximos capítulos, serão apresentadas abordagens que permitem a detecção e identificação automática de novos estados de operação do processo monitorado. Tais estados podem representar alterações na dinâmica, novos *set points* ou falhas no processo. A identificação/distinção de novos estados dá-se por meio da distribuição dos dados no espaço n-dimensional, através do agrupamentos de pontos próximos, mais tarde chamados de nuvens de dados, e do cálculo de densidade de tais agrupamentos.

1.1 Objetivos da tese

O objetivo desta tese é propor um procedimento, em duas etapas, para detecção (1) e identificação/classificação (2) de falhas em sistemas dinâmicos, que pode ser utilizado nas mais diversas aplicações industriais, reais ou simuladas.

O sistema proposto é capaz de se auto-desenvolver e adaptar-se, de forma autônoma e não-supervisionada, a partir da primeira amostra de dados coletada, sem nenhum conhecimento prévio a respeito do modelo, dinâmica ou comportamento da planta, e sem a necessidade de treinamento *off-line*. Uma vez que o procedimento é executado inteiramente *on-line*, deve-se trabalhar com a ideia de recursos computacionais limitados, minimizando-se a necessidade de esforço computacional e armazenamento.

Além disso, o procedimento visa ser claro e lógico para o usuário/operador, de modo a transparecer todo o processo de inferência, evitando elementos do tipo "caixa-preta", bastante comuns em outras técnicas, tais como as redes neurais. Dessa forma, o processo deve tornar-se intuitivo e facilmente assimilado pelo operador que, a partir de então, torna-se capaz de utilizar a sua *expertise* no processo.

1.2 Motivação

A possibilidade de criação de sistemas de DDF autônomos através de recentes abordagens evolutivas, sem a necessidade de conhecimentos prévios avançados a respeito do processo, configuração de parâmetros complexos, ou grande esforço computacional, mostra-se como um promissor estímulo à pesquisa do tema em questão. As abordagens apresentadas nesta tese devem simplificar de forma substancial a tarefa de DDF, a partir do ponto de vista do operador, bem como apresentar resultados superiores, quando comparadas a outras técnicas existentes e consolidadas.

As aplicações propostas neste trabalho são sempre ligadas ao ambiente industrial, por tratar-se de um dos maiores demandantes de novas tecnologias computacionais e matemáticas. Sendo assim, o crescimento industrial e a inserção cada vez maior de ferramentas computacionais de automação na indústria são fatores sempre motivadores de trabalhos na área de computação e automação.

1.3 Contribuições

Dentre as principais contribuições deste trabalho, podem ser destacadas:

- O desenvolvimento de um novo procedimento para detecção de falhas em plantas industriais, *on-line*, baseado no já existente algoritmo de estimativa de densidade recursiva, proposto por Angelov et al. (2008). Apesar de anteriormente utilizado em uma aplicação de detecção de falhas (Kolev et al., 2013), o algoritmo proposto é estruturalmente diferente e bem mais genérico.
- O desenvolvimento de um novo procedimento para identificação/classificação de falhas, *on-line*, de forma autônoma e não supervisionada, proporcionando

a criação automática de novas classes/grupos de falha, à medida que novos pontos de operação do processo são detectados.
- A proposição de um algoritmo, em dois estágios independentes, para detecção e diagnóstico de falhas, baseado nos procedimentos desenvolvidos, genérico e aplicável aos mais diversos processos industriais.
- O enfoque no aspecto evolutivo dos algoritmos propostos, permitindo uma profunda adaptação do sistema de detecção às condições do processo, que podem (naturalmente ou não) evoluir no decorrer do tempo.
- A simplificação da tarefa de DDF, com a redução da necessidade de intervenção do operador humano, eliminando-se a configuração de parâmetros demasiados e desnecessários.
- A validação das propostas através de aplicações industriais simuladas e reais, considerando, assim, não somente o processo em seu modelo ideal, mas também as características intrínsecas aos ambientes industriais, tais como ruído, inércia e outras variações não previstas.
- O estudo comparativo entre os procedimentos propostos e alguns dos já existentes e bem conhecidos algoritmos presentes na literatura.

1.4 Estrutura da tese

A presente tese de Doutorado é dividida em 7 capítulos, além desta introdução. O Capítulo 2 introduz brevemente os principais conceitos da teoria dos conjuntos *fuzzy*, bem como os sistemas de inferência baseados na lógica *fuzzy*. No Capítulo 3 é apresentada uma revisão dos conceitos relacionados à tarefa de detecção e diagnóstico de falhas em ambientes industriais. O Capítulo 4, por sua vez, traz uma revisão da literatura a partir das estratégias previamente propostas e relacionadas a este trabalho. No Capítulo 5, a abordagem de trabalho proposta é definida e detalhada. O Capítulo 6 detalha a configuração experimental utilizada. O Capítulo 7 apresenta os resultados obtidos com a utilização da abordagem proposta e a comparação de tais resultados com abordagens conhecidas. Por fim, o Capítulo 8 conclui o trabalho.

Capítulo 2

Sistemas Baseados em Regras Fuzzy

O conceito de conjuntos *fuzzy*, introduzido por Zadeh (1965), possibilitou o desenvolvimento de sistemas para solucionar problemas de diversas áreas de aplicações, tais como a automação e controle, a classificação de dados, a análise de decisão e a visão computacional (Silva, 2010).

Tal conceito quebrou os paradigmas da lógica booleana, onde uma afirmação deve ser 100% verdadeira ou 100% falsa. Ela aproxima-se do comportamento humano, possibilitando combinações de valores intermediários como forma de melhor representar um conceito.

A lógica *fuzzy* tem sido cada vez mais usada em sistemas que utilizam informações fornecidas por seres humanos para automatizar procedimentos quaisquer, como, por exemplo, no controle de processos e no auxílio à decisão (Carulo, 2008). Eles foram utilizados, com sucesso, em algumas aplicações que se tornaram exemplos clássicos. A primeira aplicação a tornar-se pública foi Mamdani & Assilian (1975), na Inglaterra, onde o professor Mamdani implementou um controle de uma máquina a vapor baseado em lógica *fuzzy*.

A lógica *fuzzy* é baseada no uso de aproximações, ao contrário da exatidão com que estamos naturalmente acostumados a trabalhar na resolução de problemas. O princípio fundamental da teoria *fuzzy*, a dualidade, estabelece que dois eventos opostos possam coexistir. Isto é, um elemento pode pertencer em um certo grau de pertinência a um conjunto e, em um outro grau a um outro conjunto.

2.1 Variáveis Linguísticas

Por variável linguística, entende-se uma variável cujos valores são nomes ou sentenças, ao invés de assumirem apenas valores específicos, como ocorre com variáveis numéricas.

Uma variável linguística pode ser definida pela quádrupla $\{X, \mathbf{U}, T(X), M\}$, onde definimos X como sendo o nome da variável em questão, $\mathbf{U}$ o universo de discurso, $T(X)$, como sendo um conjunto de valores para X, e M uma função que associa valores de pertinência para cada elemento do conjunto $T(X)$.

Um exemplo prático do conceito de variáveis linguísticas seria a velocidade de um carro. Poderíamos expressar tal variável de forma numérica e precisa, utilizando uma unidade conhecida como quilômetros por hora ou metros por segundo. Por outro lado, podemos expressá-la também por meio de palavras (lento, veloz, muito veloz etc.), que expressam de forma subjetiva o nível da velocidade.

O valor de uma variável linguística, ou valor *fuzzy*, é uma sentença composta por um termo primário (no exemplo anterior: *baixa, média, alta*), por conectores lógicos *(e, ou, não)* e por modificadores *(pouco, muito, extremamente)*. Podemos criar novos valores *fuzzy* adicionando conectores lógicos e modificadores aos valores *fuzzy* primários (por exemplo, podemos dizer que a velocidade do carro é *"não muito alta"*). A Figura 2.1 ilustra a variável linguística velocidade com os valores *fuzzy* dados por {*Muito Baixa, Baixa, Média, Alta, Muito Alta*}.

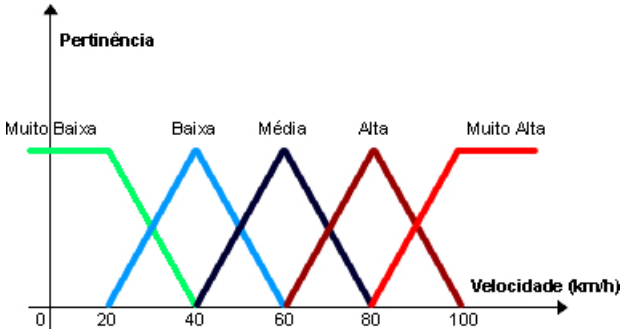

Figura 2.1: Valores *fuzzy* para a variável velocidade

Variáveis linguísticas servem para a caracterização de fenômenos complexos, quando utiliza-se uma abordagem vaga e são identificadas através de seus valores *fuzzy*, ou servem para uma caracterização quantitativa e precisa, na utilização de funções de pertinência. Esta característica dual das variáveis linguísticas torna-as convenientes para abordar problemas tanto de forma qualitativa quanto quantitativa.

2.2 Regras Se-Então

A principal característica dos sistemas *fuzzy* é o uso de regras do tipo Se-Então. Um modelo formado por um conjunto de regras deste tipo serve para caracterizar um sistema, de acordo com uma gama de dados de entrada e saída. Esta forma de raciocínio aproximado define-se da seguinte maneira:

(Antecedente) Se x é A
(Consequente) Então y é B

sendo x a variável de entrada, y a variável de saída, A e B os valores linguísticos associados aos conjuntos *fuzzy* que descrevem a variável em questão. Neste caso, a regra "Se x é A Então y é B" associa uma função de pertinência $f : A \rightarrow B(x, y)$, que mede o grau de veracidade de uma implicação.

Na definição de regras do tipo Se-Então, uma determinada regra, dentro de um conjunto de regras, é ativada no momento em que a variável em questão satisfaz as suas condições. Caso a regra seja ativada, a inferência gerada para aquele subsistema será igual à consequência da regra. Note que em sistemas *fuzzy*, para que uma regra seja ativada, basta que a premissa atinja um grau de pertinência não nulo, gerando assim uma inferência compatível com o grau de satisfação da regra em questão.

Uma regra *fuzzy*, em diversos casos, pode possuir mais de um antecedente (do inglês, *multiple inputs*), como também mais de uma implicação (do inglês, *multiple outputs*). O exemplo a seguir ilustra esta afirmação:

Se x é A ou y é B ou v é C
Então w é D e z é E

Neste caso, o método de inferência torna-se mais complexo. A conclusão depende tanto da premissa de que x é A, da premissa de que y é B quanto da premissa de que v é C. Outra peculiaridade é que, simultaneamente, duas variáveis de saída são afetadas desta vez.

Existem diversas configurações que definem como os antecedentes interagem entre si e com o consequente da regra para produzir uma conclusão. Tais configurações são chamadas mecanismos de inferência.

2.3 Funções de pertinência

Um conjunto *fuzzy* F é caracterizado por uma função de pertinência (função também conhecida como função de compatibilidade) $f : F(x)$, que associa a cada elemento do universo de discurso $\mathbf{U}$ um número no intervalo real $[0, 1]$ (Mozelli, 2008). Sua representação é feita por meio de um conjunto de parâmetros que modelam o comportamento da função.

$$F = \{(x, fF(x))\} \text{ para } x \in \mathbf{U}$$

O valor atribuído pela função característica $f : F(x)$ indica o grau de pertinência de x ao conjunto F. Aproximando-se da unidade, maior é o grau de pertinência de x em F.

Na literatura, podemos encontrar diversos modelos de funções de pertinência. O formato da função contribui no comportamento das saídas do sistema *fuzzy*, por isso, um estudo geralmente é necessário para definir qual o tipo de função de pertinência é mais adequado, bem como os parâmetros de cada função. As principais funções de pertinência encontradas na literatura são as *triangulares*, *trapezoidais* e as *gaussianas* (Pedrycz, 1993).

2.4 Sistemas de inferência *fuzzy*

Os sistemas de inferência *fuzzy* são ferramentas computacionais utilizadas nas mais diversas áreas da engenharia e da ciência, e agregam os conceitos de conjuntos

fuzzy, variáveis lingüísticas e raciocínio aproximado, processando dados por meio de mecanismo de inferência. A estrutura básica de um sistema de inferência é mostrada na Figura 2.2.

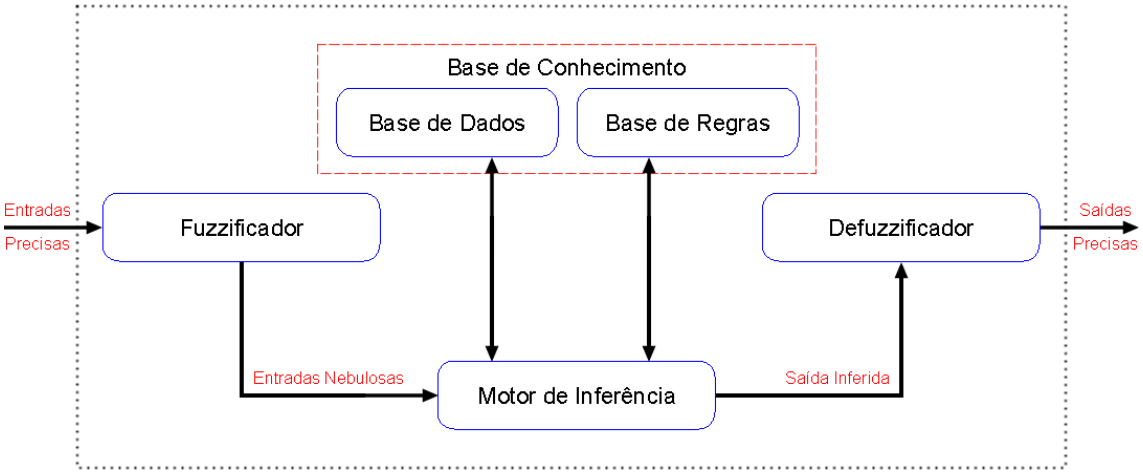

Figura 2.2: Sistema de inferência *fuzzy*

No estágio de inferência, ocorrem as operações com conjuntos *fuzzy* ao longo de regras Se-Então para processar, por meio de um mecanismo de inferência, as informações da entrada e produzir uma conclusão.

A inferência pode ser resumida em quatro etapas. Em uma primeira instância, são avaliados os graus de compatibilidade das variáveis premissas com seus respectivos antecedentes nas regras Se-Então. Normalmente, os sistemas de inferência *fuzzy*, recebem entradas precisas. Isso ocorre devido ao fato de que geralmente essas entradas são resultados de medições ou observações. Neste caso, faz-se necessário o uso de uma interface de *fuzzificação*, que relaciona o valor preciso da entrada a um valor *fuzzy*. O valor *fuzzy* é obtido pelo fuzzificador através das funções de pertinência designadas para as variáveis de entrada do sistema.

Após a etapa de fuzzificação, um mecanismo chamado *motor de inferência* é responsável por realizar as inferências do sistema, fazendo uso neste caso de uma base de dados e de uma base de regras de inferência. Na base de dados ficam armazenadas as definições sobre discretização e normalização dos universos de discurso, e as definições das funções de pertinência dos termos *fuzzy*. A base de regras é formada pelas estruturas do tipo Se-Então previamente definidas no sistema de inferência. O motor de inferência é a entidade responsável por determinar o grau de ativação de uma regra. O grau de ativação da regra é dado pela combinação dos graus de compatibilidade das variáveis premissas com seus antecedentes, através da utilização de t-normas ou s-normas previamente definidas. Ao final da inferência, o motor de inferência gera um valor *fuzzy* para a saída.

Com base no grau de ativação determina-se a *consequência* produzida por uma determinada regra. Na maioria das vezes um sistema de inferência *fuzzy* possui mais de uma regra. Cada regra produz uma consequência e o resultado global

da etapa inferência dependerá da combinação dessas conseqüências. Esta etapa é chamada de *agregação*, a qual tem por resultado um conjunto *fuzzy*, estando este diretamente relacionado com o operador de agregação aplicado, que pode ser do tipo valor máximo, soma, métodos probabilísticos, dentre outros.

A próxima etapa é chamada de *defuzzificação*. O procedimento compreende a identificação do domínio das variáveis de saída num correspondente universo de discurso, e com a saída *fuzzy* inferida evolui-se uma saída precisa/real. Existem na literatura diversos métodos para defuzzificação, tais como centróide, bissetor e média dos máximos (Lee, 1990).

2.4.1 Modelo *fuzzy* de Mamdani

O modelo de Mamdani foi utilizado em um dos primeiros sistemas de controle que utilizavam teoria dos conjuntos *fuzzy*. Como citado anteriormente, tal modelo foi proposto em 1975 por Ebrahim Mamdani como uma tentativa de controlar uma máquina a vapor, através da síntese de um conjunto de regras de controle linguístico, obtidas da experiência com operadores humanos.

O trabalho de Mamdani foi baseado no artigo de Zadeh (1973), que tratava do estudo de algoritmos *fuzzy* para sistemas complexos e processos de decisão.

Este modelo faz parte do conjunto de modelos clássicos, e seguem estritamente os passos destacados anteriormente. A peculiaridade de sistemas que seguem os modelos clássicos, é que a saída do sistema é sempre dada por um conjunto *fuzzy*, e desta forma, sempre faz-se necessária a utilização de uma interface de defuzzificação.

2.4.2 Modelo *fuzzy* de Takagi-Sugeno

O modelo *fuzzy* de Takagi-Sugeno consiste em um sistema de inferência capaz de descrever, de forma exata ou aproximada, sistemas dinâmicos não-lineares por meio de um conjunto de sistemas dinâmicos lineares, localmente válidos, interpolados de forma suave, não-linear e convexa.

O sistema de inferência de Takagi-Sugeno é uma alternativa ao sistema proposto por Mamdani. Também chamado de modelo de Takagi-Sugeno-Kang, este modelo foi introduzido por Takagi & Sugeno (1985a), e assemelha-se ao modelo de Mamdani em diversos aspectos.

As duas primeiras etapas do processo de inferência (fuzzificação das entradas e aplicação dos operadores *fuzzy*) são exatamente iguais. A principal diferença entre os dois modelos dá-se nas funções de saída, que no modelo de Takagi-Sugeno sempre é linear ou constante.

Uma típica regra Se-Então em um sistema de inferência de Takagi-Sugeno tem a seguinte forma:

$$\textbf{Se } (x_1 \textbf{ é } V_1) \textbf{ E } \ldots \textbf{ E } (x_n \textbf{ é } V_n)$$
$$\textbf{Então } (y_i = a_0 + a_1 x_1 + \ldots + a_n x_n)$$

onde x_j (para $j = 1 \ldots n$) é a j-ésima variável de entrada do sistema, V_j é o j-ésimo valor *fuzzy*, y_i é a saída da i-ésima regra de inferência, calculada a partir da combinação linear das entradas e dos coeficientes $a_0 \ldots a_n$.

Para um sistema de ordem zero, o nível de saída y é uma constante, dados $a_1 = \ldots = a_n = 0$. Desta forma, uma função constante nada mais é que um peso, a_0, atribuído a cada regra, de acordo com o grau de importância da mesma para a saída do sistema.

Diferentemente do modelo de Mamdani, um sistema Takagi-Sugeno não necessita de uma interface de defuzzificação, dado que a sua saída é calculada diretamente. O nível de saída y do sistema de inferência é calculado através da saída y_i de cada regra i, bem como pelo peso τ_i atribuído a essa regra (de Souza, 2006). A etapa de agregação é simplesmente dada pela média ponderada das regras. Assim, o nível de saída final do sistema é dado por:

$$y = \frac{\sum_{i=1}^{R} \tau_i y_i}{\sum_{i=1}^{n} \tau_i} \tag{2.1}$$

onde R é o número total de regras do sistema.

2.4.3 Comparação entre os modelos de Mamdani e de Takagi-Sugeno

Por ser mais compacto e computacionalmente mais eficiente que um sistema que segue o modelo de Mamdani, o sistema de inferência de Takagi-Sugeno volta-se ao uso de técnicas adaptativas para a construção de modelos *fuzzy*. Tais técnicas podem ser usadas para customizar as funções de pertinência de maneira que o sistema de inferência em questão modele de melhor maneira os dados.

Desta forma, podemos citar uma série de vantagens de cada um dos modelos detalhados:

Vantagens do Modelo de Mamdani

- É intuitivo;
- É amplamente aceito;
- É mais adequado a entradas humanas.

Vantagens do Modelo de Takagi-Sugeno

- É computacionalmente eficiente;
- Trabalha bem com técnicas lineares;

- Trabalha bem otimização e técnicas adaptativas;
- Tem continuidade garantida da superfície de saída;
- É mais adequado a análises matemáticas.

2.5 Modelos *fuzzy* adaptativos

Embora o conceito de lógica *fuzzy* aplicado no meio industrial, em geral, tenha sofrido bastante oposição no seu início (Sadeghi-Tehran et al., 2012), essa é uma teoria amplamente aceita nos dias de hoje. A importância do uso de tal técnica fica clara quando o modelo do sistema é desconhecido ou quando a implementação de métodos analíticos clássicos não é possível. Na verdade, quando todos os possíveis eventos e a frequência com que eles ocorrem não podem ser identificados no momento da modelagem do sistema. Tal imprecisão impõe um modelo aproximado para o sistema. É importante ressaltar que os algoritmos *fuzzy* não estão limitados a tal tipo de problema. Algumas vezes, o modelo matemático é conhecido, mas a sua alta complexidade pode consumir uma grande quantidade de tempo no projeto e na execução.

Devido ao fato de que um algoritmo *fuzzy* tem a característica de ser um aproximador universal, é possível modelar um sistema que contém não-linearidades desconhecidas através de um conjunto de regras Se-Então. Os parâmetros de um controlador são geralmente sintonizados previamente pelo operador do processo.

Como veremos no decorrer desta seção, o modelo de um sistema *fuzzy* do tipo Takagi-Sugeno possibilita a sintonização automática dos seus parâmetros. Com isso, não só é possível identificar-se o modelo de um processo, desenvolvendo-se automaticamente um controlador compatível, como também pode-se trabalhar nas alterações do comportamento do sistema.

O principal desafio neste caso é que, muitas vezes, um sistema *fuzzy* é projetado para funcionar em determinadas condições. Isso quer dizer que o seu comportamento pode não ser adequado em situações diferentes das previstas. Essas variações podem ocorrer devido a falhas na planta, perturbações, ou mesmo alterações lentas do ambiente.

Neste ponto, é importante diferenciar dois conceitos existentes e bastante difundidos na literatura: os modelos adaptativos (*adaptive*) e os modelos evolutivos (*evolving*).

O termo "*adaptive*", em geral, é utilizado para os sistemas convencionais, conhecidos na teoria de controle por trabalharem com a adaptação de parâmetros de sistemas lineares. Um ótimo exemplo de conceito são as redes neuro-*fuzzy*, que serão abordadas na próxima seção. Já o termo "*evolving*", característica buscada no desenvolvimento das propostas desta tese, associa-se aos sistemas que, além dos seus parâmetros, possuem um desenvolvimento gradual da sua estrutura (Angelov & Kasabov, 2006), como por exemplo a sua base de regras, para um sistema de inferência *fuzzy*, ou a sua estrutura, para uma rede neural.

As redes neuronais artificiais (ou simplesmente redes neurais) são utilizadas há décadas como ferramentas associadas aos processos de automação industrial. Por

se basearem na estrutura geral do cérebro humano, são capazes de aprender e de se auto-organizarem. Além disso são estruturas tolerantes a falhas e bastante flexíveis.

Desde a década de 1940, diversos modelos de utilização das redes neurais foram apresentadas na indústria. Um dos modelos mais conhecidos com capacidade de adaptação é o da rede neuro-*fuzzy*. O objetivo na implementação de um sistema neuro-*fuzzy* é a criação de um sistema de inferência *fuzzy*, com características semelhantes à implementação tradicional, através de uma rede neural.

Apesar do poder computacional de uma rede neural, um dos principais empecilhos para a sua vasta utilização é a dificuldade de explicar ao usuário, de uma forma compreensível na linguagem humana, como a rede neural chega a um determinado resultado a partir das entradas (Mitra & Hayashi, 2000). Com a associação das duas abordagens, unem-se as vantagens das redes neurais, tais como a capacidade de aprendizagem e a tolerância a falhas, com a fácil interpretação dos sistemas de inferência *fuzzy*.

Um dos sistemas neuro-*fuzzy* mais utilizados na atualidade é o *ANFIS* (do inglês, *Adaptive Neuro-Fuzzy Inteference System*) (Silva, 2010). Ele implementa um sistema de Takagi-Sugeno na forma de uma rede neural, a partir do vetor de entradas, do tipo e número de funções de pertinência das entradas, e do vetor de parâmetros da saída. Um algoritmo *backpropagation* é utilizado para aprender as funções de pertinência parte antecedente de cada uma das regras, enquanto um algoritmo de mínimos quadrados determina os coeficientes dos sub-modelos lineares de cada uma das regras do sistema Takag-Sugeno.

Em um sistema ANFIS, a base de regras deve ser conhecida antecipadamente. O sistema, então, só é capaz de ajustar as funções de pertinência e os coeficientes da parte consequente.

O modelo ANFIS descrito por Jang & Sun (1995) apresenta uma rede neural em cinco camadas, cada uma com uma finalidade específica. A Figura 2.3 ilustra uma rede *ANFIS* para três entradas (x_1, x_2 e x_3), e cada entrada dividida em dois conjuntos *fuzzy* (A e B). O modelo a seguir é baseado na descrição apresentada em Silva (2010).

- **Camada 1:** Calcula o grau de pertinência de cada uma das entradas reais em relação aos valores linguísticos (A e B) descritos para a variável. Nesse primeiro estágio, é possível ajustar tais valores.
- **Camada 2:** Calcula o nível de disparo para cada nó, que corresponde a uma regra de inferência. Nesse estágio, é executada a operação de t-norma, podendo ser usada a operação de produto. Para o exemplo da Figura 2.3, as saídas da camada são:

$$S_1 = \mu_{A_1}(x_1).\mu_{A_2}(x_2).\mu_{A_3}(x_3)$$
$$S_2 = \mu_{B_1}(x_1).\mu_{B_2}(x_2).\mu_{A_3}(x_3)$$
$$S_3 = \mu_{B_1}(x_1).\mu_{B_2}(x_2).\mu_{B_3}(x_3)$$

- **Camada 3:** Inicia o processo de *defuzzificação*, realizando a operação de normalização nos valores de saída da camada 2, calculando-se a razão entre o

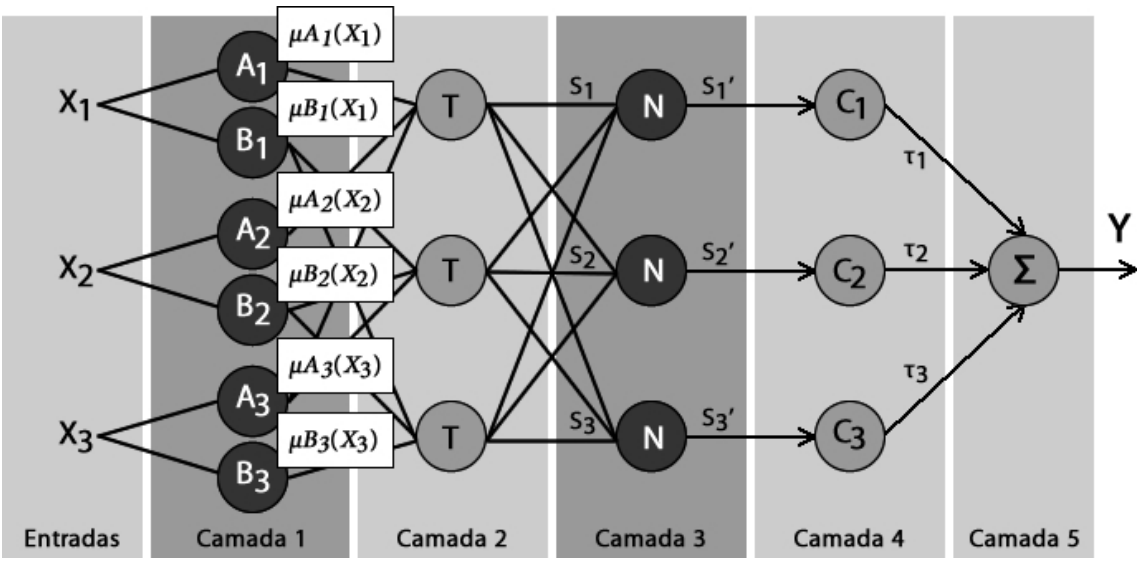

Figura 2.3: Representação de um sistema *ANFIS*

nível de disparo da regra referente àquele nó pela soma dos níveis de disparo de todas as regras. As saídas dos nós dessa camada são:

$$S'_1 = \frac{S1}{S1 + S2 + S3}$$

$$S'_2 = \frac{S2}{S1 + S2 + S3}$$

$$S'_3 = \frac{S3}{S1 + S2 + S3}$$

- **Camada 4:** Calcula o produto entre as saídas da camada anterior pelo resultado da combinação linear (y_i) das entradas do sistema de Takagi-Sugeno, referente à parte consequente de cada regra i.

$$\tau_1 = S'_1.y_1 = S'_1.(a_{10} + a_{11}x_1 + a_{12}x_2 + a_{13}x_3)$$

$$\tau_2 = S'_2.y_2 = S'_2.(a_{20} + a_{21}x_1 + a_{22}x_2 + a_{23}x_3)$$

$$\tau_3 = S'_3.y_3 = S'_3.(a_{30} + a_{31}x_1 + a_{32}x_2 + a_{33}x_3)$$

- **Camada 5:** Finaliza o processo de *defuzzificação*, calculando a saída real do sistema através da soma das saídas da camada anterior.

$$y = \tau_1 + \tau_2 + \tau_3 = S'_1.y_1 + S'2.y_2 + S'3.y_3$$

Sabendo que R é o número de regras do sistema ou, nesse caso, o número de

neurônios na camada 2, temos

$$y = \frac{\sum_{i=1}^{R} \tau_i y_i}{\sum_{i=1}^{R} \tau_i}$$

Além de Silva (2010), Mitra & Hayashi (2000) e Jang & Sun (1995), que apresentam revisões detalhadas dos algoritmos neuro-*fuzzy* para geração de regras, dentro de um ambiente de computação adaptativa que unifica as redes neurais e os modelos *fuzzy*, existem diversos outros trabalhos relevantes na literatura. Khalajzadeh et al. (2011) faz um estudo em detalhes da aplicabilidade de sistemas especialistas no projeto e controle de veículos autônomos, e destaca as técnicas neuro-*fuzzy* na identificação dos modelos. Um mecanismo de rede neural com estrutura evolutiva é apresentado em Lin et al. (2001). Tal abordagem é voltada para o controle de aplicações e usa diferentes mecanismos para atualização de regras/neurônios, baseado no erro dos estágios anteriores.

Apesar de permitir a adaptação e configuração automática de parâmetros, a natureza intrinsecamente estática das ANFIS limita a base de regras do sistema fuzzy, impossibilitando a sua evolução e crescimento da base de regras. Tal fator torna obrigatório o conhecimento prévio de detalhes do problema, como por exemplo, no caso de uma tarefa de DDF, o conjunto de falhas esperadas para o sistema. A solução proposta para o problema, diferentemente das técnicas puramente adaptativas, baseia-se em modelos evolutivos, permitindo a adaptação profunda do sistema, o que inclui atualização e criação de novas regras na sua base, a partir da detecção de novos pontos de operação do processo. A partir de conhecimento mínimo sobre a planta e configurações altamente intuitivas por parte do operador, o sistema proposto, que será apresentado nos capítulos seguintes, possui uma estrutura bastante flexível e transparente para o usuário.

Capítulo 3

Detecção e Diagnóstico de Falhas

Neste capítulo, trataremos das definições a respeito do processo de DDF. Serão apresentados os principais conceitos referentes ao tema, bem como uma revisão geral da literatura e o estado da arte dessa área de pesquisa.

3.1 Falha, defeito e mau funcionamento

Em primeiro lugar, é importante abordar algumas das nomenclaturas e definições da área de pesquisa. O termo *falha* é a mudança de um estado de operação com valores aceitáveis para um conjunto de variáveis observadas ou parâmetros calculados dentro de um processo (Venkatasubramanian et al., 2003c). Assim sendo, uma falha pode ser definida como um sintoma (por exemplo, fluxo baixo de um líquido ou alta temperatura em uma bomba) dentro do processo. Um *defeito*, também conhecido por *mau funcionamento* ou *causa raiz*, por sua vez, é o evento causador de tal anormalidade.

Em um contexto industrial, existem diversos tipos de falhas capazes de afetar a operação normal de uma planta. Dentre os principais grupos de defeitos, podemos citar (Samantaray & Bouamama, 2008):

- Alterações de parâmetros: também conhecidas como falhas paramétricas, esse grupo engloba as "perturbações no processo a partir de variáveis independentes, das quais as dinâmicas não são fornecidas ou conhecidas dentro do processo" (Samantaray & Bouamama, 2008). Como exemplos de falhas paramétricas, podemos citar uma alteração na concentração de um determinado reagente, um entupimento de um oleoduto, resultando em uma alteração do coeficiente do fluxo, e assim por diante.
- Alterações estruturais: estão relacionadas às falhas nos equipamentos propriamente ditas, as quais podem acarretar em alterações no modelo do processo. Uma ação corretiva de controle apropriada para tal anormalidade requer a extração de novas equações de modelo para descrever o estado de falha atual do processo. Exemplos de falhas estruturais incluem falhas no controlador, vazamento de um tubo ou uma válvula emperrada.

- Falhas nos sensores e atuadores: também conhecidas como falhas aditivas, estão relacionadas a entradas e saídas errôneas no processo, o que pode acarretar em valores além dos limites aceitáveis para as variáveis da planta. Alguns exemplos de tais anormalidades incluem *off-sets* constantes (positivos ou negativos), perturbações intermitentes, saturações, valores fora do alcance do instrumento e assim por diante.

A Figura 3.1 ilustra o processo geral e a estrutura de um sistema de diagnóstico de falhas.

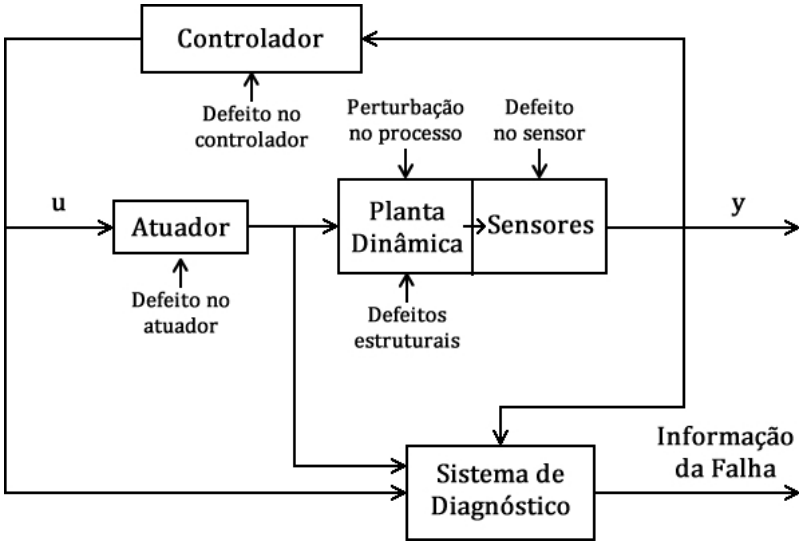

Figura 3.1: Estrutura geral de um sistema de DDF (Venkatasubramanian et al., 2003c)

Além da classificação a partir da localização, as falhas também podem ser classificadas pelo aspecto de variação no tempo, como:

- Abruptas: falhas que abruptamente/instantaneamente alteras o valor de uma variável (ou um grupo de variáveis), de um valor constante para outro. Estão frequentemente relacionadas a *hardwares* danificados.
- Incipientes: referem-se às mudanças paramétricas lentas, onde a falha gradualmente desenvolve-se para um grau mais alto no decorrer do tempo. Geralmente são mais difíceis de detectar devido às suas características, porém, são menos severas que as falhas abruptas (Edwards et al., 2010). Um bom exemplo de falha incipiente é a degradação lenta de um componente de *hardware*.
- Intermitentes: são falhas que aparecem e desaparecem, repetidamente, no decorrer do tempo. Exemplos típicos são uma fiação parcialmente danificada ou um conector solto.

A Figura 3.2 apresenta os tipos de falhas de acordo com o aspecto de variação no tempo.

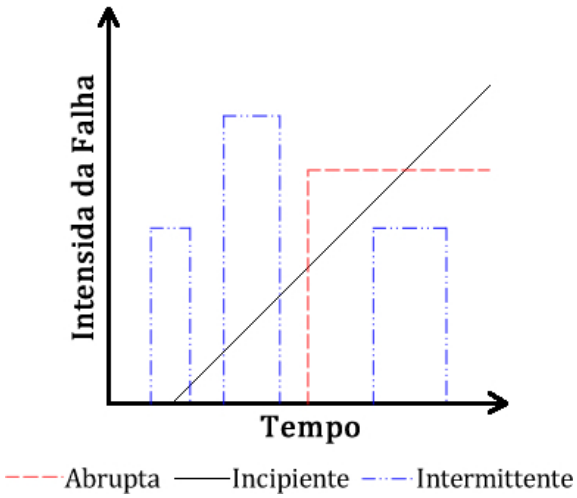

Figura 3.2: Falhas de acordo com suas variações no tempo (Patan, 2008)

É importante ressaltar que uma anormalidade genérica só é considerada uma falha caso seja possível recuperar-se do estado anormal através de uma ação de controle apropriada. No passado, abordagens passivas, fazendo uso de técnicas de controle robusto que garantia que o sistema de malha fechada se tornasse insensível a algumas situações defeituosas leves, eram popularmente utilizadas como estratégia de tolerância a falhas (Zhou & Ren, 2001). Nos dias atuais, a DDF ativa e os processos de recuperação são tidos como as melhores soluções, uma vez que eles possibilitam a acomodação da falha, permitindo uma atualização na ação de controle, de modo a ajustar o controlador, na presença de uma falha, a um novo cenário. Acomodação de falhas, também conhecido como compensação de falhas, é discutido em Lin & Liu (2007), Efimov et al. (2011) e vários outros.

3.2 Detecção, isolamento e identificação

O processo completo de GEA é, geralmente, dividido em uma série de estágios, os quais, dentro do projeto de sistemas tolerantes a falha, compõem o *esquema de diagnóstico de falhas*. Embora o número de estágios possa variar de autor para autor, a idéia geral permanece. O diagrama genérico do processo de GEA é ilustrado na Figura 3.3.

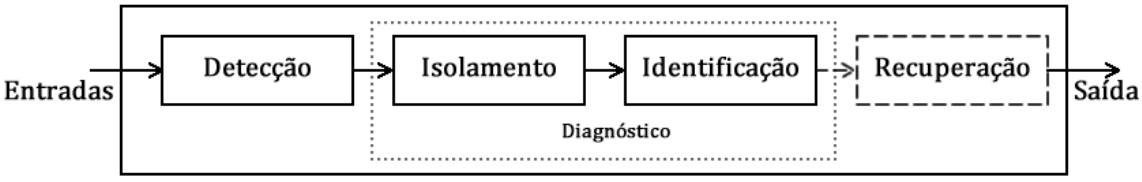

Figura 3.3: Diagrama do processo de GEA

Detecção de falhas (ou detecção de anomalias) é o primeiro estágio, e possui extrema importância nos sistemas de DDF. Nesse estágio, é possível identificar se o sistema está trabalhando em um estado normal de operação ou na presença de uma falha. Porém, nesse estágio, informações vitais a respeito da falha, tais como localização física, tamanho ou intensidade, não são fornecidas ao operador (Silva, 2008).

Nesse sentido, surge a necessidade de um estágio subsequente. O sistema detector (primeiro estágio) monitora continuamente as variáveis (ou atributos) do processo, procurando por sintomas (desvios consideráveis nos valores das variáveis), e envia tais sintomas para o sistema de diagnóstico, que é responsável pelo processo de classificação e identificação.

O sistema de diagnóstico possui os seus próprios desafios e obstáculos, e pode ser tratado independentemente do primeiro estágio. Ele demanda diferentes técnicas e soluções, e pode ser dividido em dois subestágios, chamados de *isolamento* e *identificação*. O termo isolamento refere-se à determinação do tipo, localização e instante da detecção da falha e, imediatamente, segue o sistema de detecção (Donders, 2002). Identificação, por outro lado, refere-se à determinação do tamanho e comportamento, no decorrer do tempo, de uma falha e, imediatamente, segue a etapa de isolamento.

O sistema de diagnóstico, particularmente, pode ser visto como um processo de tomada lógica de decisões, que gera informação qualitativa a partir de informação quantitativa, e pode ser tratado como um problema de classificação. A tarefa é relacionar cada padrão do vetor de sintomas com uma das classes de falha pré-determinadas, quando existentes, e o modo normal de operação (Frank & Köppen-Seliger, 1997).

Um último estágio relacionado às aplicações de DDF é a tarefa de recuperação a partir de uma falha existente e detectada. A ação a respeito do processo de reconfiguração precisa compensar o defeito atual, de maneira a manter os requisitos para um estado de operação aceitável, quando possível, ou determinar a sequência de eventos a seguir (desligamento controlado, por exemplo). Embora recuperação (ou acomodação) sejam temas diretamente relacionados ao esquema de DDF, este trabalho focará apenas nos estágios descritos anteriormente.

Em geral, técnicas baseadas em inteligência artificial, tais como as redes neurais, sistemas *fuzzy* e sistemas especialistas em geral, podem ser aplicadas em todos os estágios da DDF. Nas subseções a seguir, serão apresentadas algumas das técnicas amplamente conhecidas para DDF.

3.2.1 Técnicas baseadas em modelos quantitativos

De modo a detectar os estados anômalos de um processo, frequentemente, algum tipo de redundância é necessária. Ela é utilizada para comparar o estado atual do processo com o estado que é esperado para tais circunstâncias. Embora essa redundância possa ser fornecida por dispositivos extras de *hardware*, o que, de fato, acontece em processos de alta segurança, redundância analítica pode ser utilizada

para fornecer tal requerimento através do modelo matemático do processo (Frisk, 2001).

Quando o modelo do processo está disponível, a detecção de falhas utilizando técnicas baseadas em modelos quantititativos depende apenas da análise do sinal residual. O *resíduo* (e^r) é a diferença entre as saídas reais (y) do sistema e as suas saídas estimadas ($\hat{y}$), a partir do modelo. Em geral, espera-se que o resíduo seja nulo (ou o mais próximo disso), durante o modo de operação normal de um processo, e, consideravelmente diferente de zero, na presença de uma falha. Note que, no projeto de um sistema de DDF, devem ser consideradas as particularidades de um processo real (por exemplo, ruído do ambiente, incertezas do modelo), as quais podem levemente desviar o resíduo de zero e, ainda assim, não representar um evento de falha.

Modelos matemáticos podem estar disponíveis tanto para o estado normal de operação quanto para cada falha previamente conhecida, indicando que, sistemas de DDF baseados em modelo, não apenas são capazes de distinguir entre estados normais e de falha (detecção), mas também de identificar diferentes tipos e localizações das falhas (diagnóstico). A Figura 3.4 ilustra a estrutura geral de um sistema de DDF baseado em modelos quantitativos.

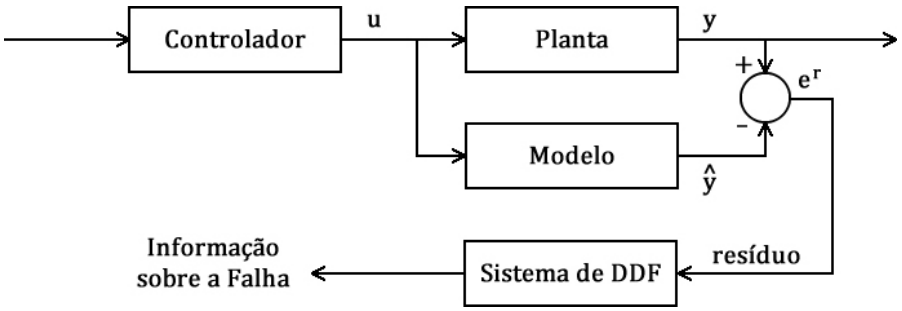

Figura 3.4: Estrutura geral de um sistema de DDF baseado em modelos quantitativos

Apesar de bastante eficientes em teoria, na prática, os métodos baseados em modelos quantitativos são dificilmente passíveis de implementação, uma vez que requerem descrições matemáticas detalhadas do sistema em pleno funcionamento e, preferencialmente, de cada uma das falhas previstas. É fácil notar que, dessa forma, a arquitetura do sistema virtualmente elimina a possibilidade de adaptação a novos cenários e surgimento de falhas desconhecidas, o que é extremamente indesejável na implementação de um sistema de DDF dinâmico.

Diversas abordagens para DDF utilizando métodos baseados em modelos quantitativos foram investigados na literatura. Podemos citar Venkatasubramanian et al. (2003c) e Isermann (2005) como duas das principais referências sobre o tópico. No primeiro, os autores apresentam uma revisão sistemática e comparativa das inúmeros métodos baseados em modelos quantitativos, a partir de diferentes perspectivas. No segundo, por sua vez, o autor inclui algumas aplicações detalhadas de tais métodos,

todas baseadas em diferentes problemas industriais. Ainda a respeito das abordagens baseadas na análise residual utilizando modelos analíticos, a leitura dos trabalhos de Chen & Patton (1999) e Simani et al. (2002) é altamente recomendada.

3.2.2 Técnicas baseadas em modelos qualitativos

A respeito dos modelos de processo, existem métodos que requerem modelos matemáticos detalhados, porém, existem também aqueles que apenas necessitam da descrição qualitativa do modelo. No segundo caso, tais métodos são baseados na *expertise* do operador, no conhecimento qualitativo e entendimento básico da física, dinâmica e comportamento do processo.

Modelos qualitativos são particularmente úteis no sentido de que, mesmo que os modelos matemáticos precisos do processo estejam disponíveis, é frequentemente impraticável (ou impossível) obter todas as informações a respeito de parâmetros físicos relevantes do sistema, sem mencionar que parâmetros externos, tais como perturbações imprevisíveis, incertezas de modelo e assim por diante, não são consideradas nos modelos quantitativos. Assim sendo, os métodos de DDF baseados em descritores qualitativos são particularmente robustos (Glass et al., 1995).

Ao invés de saídas reais (numéricas) e sinais de resíduo, modelos qualitativos trabalham com uma base de dados qualitativa, que alimenta um detector de discrepâncias. O sinal resultante, ao invés de uma simples subtração, é representado por uma discrepância qualitativa, baseado no comportamento esperado para o estado atual e a saída real do sistema. A Figura 3.5 ilustra a estrutura geral de um sistema de DDF baseado em modelos qualitativos.

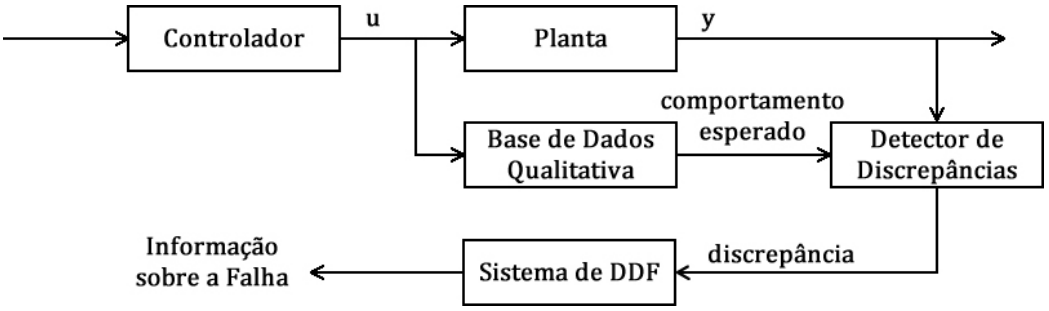

Figura 3.5: Estrutura geral de um sistema de DDF baseado em modelos qualitativos

Dentre os trabalhos relevantes sobre o tópico, é imprescindível, citar Venkatasubramanian et al. (2003a). Nesse trabalho, os autores apresentam uma revisão completa das técnicas baseadas em representação de modelos qualitativos e estratégias de busca em DDF, destacanto as vantagens e desvantagens relativas desses métodos. Outro trabalho que vale a pena mencionar é Katipamula & Brambley (2005), a primeira parte de uma revisão em duas partes, que sumariza algumas das técnicas de sucesso baseadas em modelos qualitativos e, embora aplicadas exclusivamente a problemas de aquecimento, ventilação e resfriamento, o trabalho foca em DDF genérica

e prognósticos, fornecendo uma estrutura de categorização, descrição e identificação de métodos, e apresentando suas vantages e desvantagens primárias.

3.2.3 Técnicas baseadas no histórico do processo

O terceiro e último grande grupo de métodos para DDF refere-se a um tipo particular de técnicas que são baseadas completamente nos dados disponíveis do processo. Técnicas baseadas no histórico do processo não requerem nenhum conhecimento prévio, seja ele quantitativo ou qualitativo, a respeito do processo/planta. Ao invés disso, elas utilizam massivas quantidades de informação histórica coletada e calculada a partir dos instrumentos de medição da planta. Os dados são, então, transformados e apresentados como informação *a priori* ao sistema de DDF, através de um processo conhecido como *extração de características*.

Extração de características (ou seleção de características) é responsável pela redução da dimensionalidade dos dados, cuidadosamente extraindo apenas as informações relevantes a patir do vetor de entrada, que, frequentemente, consiste das saídas medidas dos sensores - variáveis observáveis - (por exemplo: nível do tanque, pressão na bomba), ou parâmetros calculados - atributos do processo (por exemplo: erro, oscilação da pressão). Métodos estatísticos, sistemas especialistas e redes neurais são, comumente, utilizados nesse tipo de abordagem. A Figura 3.6 detalha a estrutura geral de um sistema de DDF baseado no histórico do processo.

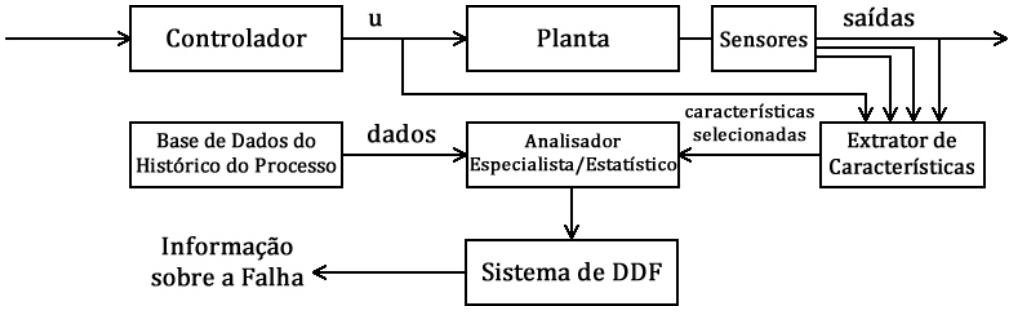

Figura 3.6: Estrutura geral de um sistema de DDF baseado no histórico do processo

Como referências na literatura, podemos citar Venkatasubramanian et al. (2003b) e Yang et al. (2003) como dois importantes trabalhos acerca do tópico. No primeiro, os autores apresentam a terceira parte de uma revisão completa da literatura, focando nos métodos de DDF baseados no histórico do processo. Na última parte de um exaustivo estudo, os autores sugerem que "nenhum método sozinho possui todas as características desejáveis para um sistema de diagnósticos" e, de modo a superar as limitações de estratégias individuais, o uso de sistemas de DDF híbridos é comumente recomendado. O último dos dois trabalhos apresenta uma revisão a respeito dos métodos de extração de características, analisando uma grande variedade de técnicas de extração de características de vibração já validadas, aplicadas a máquinas rotativas.

3.3 Sistemas *fuzzy* para detecção e diagnóstico de falhas

Sistemas baseados em regras *fuzzy* são investigados nas comunidades de detecção de falhas e confiabilidade, como poderosas ferramentas para modelagem e tomadas de decisão (Serdio et al., 2014), (Angelov et al., 2006), (Lemos et al., 2013), (Laukonen et al., 1995), juntamente com as redes neurais e outras técnicas mais tradicionais, tais como os observadores não-lineares e robustos, espaço de paridade, dentre outros (Mendonça et al., 2005). A teoria de conjuntos *fuzzy* possibilita a quantificação de afirmações intrinsecamente qualitativas, subjetividade e incerteza.

Os conceitos principais da lógica *fuzzy* tornam-a adequada para a detecção e diagnóstico de falhas. Enquanto a modelagem *fuzzy* não-linear pode ser bastante útil na tarefa de detecção, o sistema de inferência transparente e logicamente semelhante ao humano é altamente apropriado para o estágio de diagnóstico, que pode incluir a *expertise* do operador humano, mas também aprender a partir dos dados experimentais e/ou simulados. Outro benefício do uso dos sistemas *fuzzy* em aplicações de DDF é o bom desempenho na reprodução de mapeamentos não-lineares, bem como as suas habilidades de generalização, uma vez que os sistemas *fuzzy* são aproximadores universais, ou seja, capazes de modelar qualquer grau de não-linearidade com um grau de precisão arbitrário desejado (Castro & Delgado, 1996). Assim sendo, sistemas de detecção de falhas baseados em lógica *fuzzy* são vantajosos no sentido de permitirem a incorporação de conhecimento prévio e os seus motores de inferência serem de fácil entendimento para o operador humano (Mendonça et al., 2006).

O processo de DDF, especialmente nos últimos estágios, deve ser visto como um problema de classificação, trazendo certas particularidades quando comparados com outros grupos de aplicação. Quando estamos lidando com um problema de classificação, é útil pensar na saída do sistema como um valor linguístico ao invés de um valor real, ignorando-se, assim, o passo de *defuzzificação*. Desse modo, a saída de um sistema baseado em regras *fuzzy* para classificação deve se apresentar como um rótulo que representa a classe de falha atribuída ao estado atual do processo/planta.

Considerando um vetor de entradas reais $x \in \Re^n$, composto pelos valores das variáveis/atributos/características selecionadas, uma base de regras de inferência *fuzzy* $\mathcal{R}$, com R regras, para um sistema de DDF genérico, pode ser representada por:

$$\begin{aligned}
\mathcal{R}^1 &: \textbf{Se } (x_1 \textbf{ É } X_{1,1} \textbf{ E } \ldots \textbf{ E } x_n \textbf{ É } X_{1,n}) \textbf{ Então } (y_1 = \text{Classe}^1) \\
\mathcal{R}^2 &: \textbf{Se } (x_1 \textbf{ É } X_{2,1} \textbf{ E } \ldots \textbf{ E } x_n \textbf{ É } X_{2,n}) \textbf{ Então } (y_2 = \text{Classe}^2) \\
&\vdots \\
\mathcal{R}^R &: \textbf{Se } (x_1 \textbf{ É } X_{R,1} \textbf{ E } \ldots \textbf{ E } x_n \textbf{ É } X_{R,n}) \textbf{ Então } (y_R = \text{Classe}^R)
\end{aligned}$$

onde X é o conjunto de valores *fuzzy* para as variáveis de entrada e y é a saída do sistema.

Note que a saída y é inferida como o rótulo que representa cada classe de falha, podendo incluir a natureza (por exemplo: estrutural, perturbações), a localização (por exemplo: tanque 1, bomba, válvula A), o tipo (por exemplo: vazamento, *off-set*), e o grau (por exemplo: leve, severa) da falha, assim como pode representar o estado de operação normal da planta.

A inferência de um sistema baseado em regras *fuzzy* para classificação deve ignorar o processo de *defuzzificação*, gerando um rótulo na saída. Na prática, isso pode ser realizado através da utilização da regra do "vencedor leva tudo" (Angelov & Zhou, 2008):

$$Classe = Classe^{(i^*)}, \quad i^* = argmax_{i=1}^{R} \left(y^i\right) \tag{3.1}$$

onde y^i representa o grau de pertinência do vetor de entrada $\overrightarrow{x}$ ao conjunto *fuzzy* X^i, considerando R regras de inferência.

A solução proposta neste trabalho incorpora características dos métodos baseados em modelos qualitativos e, principalmente, baseados no histórico do processo. O procedimento de DDF, dividido em dois estágios - detecção e classificação - e executado de forma *on-line*, não-supervisionada, através sistemas baseados na distribuição espacial dos dados oriúndos da planta, previamente selecionados pelo operador, é detalhado nos capítulos a seguir.

Capítulo 4

Trabalhos Relacionados

Os sistemas de inferência *fuzzy* podem ser utilizados em todas as fases de uma tarefa de DDF e podem ser aplicados às diferentes estratégias, detalhadas na subseção 3.2. No caso das abordagens de DDF baseadas em modelos quantitativos, os sistemas *fuzzy*, em geral, são utilizados na análise do resíduo gerado pela diferença entre as saídas real e estimada a partir do modelo matemático previamente conhecido. Essa análise é realizada a partir de uma base de regras que consideram sinal (positivo ou negativo), valor e variação do resíduo no tempo, e já foi exaustivamente debatida na literatura por Zhao et al. (2009), Kulkarni et al. (2009), Mendonça et al. (2009) e muitos outros.

Como discutido no capítulo anteior, embora as estratégicas de DDF sejam, em teoria, bastante eficientes, na prática, a dependência do modelo matemático não é apropriada para aplicações reais. Até nos casos em que o modelo está disponível, muitas vezes, ele não considera variáveis que, geralmente, estão presentes em ambientes industriais (e.g. inércia do processo, perturbações do ambiente). Partindo de um ponto de vista similar, pode-se considerar o uso de estratégias de DDF baseadas em modelos qualitativos, nas quais os sistemas *fuzzy* mostram-se particularmente efetivos no processo de detecção e classificação das falhas. A base de regras desse sistema considerará o estado e variação das diversas variáveis do sistema, bem como atributos calculados a partir de operações/combinações de tais variáveis, tais como erro, amplitude e frequência do sinal. O processo de inferência é baseado na *expertise* do operador, que, na prática, disponibiliza um modelo qualitativo do processo, a partir do seu conhecimento a respeito do comportamento, dinâmica e física da planta. Tal abordagem é bastante recorrente na literatura, como em Patton et al. (2000), Insfran et al. (1999) e muitos outros.

Por último, abordagens de DDF baseadas no histórico do processo são frequentemente utilizadas em aplicações onde nem o modelo quantitativo (matemático) nem o qualitativo (físico/dinâmico/de comportamento) estão disponíveis, ou quando o operador decide confiar inteiramento nos dados adquiridos do processo. Métodos de extração de características, utilizados para transformar grandes quantidades de dados em "conhecimento prévio", são, muitas vezes, baseados nos aspectos da teoria *fuzzy*. Dada a ausência da *expertise* do operador, as características importantes de cada uma das falhas devem ser extraídas a partir de um grande conjunto de características (em geral, todas as informações disponíveis sobre a planta) e a

saída inferida a partir desse conjunto reduzido e computacionalmente viável de características. Um sistema de DDF pertencente ao referido grupo, frequentemente, utiliza variáveis estatísticas que determinam o comportamento da planta de acordo com o seu histórico, tais como variância, desvio padrão, limites superiores e inferiores. Abordagens desse tipo são apresentadas na literatura em Bae et al. (2005), Murphey et al. (2003) e muitos outros.

Muitos dos métodos para DDF abordados na literatura são orientados ao problema e, embora classificáveis em um dos três principais grupos de métodos recentemente citados, ainda podem ser bem diferentes em diversos aspectos.

Detecção de falhas em robôs baseada em observadores de estados é discutida em Sneider & Frank (1996). O método de supervisão proposto utiliza informações não-mensuráveis do processo. Esse método é revisado e aplicado no problema de detecção de falhas em um robô industrial, utilizando modelos dinâmicos, aprimorado pela inclusão de termos de fricção não-lineares. Uma abordagem para avaliação residual, baseada em lógica *fuzzy*, de técnicas de detecção de falhas baseadas no modelo matemático é investigada em processos com perturbações não-estruturadas.

Em Simani & Patton (2008), os autores propõem uma abordagem baseada em redundância analítica, focando-se na identificação *fuzzy* orientada ao projeto de um conjunto de estimadores *fuzzy* para DDF. Diferentes aspectos do problema de detecção de falhas são tratados no artigo, tais como estrutura do modelo, estimativa de parâmetros e geração de resíduos. A abordagem proposta é aplicada a um motor a diesel industrial real.

Uma abordagem baseada em modelo para DDF utilizando correspondência *fuzzy* é proposta em Dexter & Benouarets (1997). O esquema utiliza um conjunto de modelos de referência *fuzzy*, obtidos de simulações *on-line*, que descrevem os modos de operação normal e de falhas. Um classificador baseado em correspondentes *fuzzy* avalia o grau de similaridade a cada vez que o modelo *fuzzy on-line* é identificado. O método também trabalha com quaisquer ambiguidades, que podem resultar tanto dos estados de operação normal e de falhas, ou diferentes tipos de falha com sintomas similares em um dado estado de operação.

Um método para projeto de observadores *fuzzy* desconhecidos para modelos de Takagi-Sugeno (T-S) é abordado em Akhenak et al. (2009). O artigo apresenta o desenvolvimento de um observador robusto na presença de perturbações, utilizado para detecção e isolamento de falhas que afetam o modelo T-S. A metodologia proposta é aplicada a um ambiente simulado que estima a taxa de guinada e falhas em um veículo de direção automática.

Em Gmytrasiewicz et al. (1990), Tanaka et al. (1983) e Peng et al. (2008), diferentes abordagens para DDF para sistemas baseados em análise de árvores de falha *fuzzy* são discutidos. Esse métodos têm o intuito de diagnosticar os defeitos em componentes a partir da observação dos sintomas *fuzzy*, utilizando a informação contida na árvore de falhas. Enquanto em uma análise convencional (não-*fuzzy*) as probabilidades de defeitos em componentes de um sistema são tratadas como valores exatos na estimativa da probabilidade de falha do evento-topo, uma árvore de falhas *fuzzy* emprega a "possibilidade", ao invés da probabilidade, de falha, através de um

conjunto *fuzzy* definido no espaço de probabilidades.

Um método para diagnóstico de falhas baseado em padrões de tendência mostrados nas medições dos sensores é apresentado em Dash et al. (2003). O processo de análise da tendência envolve representação gráfica do sinal como padrões temporais, extração das tendências e comparações, através de estimativa *fuzzy* de similaridade, para inferir o estado do processo. A técnica é ilustrada com a sua aplicação à detecção de falhas de um reator exotérmico.

Nos trabalhos de Lughofer & Guardiola (2008) e Oblak et al. (2007), os autores apresentam diferentes abordagens para o uso de intervalos de confiança *fuzzy* para as saídas, de modo a normalizar os resíduos com incertezas no modelo. Enquanto no primeiro trabalho os autores avaliam os resultados baseados na medição de dados a partir de *benchmarks* de testes em motores, no último trabalho, o método proposto é usado na modelagem de uma planta não-linear para tratamento de água.

Em Oblak et al. (2007), os autores introduzem uma aplicação de modelos *fuzzy* intervalares para detecção de falhas em sistemas não-lineares com parâmetros incertos. Uma aplicação da abordagem proposta em um sistema de detecção de falhas para uma planta hidráulica de dois tanques é apresentada para demonstração dos benefícios do método proposto.

Em Hu et al. (2005), um método de dois estágios para diagnóstico de falhas baseado em decomposição empírica de modos (DEM), extração de características *fuzzy* e máquinas de vetor de suporte (sigla em inglês, SVM) é descrito. No primeiro estágio, componentes intrínsecos ao modo são obtidos com DEM a partir dos sinais originais, convertidos em vetores de características *fuzzy* e, então, a falha mecânica pode ser detectada. No estágio seguinte, esses vetores de características *fuzzy* servem de entrada para a multi-classificação SVM, a fim de identificar os diferentes casos de anormalidade. O método proposto é aplicado à classificação de um conjunto de turbogeradores, sob três diferentes condições de operação.

Um algoritmo *fuzzy*-genético para DDF automática em sistemas de aquecimento, ventilação e condicionamento de ar (AVCA) é apresentado em Lo et al. (2007). O sistema de DDF proposto monitora os estados do sistema AVCA continuamente através de um sistema *fuzzy*, com regras de inferência otimamente geradas por um algoritmo genético. Falhas são representadas em diferentes níveis e classificadas de maneira *on-line*, concomitantemente com o ajuste da base de regras.

Por último, mas não menos importante, o leitor é direcionado aos trabalhos de Rahman et al. (2010) e Calado & da Costa (2006), que apresentam revisões da literatura a respeito das aplicações de técnicas neuro-*fuzzy* em sistemas de DDF. Esses trabalhos mostram diversas aplicações de detecção, isolamento e classificação, utilizando abordagens neuro-*fuzzy*, individuais ou combinadas, destacando as vantagens e desvantagens de cada abordagem apresentada.

Das alternativas apresentadas, todas sofrem de problemas relacionados à necessidade de disponibilidade de modelos quantitativos e/ou qualitativos, fase de treinamento *off-line*, impossibilidade de adaptação a novas condições de operação, *expertise* do operador, necessidade de supervisão, esforço computacional extensivo ou grande quantidade de parâmetros e *thresholds* que são específicos do usuário e

do problema.

Dentre os trabalhos diretamente relacionados à proposta a ser descrita nesta tese, é importante mencionar algumas das propostas recentemente apresentadas na área de detecção e diagnóstico de falhas, utilizando abordagens *on-line*, modelos adaptativos e evolutivos baseados em regras *fuzzy*. Quanto ao primeiro estágio (detecção), uma abordagem bastante conhecida e difundida e que, posteriormente, será comparada à abordagem proposta neste trabalho, é o controle estatístico de processos (CEP). A estratégia trabalha com dados que são capturas de janelas móveis no histórico de um sistema de controle (Hossain et al., 1996). É utilizada no monitoramento de variáveis de processo, sendo baseada na análise estatística (valores de média e desvio padrão), calculadas em janelas de tempo e comparadas com limiares pré-definidos. Embora CEP seja uma abordagem *on-line*, muitas das aplicações hoje desenvolvidas baseiam-se na premissa de que os parâmetros do processo a ser controlado seguem distribuições Gaussianas/normais. Independência das entradas e número infinito de observadores são outras premissas que, na prática, não são satisfeitas. Para maiores informações sobre os métodos de CEP, a leitura dos trabalhos de Martin et al. (1996), Cook et al. (1997), Liukkonen & Tuominen (2004) and Kano et al. (2010) é indicada.

A respeito do estágio posterior, o artigo de Serdio et al. (2014) apresenta uma abordagem para DDF baseada em modelos *fuzzy* evolutivos/orientados a dados e análise dinâmica de resíduos para extração de indicadores de falhas. Os autores introduzem um algoritmo em duas etapas, uma *off-line* (identificação e treinamento do modelo) e uma *on-line* (detecção de falhas), onde nem o armazenamento de amostras nem modelos/padrões de falhas precisam estar disponíveis *a priori*. O sistema de DDF é aplicado com sucesso a moinhos de carvão de usina.

Lemos et al. (2013) e Lughofer & Guardiola (2008) apresentam dois sistemas diferentes para DDF *on-line* não-supervisionados, utilizando classificadores *fuzzy* evolutivos, baseados no algoritmo *evolving Takagi-Sugeno* (eTS), introduzido por Angelov & Filev (2004) e Angelov & Zhou (2008). Também é importante mencionar o trabalho de Lughofer (2010), uma vez que o autor desenvolve um classificador de imagens, capaz de separar imagens em "boas" (ítens livres de falha na produção) e "ruins" (ítens com falha na produção). A respeito da extração de regras de decisão a partir de *streams* de dados e tratamento de dados variáveis no tempo, algumas abordagens podem ser mencionadas, como por exemplo Gama & Kosina (2011) e Kosina & Gama (2012). No primeiro artigo, os autores apresentam um novo algoritmo de aprendizagem de conjuntos de regras, projetados para *streams* abertos e, no segundo, um algoritmo *on-line*, insensível ao tempo e de único passo para aprendizagem de regras de decisão no contexto de dados variáveis no tempo é introduzido.

Por último, mas não menos importante, Suvorov et al. (2013) introduzem um sistema baseado em SVM de uma classe, sendo a abordagem aplicada a dados reais de vôo oriúndos da indústria mundial de aviação. O algoritmo de dois estágios proposto neste trabalho e detalhado nos próximos capítulos, difere das abordagens mencionadas no sentido de que ou não necessita de fase separada de treinamento, ou não é baseado na estrutura tradicional do eTS. Ao invés disso, é baseado nos modelos *fuzzy*

AnYa (Angelov & Yager, 2012), (Angelov & Yager, 2011) . As regras de inferência não possuem parâmetros ou formas específicas para as funções de pertinência e é inteiramento orientado a dados. Além disso, o algoritmo é não-supervisionado, o que indica a não necessidade de uma base de regras pré-determinada, e novas falhas e rótulos de falhas são criados automaticamente na presença de cenários consideravelmente discrepantes dos existentes, sem a necessidade de intervenção do operador. Comparando-a especificamente com a última abordagem mencionada, o maior problema é que métodos utilizando SVM, em sua maioria, trabalham de maneira *off-line*, e mesmo as versões *on-line* de SVM de uma classe requerem massivo esforço computacional e definição de parâmetros que também são específicos do usuário e do problema.

Capítulo 5

Proposta de Trabalho

Como introduzido anteriormente, esta tese propõe um procedimento para DDF dividido em dois estágios subsequentes e independentes. O primeiro, que trata da *detecção* de falhas, é baseado na estimativa de densidade, calculada em tempo real, de maneira recursiva, a cada nova amostra de dados lida. O segundo estágio, responsável pela *classificação/identificação* das falhas, é executado através de um novo algoritmo de classificação, que apresenta-se como uma evolução de um grupo de classificadores auto-evolutivos existentes, sendo utilizado de maneira *on-line*, a cada falha detectada pelo primeiro estágio.

Os dois estágios, que podem ser executados separadamente e associados a outras técnicas existentes na literatura, são descritos em detalhes nas subseções a seguir.

5.1 Estimativa de densidade recursiva

A densidade de dados desempenha um papel muito importante na tarefa de detecção de falhas, bem como em diversos problemas correlatos na indústria. A sua utilização, porém, é bastante limitada pela complexidade e necessidade de cálculos intrinsecamente *off-line*, que acabam por impor limitações computacionais, tanto em termos de quantidade de memória quanto de processamento (Angelov, 2012a).

O conceito de estimativa de densidade recursiva (EDR) foi originalmente introduzido por Angelov et al. (2011), mas recebeu o nome de EDR (do inglês, *recursive density estimation - RDE*) em 2008 (Angelov et al., 2008), e a sua versão mais atual é parte de um pedido de patente (Angelov, 2012b). Desde então, EDR tem sido utilizada nas mais variadas aplicações (Angelov et al., 2008), (Kolev et al., 2013), (Ramezani et al., 2008).

Esse conceito utiliza uma função de Cauchy, que possui propriedades similares à Gaussiana, porém, pode ser atualizada recursivamente (Angelov, 2004), além de ser não-paramétrica. Além disso, não existe a necessidade de pré-suposições a respeito da distribuição dos dados. Isso significa que apenas uma quantidade pequena de informação - apenas a média das amostras de dados, μ_k, e o produto escalar, Σ_k, calculado no instante de tempo k - necessitam ser guardados na memória e atualizada. A amostra, x_k, também é utilizada, mas está disponível na interface de entrada, e não necessita ser guardada ou atualizada.

Tais premissas trazem implicações significantes, uma vez que permitem, em teoria, uma quantidade infinita de dados (conjuntos de tamanho infinito ou *streams* infinitos) ser processada em tempo real, recursivamente e de maneira exata (não aproximada).

Seja o vetor $x \in \Re^n$ composto por todas as variáveis do processo e dividido em diversos clusters. Então, para qualquer vetor $x \in \Re^n$, o valor da densidade para o seu Λ-ésimo cluster, calculado através da distância Euclideana, é (Angelov, 2012a):

$$d_\Lambda = \frac{1}{1 + \frac{1}{N_\Lambda} \sum_{i=1}^{N_\Lambda} ||x_k - x_i||^2} \tag{5.1}$$

onde d_Λ denota a densidade local do cluster Λ e N_Λ denota o número de amostras associados ao cluster Λ. No caso de aplicações de detecção de falhas, x_k representa o vetor de características com os seus respectivos valores para o instante k.

A distância é calculada entre o vetor dado (medido no instante k) e outros vetores que pertencem ao cluster ao qual o vetor x pertence (medidos em instantes anteriores). Demonstrou-se que essa fórmula pode ser derivada exatamente (não aproximadamente) em (Angelov, 2012a)

$$D(x_k) = \frac{1}{1 + ||x_k - \mu_k||^2 + \Sigma_k - ||\mu_k||^2} \tag{5.2}$$

sendo $D(x_k)$ a densidade calculada para a amostra x_k, onde tanto a média, μ_k quanto o produto escalar, Σ_k podem ser atualizados recursivamente por

$$\mu_k = \frac{k-1}{k}\mu_{k-1} + \frac{1}{k}x_k, \quad \mu_1 = x_1 \tag{5.3}$$

$$\Sigma_k = \frac{k-1}{k}\Sigma_{k-1} + \frac{1}{k}||x_k||^2, \quad \Sigma_1 = ||x_1||^2 \tag{5.4}$$

Os dados são coletados continuamente, de maneira *on-line*, durante a execução do processo. Alguns dos novos dados reforçam e confirmam a informação contida dos dados anteriormente adquiridos. Outros dados, entretanto, trazem nova informação, o que pode indicar uma variação nas condições de operação, desenvolvimento de uma falha ou, simplesmente, uma variação mais significativa na dinâmica do processo (Angelov, 2002), (Angelov & Buswell, 2002), (Angelov & Filev, 2004), (Angelov & Filev, 2002). O julgamento da importância dos dados é realizado baseando-se na sua proximidade espacial, o que pode corresponder às condições de operação, variações sasonais ou diferentes falhas.

Na detecção de *outliers* dentro de um *stream* de dados, a suposição é que, na análise de um determinado conjunto de características, o comportamento normal de tais características deve ser invariante. Entende-se por "invariante" ou "estável" o conjunto de características que, em regime permanente, não apresentam valores substancialmente oscilatórios, mas, obviamente, podem variar dentro dos limites de operação do regime para um processo industrial real. O vetor x_k é n-dimensional e

composto pelos valores das n características selecionadas, no instante de tempo k.

É importante mencionar que tal abordagem de detecção de falhas *on-line*, sendo baseada inteiramente no conceito de densidade dos dados no espaço n-dimensional, é altamente adequada e aplicável em condições onde não é possível executar um estágio de treinamento ou pré-determinar todas as falhas possíveis. Falhas não esperadas podem aparecer no decorrer do tempo, especialmente em ambientes dinâmicos, tais como a indústria. As redes neurais, por exemplo, devido à sua natureza intrínseca, são frequentemente restritas a configurações limitadas, ignorando a evolução implícita do ambiente. Além disso, os modelos tradicionais, tais como as redes neurais, tendem a desviar-se com o tempo, e um processo de recalibração torna-se necessário. EDR, bem como o método a ser proposto a seguir, não sofre de tal desvantagem, uma vez que é capaz de adaptar-se e evoluir. Ressalta-se que essa característica é crucial na área de estudos de sistemas evolutivos.

5.2 Detecção de falhas utilizando estimativa de densidade recursiva

O procedimento de detecção de falhas *on-line* proposto começa com a inicialização dos iteradores $k \leftarrow 1$ e $ks \leftarrow 1$. Enquanto k conta o número de amostras de dados lidos (ou seja, o número total de iterações do algoritmo), ks é responsável pela contagem do número de iterações em que o sistema permanece dentro do mesmo status ("normal"/"falha"). A variável *status* também é inicializada com o valor "normal", a partir da suposição de que, ao início da execução do sistema, o processo estará sob estado de operação normal.

A partir desse ponto, o vetor de amostras x_k é lido de uma das interfaces do sistema (arquivos de texto, banco de dados, protocolos industriais). Na primeira execução ($k = 1$), as variáveis *densidade* ($D(x_k) \leftarrow 1.0$), *media* da densidade ($\mu_D \leftarrow D_k$), μ_k and Σ_k são inicializadas e os contadores k and ks incrementados em 1 ($k \leftarrow k + 1$, $ks \leftarrow ks + 1$).

Da segunda iteração em diante ($k > 1$), as variáveis μ_k, Σ_k e $D(x_k)$ são atualizadas recursivamente pelas equações (5.3), (5.4) e (5.2), respectivamente. A variável Δ_D é, então, calculada pelo valor absoluto de $D(x_k) - D(x_{k-1})$, onde $D(x_k)$ é a densidade calculada para a amostra de dados atual (x_k) e $D(x_{k-1})$ é a densidade calculada para a amostra imediatamente anterior (x_{k-1}).

A média da densidade (μ_D) é calculada por

$$\mu_D = \left(\frac{ks-1}{ks}\mu_D + \frac{1}{ks}D(x_k)\right)(1 - \Delta_D) + D(x_k)\Delta_D \tag{5.5}$$

Essa informação será utilizada como uma medida para decisão de quando o sistema deve entrar ou deixar um estado de falha. Como é calculada recursivamente, não faz-se necessário o armazenamento de nenhum dos seus valores anteriores na memória, o que é apropriado para uma abordagem *on-line*. O cálculo de μ_D segue a premissa da equação (5.3), entretanto, é bem mais conservativo, no sentido de que

μ_D é baseado não somente nos valores passados de D, mas também é sensível às variações abruptas de densidade. O coeficiente $(1 - \Delta_D)$ irá guiar μ_D para próximo da média real de densidade quando o sinal varia suavemente, e Δ_D guiará μ_D para próximo do novo valor de D no caso de uma variação brusca.

Nesse momento, os seguintes cenários podem ocorrer:

a) **Se** o status atual do sistema é "normal" **e** $D(x_k) < \mu_D$, consecutivamente, pelas últimas ϵ_1 iterações **Então** mude o status para "falha" e reinicialize ks $(ks \leftarrow 0)$

b) **Senão Se** o status atual do sistema é "falha" **e** $D(x_k) >= \mu_D$, consecutivamente, pelas últimas ϵ_2 iterações **Então** mude o status para "normal" e reinicialize ks $(ks \leftarrow 0)$

c) **Senão** não faça nada.

Note que, nos casos (a) e (b), foram utilizados dois limiares: ϵ_1, de entrada e ϵ_2, de saída, que são bastante intuitivos para ao usuário. Após ϵ_1 iterações consecutivas com a densidade abaixo da média, o sistema irá **entrar** em um estado de falha e, após ϵ_2 iterações consecutivas com a densidade acima da média, o sistema irá **sair** de um estado de falha. Esses valores devem representar um bom custo-benefício entre tempo de resposta e robustez do sistema de detecção, e ambos devem ser baseados na ordem de magnitude do processo (segundos, para o caso de plantas de resposta rápida) e no período de amostragem do sistema. Em todos os exemplos apresentados nas próximas seções, foram utilizados os valores de $\epsilon_1 = 20$ e $\epsilon_2 = 80$, que representam intervalos de $2s$ e $8s$, respectivamente, para o período de amostragem de $100ms$ $(10Hz)$ utilizado nos experimentos. As variáveis ϵ_1 e ϵ_2 são os únicos parâmetros necessários para configuração do procedimento de detecção de falhas proposto.

O processo é terminado e começa novamente da leitura da próxima amostra de dados x_k, com $k \leftarrow k + 1$ e $ks \leftarrow ks + 1$. Em tratando-se de um processo *on-line*, o número total de leituras e iterações é, em teoria, indeterminado e, na prática, pode ser definido pelo usuário.

O procedimento recursivo proposto para detecção de falhas *on-line* utilizando EDR é detalhado no Algoritmo (1).

Algoritmo 1: Algoritmo proposto para detecção de falhas proposto

```
k ← 1;
ks ← 1;
status ← "normal";
while x_k ← leia próxima amostra de dados do
    if k = 1 then
        /* Inicialização */
        D(x_k) ← 1.0;
        μ_D ← D_k;
        μ_k ← x_k;
        Σ_k ← ||x_k||^2;
    else
        μ_k ← atualize pela equação (5.3);
        Σ_k ← atualize pela equação (5.4);
        D(x_k) ← atualize pela equação (5.2);
        Δ_D ← abs(D(x_k) − D(x_{k−1}));
        μ_D ← atualize pela equação (5.5);
        if status = "normal" then
            if D(x_k) < μ_D pelas últimas ε_1 iterações then
                status ← "falha";
                ks ← 0;
            end
        else
            if D(x_k) >= μ_D pelas últimas ε_2 iterações then
                status ← "normal";
                ks ← 0;
            end
        end
    end
    ks ← ks + 1;
    k ← k + 1;
end
```

5.3 Classificadores evolutivos

Os sistemas *fuzzy* têm sido amplamente utilizados em diferentes tarefas de classificação, tais como tomada de decisões, reconhecimento de padrões, processamento de imagens e, claro, detecção de falhas. Uma vez que a tarefa de classificação consiste em mapear um conjunto de características (e os seus valores em uma amostra de dados) em um conjunto de rótulos de classe, os sistemas *fuzzy* servem particularmente como comprovados aproximadores universais (Wang, 1992).

Os desafios enfrentados no processamento de informações, e na classificação em particular, são: 1) Necessidade de suportar grandes quantidades de dados e 2) processar *streams* de dados *on-line*, em tempo real (Angelov & Zhou, 2008). A respeito

do primeiro ponto, é importante salientar que o armazenamento de todas as amostras para posterior análise é praticamente impossível. A capacidade de processamento e armazenamento é um fator limitante em ambientes industriais reais. Sobre o último ponto, surge uma questão extremamente relevante: os dados coletados em ambiente industrial tendem a sofrer alterações, na maioria das vezes não previstas, no decorrer do tempo. Tais alterações podem estar relacionadas a variações sazonais, perturbações, alterações lentas do ambiente ou novos pontos de operação, que podem ou não indicar a presença de falhas.

A seguir, serão apresentadas duas técnicas para classificação autônoma e *on-line* de dados, que, posteriormente, serão utilizadas na identificação/classificação de falhas em plantas industriais.

5.3.1 Classificador eClass0

O algoritmo *eClass0* foi introduzido por Angelov & Zhou (2008), e é descrito aqui exatamente como apresentado no trabalho original. Uma regra de inferência da base de um sistema eClass0 segue o modelo típico de um classificador *fuzzy*:

$$\mathcal{R}^i : \textbf{Se } (x_1 \textbf{ É } X_1^{i*} \textbf{ E } \ldots \textbf{ E } x_n \textbf{ É } X_n^{i*}) \textbf{ Então } (\text{Classe}^i) \tag{5.6}$$

Onde $x = [x_1, x_2, ..., x_n]$ é o vetor de características, $\mathcal{R}^i$ denota a i-ésima regra de inferência *fuzzy*, para $i = [1, R]$, R é o número de regras *fuzzy*, X_j^{i*} denota o j-ésimo conjunto *fuzzy* da i-ésima regra, para $j = [1, n]$, X^{i*} é o ponto focal da parte antecedente da i-ésima regra, e $Classe^i$ é o rótulo da classe do i-ésimo ponto focal. Note que X^{i*} trata-se de um protótipo, uma amostra de dados real, não uma média.

A inferência de um classificador eClass0 utiliza a regra do "vencedor leva tudo":

$$\text{Classe} = \text{Classe}^{(i^*)}, \quad i^* = argmax_{i=1}^{R} \left(\tau^i\right) \tag{5.7}$$

onde τ^i representa o nível de disparo do vetor de entrada $\overrightarrow{x}$ para a i-ésima regra, que é determinado pelo produto dos valores de pertinência μ_j^i da j-ésima característica e do conjunto *fuzzy* X_j^{i*}, definidos pela função Gaussiana

$$\mu_j^i = \exp\left(-\frac{1}{2}\left(\frac{d_j^i}{r_j^i}\right)^2\right) \tag{5.8}$$

onde d_j^i é a distância entre a amostra de dados e o ponto focal da i-ésima regra e r_j^i é a abertura da função de pertinência, que também representa o raio da zona de influência da regra.

Note que essa representação se assemelha à distribuição normal, com a abertura da função de pertinência também podendo ser representada pelo desvio padrão. A abertura r_j^i é determinada com base na dispersão σ_j^i dos dados por cluster/regra ($i = [1, R]$) e por característica ($j = [1, n]$)

$$\sigma_{jk}^{i} = \sqrt{\frac{1}{S_k^i}\sum_{l=1}^{S_k^i} d^2(x_j^l, X_j^{i*})}, \quad \sigma_{j1}^{i} = 1 \tag{5.9}$$

onde S_k^i denota número de amostras associadas ao i-ésimo cluster/regra no k-ésimo instante de tempo. Tal valor é inicializado com 1 no momento da criação de um cluster e incrementado em uma unidade, a cada vez que uma amostra lida está próxima desse cluster.

Quando um novo cluster/regra é criado, a sua abertura inicial será a média das aberturas dos clusters existentes para aquela característica, como em

$$\sigma_{R+1}^{k} = \frac{1}{R}\sum_{i=1}^{R}\sigma_k^i, \quad S_k^R = 1, \quad k = 2, 3, 4, ... \tag{5.10}$$

O algoritmo eClass0 é inicializado a partir da primeira amostra x_k. Inicialmente, a base de regras está completamente vazia, o que significa que nenhuma regra de inferência, cluster, ou rótulo foram criados ainda. Uma regra *fuzzy* (cluster) é criada a partir de x_1. O potencial do protótipo é definido como $P(X^{1*}) \leftarrow 1$. Consequentemente, a primeira amostra é classificada como pertencente à primeira classe definida, e a primeira regra de inferência segue o formato

$$\mathcal{R}^1 : \textbf{Se } (x_1 \textbf{ É } X_1^{1*} \textbf{ E } \ldots \textbf{ E } x_n \textbf{ É } X_n^{1*}) \textbf{ Então } (\text{Classe}^1) \tag{5.11}$$

A decisão de quando uma amostra é utilizada para formar uma nova regra ou substituir uma regra existente é baseada na medida de densidade espacial, chamada potencial. O potencial calculado para uma amostra é uma função da distância acumulada entre tal amostra e todas as outras amostras por classe. Assim, representa a densidade dos dados que circundam uma certa amostra. Originalmente, tem a forma (Angelov et al., 2007):

$$P_k(x_k) = \frac{1}{1 + \left(\sum_{j=1}^{n}\sum_{i=1}^{S_k^{R-1}} ||x_i - x_k||_j^2\right) / \left(S_k^{R-1}\right)}, \quad k = 2, 3, 4... \tag{5.12}$$

onde $p_k(x_k)$ denota o potencial da k-ésima amostra, x_k.

Note que a equação (5.12) requer o somatório da informação histórica de todas as amostras, o que contradiz os requisitos de uma aplicação *on-line*. Angelov & Zhou (2008) apresentam uma versão recursiva exata dessa equação, onde o potencial de cada cluster afetado $\mathcal{C}^l$ por uma nova amostra lida, pode ser atualizado por

$$P_k^l(X^{l*}) = \frac{(S_k^l - 1)P_{k-1}^l(X^{l*})}{S_k^l - 2 + P_{k-1}^l(X^{l*})\sum_{j=1}^{n}||x_k - X^{l*}||^2} \tag{5.13}$$

Se o potencial da nova amostra, x_k, é maior que o potencial de todos os clusters

existentes, um novo cluster $\mathcal{C}^{R+1}$ deverá ser criado no instante k:

$$X^{(R+1)*} = x_k, \quad R \leftarrow R + 1 \tag{5.14}$$

A criação de uma nova regra *fuzzy* a partir de um protótipo recentemente adicionado leva a um incremento gradual do tamanho da base de regras, o que explica a nomenclatura "evolutivo" (Angelov & Zhou, 2008).

5.3.2 Classificador AutoClass

O algoritmo *AutoClass* foi desenvolvido como alternativa para o segundo estágio do processo de DDF. Ele é inspirado na arquitetura do algoritmo eClass0, e pertence à família dos algoritmos de clusterização - eClustering (Angelov, 2004), ELM (Baruah & Angelov, 2012), DEC (Baruah & Angelov, 2013) - e classificação - eClass (Angelov & Zhou, 2008), simpl_eClass (Angelov et al., 2011) - evolutivos, porém a sua estrutura é baseada nos sistemas *fuzzy* do tipo AnYa, introduzidos por Angelov & Yager (2012) e Angelov & Yager (2011).

Diferentemente dos sistemas tradicionais de Mamdani (Mamdani & Assilian, 1975) e Takagi-Sugeno (Takagi & Sugeno, 1985b), um sistema AnYa não requer a definição explícita dos conjuntos *fuzzy*, nem as suas funções de pertinência correspondentes para cada valor de entrada. Ao invés disso, AnYa aplica os conceitos de *nuvens de dados* (do inglês, *data clouds*) (Angelov & Yager, 2012) e densidade de dados relativa para definir os antecedentes que representam exatamente a densidade e distribuição real dos dados, que podem ser obtidos *on-line*, a partir de *streams* de dados.

Nuvens de dados são subconjuntos das amostras de dados anteriores com propriedades comuns (proximidade no espaço de dados) (Angelov & Yager, 2012). Ao contrário das funções de pertinência tradicionais, elas representam direta e exatamente todas as amostras de dados. Uma dada amostra pode pertencer a todas as nuvens em diferentes graus, $\gamma \in [0, 1]$, sendo o aspecto *fuzzy* preservado no modelo. É importante enfatizar que, diferentemente dos clusters tradicionais, como em eClass0, nuvens de dados não possuem formas específicas e, dessa forma, não requerem a definição de limites, parâmetros ou funções de pertinência para o antecedente das regras *fuzzy*.

AutoClass, assim como o eClass0, é um mapeamento do espaço de características no espaço de rótulos. Uma das características importantes do AutoClass, é que os rótulos de classe são gerados automaticamente, em sequência ("Classe1", "Classe2", ... , "Classe R"). Um classificador *fuzzy* genérico descreve, na sua parte antecedente, um particionamento *fuzzy* do espaço de características $x \in \Re^n$, e na sua parte consequente, o rótulo da classe. A estrutura do AutoClass, por sua vez, segue o modelo dos sistemas AnYa:

$$\mathcal{R}^i : \textbf{Se } \left(\vec{x} \sim \aleph^i\right) \textbf{ Então } \left(\text{Classe}^i\right) \tag{5.15}$$

onde $\sim$ denota a pertinência *fuzzy* expressada linguisticamente como "está associado

com", $\aleph^i \in \Re^n$ é a i-ésima nuvem de dados, definida no espaço de entrada, $\vec{x} = [x_1, x_2, \ldots, x_n]^T$ é o vetor de características e $Classe^i$ é o rótulo de classe da i-ésima nuvem de dados.

A inferência em AutoClass é produzida por:

$$Classe = Classe^{i*}, \quad i^* = argmax_{i=1}^{n}(\gamma^i) \tag{5.16}$$

onde γ^i denota o grau de pertinência do vetor de entrada x_k à nuvem de dados $\aleph^i$, definida aqui como a densidade relativa normalizada, por (Angelov & Filev, 2004)

$$\lambda_k^i = \frac{\gamma_k^i}{\sum_{j=1}^{N} \gamma_k^i}, \quad i = 1, ..., N \tag{5.17}$$

onde γ_k^i é a densidade local da i-ésima nuvem de dados estimada a partir dessa amostra.

A densidade local é definida por uma função de Cauchy sobre a distância entre x_k e todas as outras amostras naquela nuvem de dados, que pode ser calculada recursivamente (Angelov, 2012a) por

$$\gamma_k^i = \frac{1}{1 + ||x_k - \mu_k||^2 + \sum_k - ||\mu_k||^2} \tag{5.18}$$

onde γ_k^i denota a densidade relativa da i-ésima nuvem de dados calculada no k-ésimo instante de tempo; μ_k denota a média e $\sum_k$ o produto escalar para a amostra x_k, calculados pelas equações (5.3) and (5.4), respectivamente.

O algoritmo AutoClass começa com a definição da "zona de influência" inicial r_0 pelo usuário. Embora o conceito de nuvens de dados distingue-se dos clusters tradicionais no sentido de que não existem limites definidos, ainda consideramos aqui a zona de influência de uma nuvem de dados e, a partir desse limite, definimos se uma nuvem exerce ou não influência sobre uma determinada amostra. Esse é o único parâmetro definido pelo usuário e, ainda assim, é bastante intuitivo. Valores muito altos da zona de influência inicial r_0 levarão à criação de poucas nuvens, e vice-versa. Valores iniciais de $r_0 \in [0.3, 0.5]$ são recomendados (Angelov & Filev, 2004), assumindo uma extensão de $[0, 1]$ (normalizado). Então, a primeira amostra de dados é lida no instante de tempo $k = 1$.

Inicialmente, a base de regras está completamente vazia, o que significa que nenhuma regra de inferência, nuvem de dados, ou rótulo foram criados ainda. Depois da leitura da primeira amostra, uma nuvem $\aleph^1$ é criada, e o número de nuvens/regras existentes passa a ser 1 ($R \leftarrow 1$).

O ponto focal X^{1*} (que neste caso é definido pela média das amostras) de $\aleph^1$ será a própria amostra x_1:

$$X^{1*} = x_1 \tag{5.19}$$

A zona de influência inicial, r_0, previamente definida pelo usuário, é atribuída a

$\aleph^1$:

$$r^1 = r_0 \tag{5.20}$$

Uma vez que x_1 é a primeira amostra, o número de amostras associados à nuvem $\aleph^1$, até o momento, é 1 ($S_1^1 \leftarrow 1$). O rótulo $Classe^1$ como consequente completa a primeira regra de inferência:

$$\mathcal{R}^1 : \textbf{Se } \left(\vec{x} \sim \aleph^1\right) \textbf{ Então } \left(\text{Classe}^1\right) \tag{5.21}$$

Note que não é necessário armazenar todas as amostras. As informações que representam uma nuvem existente são o seu ponto focal (média), a sua zona de influência, a sua densidade e o número de pontos associados a essa nuvem. Tal fator é de extrema importância nas aplicações *on-line*.

Da segunda iteração em diante ($k > 1$), AutoClass irá trabalhar com essa base de regras, atualizando as regras existentes e criando novas, quando necessário.

Com cada amostra subsequente x_k que é lida, para $k > 1$, dois cenários podem ocorrer: a) a amostra x_k está associada a uma ou mais nuvens existentes, ou b) a amostra x_k não está dentro da zona de influência de nenhuma nuvem de dados, o que significa que x_k é i) um *outlier*, ou ii) pode fazer parte de um novo ponto de operação e deve compor uma nova nuvem.

No caso (a), considerando *próximo* um ponto que está dentro de duas vezes a zona de influência de uma ou mais nuvens, todas as nuvens que exercem alguma influência sobre x_k serão atualizadas (aqui, novamente, é utilizada a distância Euclideana, porém, outras abordagens também são aceitáveis). Esse é um passo importantíssimo de modo a preservar o aspecto *fuzzy* do sistema. Para cada nuvem $\aleph^l$ afetada, os seguintes passos serão executados por AutoClass:

- O ponto focal (média) X^{l*} é atualizado. O deslocamento do ponto focal é proporcional ao valor de x_k, e o número de pontos S_k^l sob influência de $\aleph^l$, o que significa que, quanto mais populosa for $\aleph^l$, menos o ponto focal X^{l*} será movido em direção a x_k.

$$X^{l*} = \frac{(X^{l*} S_k^l + x_k)}{S_k^l + 1} \tag{5.22}$$

- A zona de influência de $\aleph^l$ é atualizada. Seguindo a mesma ideia do ponto focal, a abertura da zona de influência é proporcional à distância entre x_k e X^{l*} no espaço n-dimensional de características e o número de pontos S_k^l sob influência de $\aleph^l$, o que significa que, quanto mais populosa for $\aleph^l$, menos a zona de influência r^l será aberta em direção a x_k.

$$r^l = \frac{(r^l S_k^l + ||x_k - X^{l*}||^2}{S_k^l + 1} \tag{5.23}$$

  Note que, após uma longa execução, o tamanho das projeções da zona de influência de cada nuvem será consideravelmente diferente. Nuvens mais densas

tendem a decrementar a sua zona de influência, enquanto nuvens mais esparsas tenderão a aumentá-la.

- O número de pontos sob influência de $\aleph^l$ é, então, incrementado.

$$S_k^l = S_k^l + 1 \tag{5.24}$$

No caso b), a amostra x_k não está próxima a nenhuma das nuvens de dados existentes, sendo considerado, então, um *outlier temporário*. Após um determinado período de tempo, um certo número de *outliers* suficientemente próximos uns dos outros podem vir a formar uma nova nuvem. Diferentemente do eClass0, AutoClass armazena os *outliers* em um pequeno vetor, $\vec{\mathcal{O}}$, evitando o descarte imediato de um *outlier* que, posteriormente, pode passar a pertencer a uma nuvem de dados. Note que o vetor citado não aumenta consideravelmente o esforço computacional, uma vez que o seu tamanho, que é um parâmetro definido pelo usuário, deve ser bastante limitado. Recomenda-se o tamanho máximo do vetor $\vec{\mathcal{O}}$ definido por

$$\text{maxsize}_{\vec{\mathcal{O}}} = min(100, 0.05k) \tag{5.25}$$

onde tais valores representam um bom custo-benefício entre acessibilidade das amostras passadas e memória necessária para execução. Se, após a leitura de uma nova amostra x_k, o tamanho do vetor é excedido, a amostra mais antiga é removida de $\vec{\mathcal{O}}$.

Depois da atualização de $\vec{\mathcal{O}}$, dois cenários podem ocorrer: i) a amostra é, de fato, um *outlier*, e é temporariamente ignorada, ou ii) existe uma quantidade de *outliers* próximos uns dos outros armazenados suficiente para a criação de uma nova nuvem de dados **e** a densidade desta nuvem em potencial é maior que a média das densidades de todas as nuvens existentes. O número de *outliers* próximos necessário para a criação de uma nova nuvem de dados é um parâmetro definido pelo operador. Recomenda-se o uso de

$$\text{minpoints}_{\vec{\mathcal{O}}} = max(3, 0.15S_k^q) \tag{5.26}$$

onde S_k^q é o número de pontos associados à nuvem menos populosa. Dessa maneira, a formação de uma nuvem dependerá, não somente, de um número mínimo fixo (neste caso, >= 3) de *outliers* próximos, mas também do tamanho das nuvens existentes e, consequentemente, do número de amostras lidas até o momento, evitando disparidades de tamanho entre as nuvens existentes e as a serem criadas. A densidade também é um fator crucial a ser considerado, uma vez que, juntamente com o número de pontos, reflete a informatividade da nova nuvem. Aqui, foi utillizado o conceito de densidade relativa local, medida de cada amostra para cada nuvem existente, e calculada pela equação (5.18).

Se as duas condições (o número de *outliers* próximos uns dos outros é maior que $minpoints_{\vec{\mathcal{O}}}$ e a densidade da nuvem candidata é maior que a média das densidades de todas as nuvens) são satisfeitas, os seguintes passos serão executados pelo AutoClass:

- Uma nova nuvem de dados $\aleph^{R+1}$ é criada.

- O ponto focal $X^{(R+1)*}$ da nova nuvem é definido como a média de todos as amostras associadas a $\aleph^{R+1}$ (antigos *outliers*)

$$X^{R+1} = \frac{1}{m} \sum_{j=1}^{m} \mathcal{O}^j \tag{5.27}$$

  onde m é o número de *outliers* próximos, que irão compor a nuvem $\aleph^{R+1}$.
- A zona de influência r^{R+1} é definida pela média entre i) a zona de influência de todas as nuvens existentes, e ii) o valor de r_0, definido no início do algoritmo pelo usuário. Note que essa relação considera tanto as zonas de influência calculadas e atualizadas, quanto o valor inicial, fixo, definido pelo usuário. Essa é uma característica conservativa do AutoClass, buscando mesclar a base de conhecimentos atual do sistema com a *expertise* do operador.
- O número de amostras S_k^{R+1} sob influência da nova nuvem é o número de *outliers* próximos, m.
- Os antigos *outliers*, que são agora parte da nuvem $\aleph^{R+1}$, são removidos do vetor $\vec{\mathcal{O}}$.
- A nuvem de dados $\aleph^{R+1}$, como parte antecedente, juntamente com o rótulo $Classe^{R+1}$, como parte consequente, formam a nova regra de inferência:

$$\mathcal{R}^{R+1} : \textbf{Se } \left(\vec{x} \sim \aleph^{R+1}\right) \textbf{ Então } \left(\text{Classe}^{R+1}\right) \tag{5.28}$$

- Então, o número de nuvens de dados existentes é incrementado ($R \leftarrow R + 1$).

Finalmente, o contador k é incrementado em uma unidade ($k \leftarrow k + 1$) e o algoritmo continua com a leitura da próxima amostra, x_k. O procedimento completo é detalhado no Algoritmo (2).

Algoritmo 2: AutoClass

$R \leftarrow 0$; $r_0 \leftarrow$ **leia** do usuário; $k \leftarrow 1$;
maxsize$_{\vec{\mathcal{O}}} \leftarrow$ **leia** do usuário; $minpoints_{\vec{\mathcal{O}}} \leftarrow$ **leia** do usuário;
while $x_k \leftarrow$ **leia** *próxima amostra de dados* **do**
  if $k = 1$ **then**
    /* Criação da primeira nuvem */
    $\aleph^{R+1} \leftarrow$ **nova** nuvem de dados;
    $X^{(R+1)*} \leftarrow x_k$; $r^{R+1} \leftarrow r_0$; $S_k^{R+1} \leftarrow 1$; $R \leftarrow 1$;
  else
    for $l \leftarrow 1$ **to** R **do**
      if x_k *está* **próximo** *a* $\aleph^l$ **then**
        /* dentro de duas vezes a zona de influência da nuvem */
        $X^{l*} \leftarrow$ **atualiza** ponto focal; /* equação (5.22) */
        $r^l \leftarrow$ **atualiza** zona de influência; /* equação (5.23) */;
        $S_k^l \leftarrow S_k^l + 1$;
      end
    end
    if x_k **não** *está* **próximo** *a nenhuma das nuvens existentes* **then**
      $\vec{\mathcal{O}}.adicione(x_k)$;
      if $\vec{\mathcal{O}}.tamanho > maxsize_{\vec{\mathcal{O}}}$ **then** $\vec{\mathcal{O}}.remova(0)$;
      $m \leftarrow$ número **máximo** de *outliers* próximos uns dos outros em $\vec{\mathcal{O}}$;
      $\aleph^q \leftarrow$ nuvem **menos populosa** dentre todas as existentes;
      $d_{max} \leftarrow$ **maior densidade** dentre as nuvens em potencial de $\vec{\mathcal{O}}$;
      $\mu_{density} \leftarrow$ **média** das densidades de todas as nuvens existentes;
      if $m > minpoints_{\vec{\mathcal{O}}}$ **and** $d_{max} > \mu_{density}$ **then**
        /* Cria uma nova nuvem */
        $\aleph^{R+1} =$ **nova** nuvem de dados;
        $X^{(R+1)*} \leftarrow$ média dos antigos *outliers*; /* equação (5.27) */
        $\mu_r \leftarrow$ **média** das zona de influências de todas as nuvens;
        $r^{R+1} = \mu_r * 0.5 + r_0 * 0.5$; $S_k^{R+1} =$ m;
        remova todos os antigos *outliers* de $\vec{\mathcal{O}}$.
      else
        /* É um *outlier*, não faça nada. */
      end
    end
  end
  $k \leftarrow k + 1$;
end

5.4 Identificação de falhas utilizando AutoClass

Identificação de falhas, o segundo estágio do esquema de DDF, pode ser tratado como um problema de classificação. A ideia geral da abordagem proposta é selecio-

nar *a priori* características específicas e representativas, que podem ser variáveis do processo ou atributos calculados, e clusterizar/dividir, *on-line*, os dados de entrada no espaço n-dimensional de características. O algoritmo AutoClass, na abordagem proposta, é responsável pela geração e atualização das regras de inferência *fuzzy*, de uma maneira não-supervisionada, criando, assim, diferentes classes às quais as amostras de dados lidas serão atribuídas.

AutoClass, como um classificador evolutivo e, diferentemente dos modelos *fuzzy* tradicionais, é capaz de modificar a sua estrutura, crescer e atualizar quando necessário, apresentando, assim, um nível mais alto de adaptação (Angelov & Kasabov, 2006). Isso significa que as regras de inferência, e não apenas os parâmetros, podem ser criadas ou atualizadas a cada iteração do algoritmo, representando novos tipos de falha descobertas a partir do padrão dos dados, de forma autônoma.

O principal objetivo da abordagem é separar espacialmente os dados em diferentes estados/regimes de operação da planta, agrupando dados similares no mesmo grupo. Deve-se ressaltar a importância do procedimento de seleção das características, como mencionado anteriormente. A escolha de quais variáveis/atributos do processo devem ser monitoradas é fator crucial no desenvolvimento de um sistema de classificação. As características selecionadas precisam refletir as diferenças entre os diferentes pontos de operação da planta, e é preciso alcançar um bom custo-benefício entre o número de características selecionadas e o esforço computacional. Enquanto um grande número de características garantem uma representação mais realística dos dados, os requisitos computacionais podem ser proibitivos. Por outro lado, com um número pequeno de características, o sistema pode não ser capaz de distinguir diferentes classes, enquanto mantém um esforço computacional mínimo.

A cada iteração, se o algoritmo de detecção (primeiro estágio) dispara um estado de falha, AutoClass recebe como entrada o vetor de dados $x = [x_1, x_2, ..., x_n]$, para n características selecionadas. Percebe-se que, embora a proposta apresente uma abordagem de 2 estágios para detecção e identificação, ambos os estágios podem ser utilizados separadamente com outras abordagens existentes, o que, de fato, será mostrado nos próximos capítulos. A falha detectada poderá, então, ser automaticamente associada a um estado de falha já encontrado, ou uma nova nuvem de dados/regra iniciada.

Deve ser ressaltado que os rótulos de classe são gerados automaticamente, em sequência, à medida que diferentes classes/falhas são detectadas. Obviamente, esses rótulos não representam o tipo ou localização da falha, porém, são bastante úteis para distinguir diferentes falhas. Uma vez que não existe fase de treinamento ou pré-definição de falhas ou modelos, a rotulação correta pode ser executada de uma maneira semi-supervisionada pelo operador humano, sem a necessidade de ação imediata/sincronizada.

A identificação não-supervisionada é uma abordagem teórica. Na prática, principalmente em ambientes industriais reais, o operador deve ter o controle da operação, ainda que com a menor necessidade de intervenção possível. Digamos, por exemplo, que dois *streams* de dados de falhas diferentes, utilizando duas características ($n = 2$) são lidos e processados por AutoClass. Devido ao formato da distribuição

espacial, os dados são classificados como na Figura 5.1.

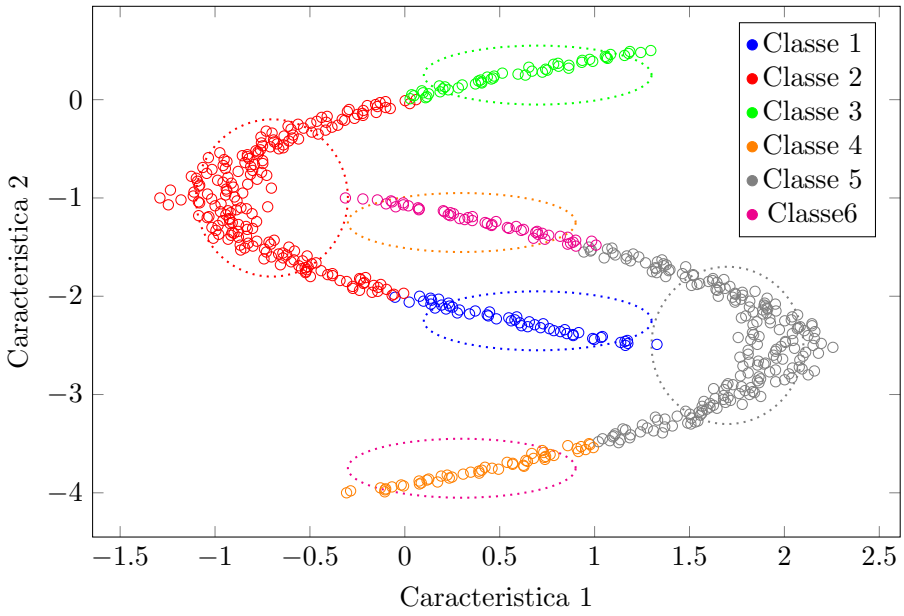

Figura 5.1: Classificação não-supervisionada de falhas

É possível notar que, apesar da distribuição dos dados visualmente indicar que a classificação correta deve utilizar duas classes de falha, o algoritmo resultou em saída com 6 classes distintas. O operador, por sua vez, pode, a qualquer momento (não necessariamentente imediatamente), aplicar a rotulação correta, que de fato, indique o tipo, o local e a intensidade da falha. Seguindo o exemplo, após a rotulação do operador, o algoritmo proposto exibe a nova saída de classificação, como na Figura 5.2.

Na última image, as classes “Classe 1”, “Classe 2” e “Classe 3” foram mescladas no novo rótulo “Falha - Tipo A”, e as falhas “Classe 4”, “Classe 5” e “Classe 6” passaram a compor o rótulo “Falha - Tipo B”.

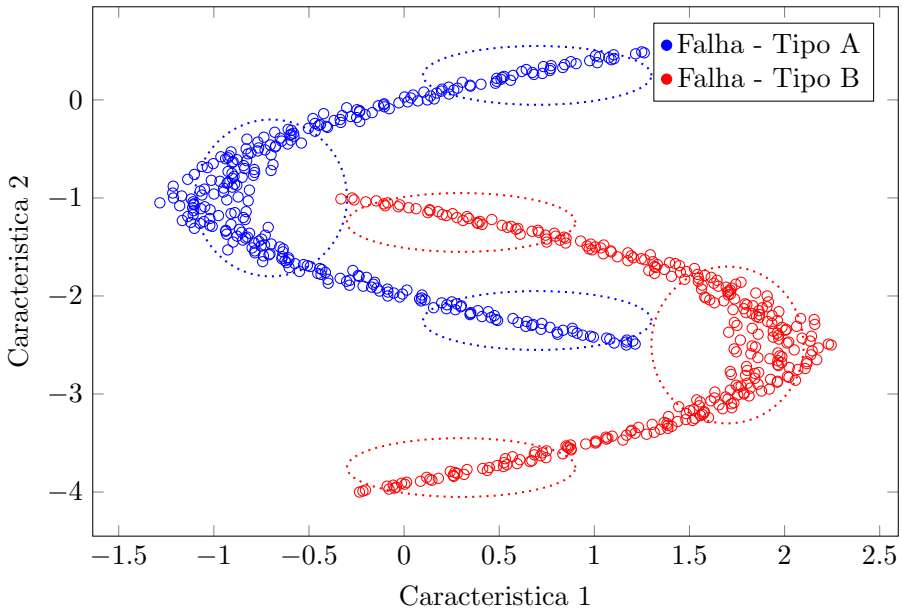

Figura 5.2: Classificação após a intervenção do operador

Capítulo 6

Configuração do Experimento

Para validação da proposta, duas abordagens práticas foram realizadas. Na primeira parte, uma aplicação simulada de um sistema de tanques acoplados foi desenvolvida, e o desempenho dos algoritmos AutoClass e eClass0 são avaliados no estágio de identificação de falhas. Na segunda parte, uma planta experimental real foi utilizada. Nesse experimento, o algoritmo de detecção baseado em EDR foi comparado ao conhecido algoritmo de CEP, no primeiro estágio, e AutoClass e eClass0 foram utilizados novamente, no segundo estágio.

6.1 Experimentos com processo simulado

O problema escolhido é apresentado e amplamente descrito em Costa et al. (2013), tendo sido utilizado em várias e diferentes aplicações (Costa et al., 2010), (Costa et al., 2012), (Canureci et al., 2008). A planta em questão possui dois tanques acoplados, e foi desenvolvida pela Quanser (Quanser, 2004).

A planta consiste de uma bomba, dois tanques e um reservatório de água. Os tanques, montados no painel frontal, são configurados de modo que o líquido do primeiro tanque (saída 1) flua para o segunto tanque, por gravidade, e do segundo tanque (saída 2), flua para o reservatório principal. A bomba, por sua vez, suga a água do reservatório para o primeiro tanque. A planta é mostrada na Figura 6.1.

Por propósitos didáticos, nesse experimento, utilizamos a versão simulada da planta em questão, a qual o comportamento é bastante similar à versão física equivalente, no entanto, livre de ruídos de ambiente e perturbações não previstas. Embora a planta permita controle de segunda ordem, este experimento trata apenas da aplicação de primeira ordem, ou seja, a monitoração do nível no primeiro tanque.

O sistema consiste, então, de duas variáveis: 1) a tensão/sinal de controle (u) aplicada ao motor da bomba que, por questões de segurança, é limitada de 0 a 15V DC, e 2) o nível (y) do tanque 1, que pode variar de 0 a 30cm.

O modelo matemático do tanque é descrito em Meneghetti (2004). O fluxo fornecido pela bomba, que é alimentada por um motor DC, é diretamente proporcional à tensão aplicada ao motor. Para uma configuração de primeira ordem, toda a água flui para o tanque 1. A variável que representa esse fluxo é chamada de $\mathcal{F}_1$, e é calculada por

Figura 6.1: Planta didática da Quanser

$$\mathcal{F}_1^{entrada} = K_m V_p [cm^3/s] \tag{6.1}$$

onde V_p é a tensão aplicada ao motor e K_m é a constante da bomba, que, nesse caso, é $K_m = 250$.

A velocidade na qual o líquido flui através do orifício de saida é dada pela equação de Bernoulli para orifícios pequenos (Cengel et al., 2012):

$$v^{saida} = \sqrt{2gL_1}[cm/s] \tag{6.2}$$

onde g é a aceleração da gravidade em cm/s^2 e L_1 é o nível da água no tanque 1.

O fluxo de saída $\mathcal{F}_1^{saida}$ é calculado por

$$\mathcal{F}_1^{saida} = a_1 v^{saida} [cm^3/s] \tag{6.3}$$

onde a_1 é a área do orifício do tanque 1 em cm^2, que, nesse caso, é $a_1 = 0.47625 cm^2$.

A taxa de variação do nível no tanque 1 ($\dot{L}_1$) é dada pela razão entre a variação volumétrica ($\dot{V}$) e a área da base do tanque (A_1), para $\dot{V} = \mathcal{F}_1^{entrada} - \mathcal{F}_1^{saida}$.

Vale ressaltar que, em momento algum na abordagem proposta, o modelo matemático é necessário para a tarefa de detecção e identificação de falhas, uma vez que os algoritmos são completamente baseados nos dados lidos. O mesmo é apenas apresentado para fins didáticos.

O controle da planta é, então, realizado por um controlador PID bastante simples, no qual o sinal de controle é calculado por

$$u(t) = K_p e(t) + K_i \int_0^{\tau} e(\tau) d(\tau) + K_d \frac{d}{dt} e(t) \tag{6.4}$$

onde K_p é o ganho proporcional, K_i é o ganho integral, K_d é o ganho derivativo, e é o erro, calculado pela diferença da referência r e o nível real do tanque, t é o instante de tempo e τ é a variável de integração. Nessa aplicação, o período de amostragem é de $100ms$, $r = 5cm$, $K_p = 10$, $K_i = 0,1$ e $K_d = 0,1$. O gráfico resultante do controle é mostrado na Figura 6.2, e serve de referência para o caso livre de falhas (modo de operação normal).

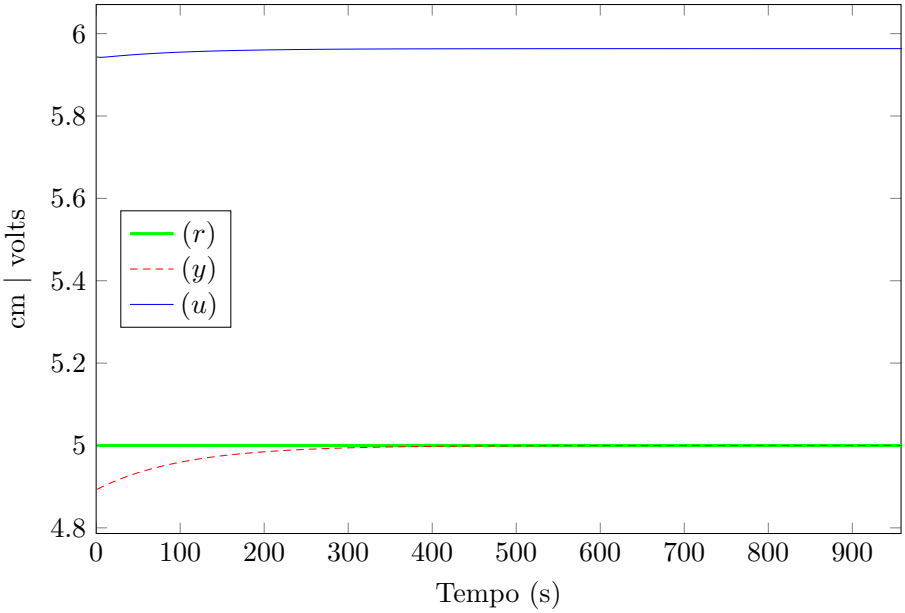

Figura 6.2: Planta simulada no estado normal de operação

É importante ressaltar que consideraremos aqui apenas o processo em regime estacionário, sendo, então, o regime transitório, após uma mudança de *set-point*, ignorado. O nível (variável observável), atinge a referência e permanece estável, com erro ($e = r - y$) próximo de zero e sem oscilação significante. Da mesma maneira, o sinal de controle é praticamente constante, ignorando-se as pequenas oscilações ruidosas intrínsecas aos ambientes industriais.

O objeto deste estudo é um conjunto de 6 falhas, geradas sempre entre as amostras 500 e 800, sendo todas referentes ao atuador (bomba). Esse grupo contém experimentos com diferentes padrões e níveis, sendo 3 níveis da falha *off-set positivo* e 3 níveis da falha *saturação*. As falhas geradas são descritas na Tabela 6.1.

O experimento utilizando a planta simulada foi realizado somente no estágio de classificação, onde o algoritmo proposto AutoClass é comparado com o seu antecessor, eClass0. Nesse estágio, os algoritmos AutoClass e eClass0 foram utilizados para monitoração dos atributos *erro* (e) e *sinal de controle* (u). Uma vez que, no estado normal de operação, o sistema assume um comportamento praticamente constante e estável, oscilações ou variações nesses sinais vêm a representar estados de falha da

Tabela 6.1: Conjunto de falhas geradas para o experimento simulado

ID	Grupo	Tipo	Nível
G_1	Atuador	*Off-set* positivo	1V
G_2			2.5V
G_3			4V
G_4		Saturação	5V
G_5			4V
G_6			3V

planta.

6.2 Experimentos com processo real

A segunda parte dos experimentos se deu com a aplicação dos algoritmos propostos a uma planta piloto de controle industrial desenvolvida pela DeLorenzo (Marins, 2009). A planta piloto permite o estudo de controle de processos contínuos, baseado em quatro variáveis típicas, *pressão*, *temperatura*, *vazão* e *nível*, definidas aqui pelo vetor de entrada $S = (u, t, f, y)$.

A planta piloto inclui (DeLorenzo, 2009): indicadores que convertem o sinal físico em elétrico, a ser processado pelo controlador lógico programável (CLP); um barramento terminal, onde todos os sinais elétricos estão disponíveis para o controlador externo; *software* supervisório e de aquisição de dados (SCADA) para configuração paramétrica e visualização do processo. A planta é composta por: um painel com CLP e todos os componentes elétricos para o controle da planta; dois reservatórios pressurizados, um feito de acrílico, T_1, e o outro feito de aço inoxidável, T_2; uma bomba de recirculação centrífuga controlada por um inversor de frequência; um sistema de aquecimento e troca de calor; duas válvulas direcionais, V_1 e V_2; sensores de temperatura, pressão, vazão e nível. A Figura 6.3 mostra a planta piloto utilizada.

Os dois tanques são conectados por um sistema de tubulação que permite o fluxo entre eles. A planta funciona de maneira que é possível transferir o líquido entre ambos os tanques, estando T_1 posicionado acima de T_2 em relação ao nível do solo. O líquido flui sempre em uma direção: de T_1 para T_2 por gravidade, e de T_2 para T_1 a partir da pressão gerada na bomba centrífuga. A Figura 6.4 ilustra o diagrama esquemático da planta (Costa et al., 2013).

Este trabalho considerou apenas a aplicação de controle de nível. A planta foi controlada por um controlador *fuzzy* multiestágios, desenvolvido no *software* JFuzZ (Costa et al., 2010), sendo a comunicação realizada através de uma interface OPC (*OLE for Process Control*) (Liu et al., 2005), (Schwarz & Boercsoek, 2007). O comportamento gerado pelo controlador representa o estado de operação "normal" da planta. Os detalhes de implementação do controlador são apresentados em (Costa et al., 2012). A Figura 6.5 ilustra as variáveis *nível* (y, variável observável),

Figura 6.3: Planta piloto utilizada

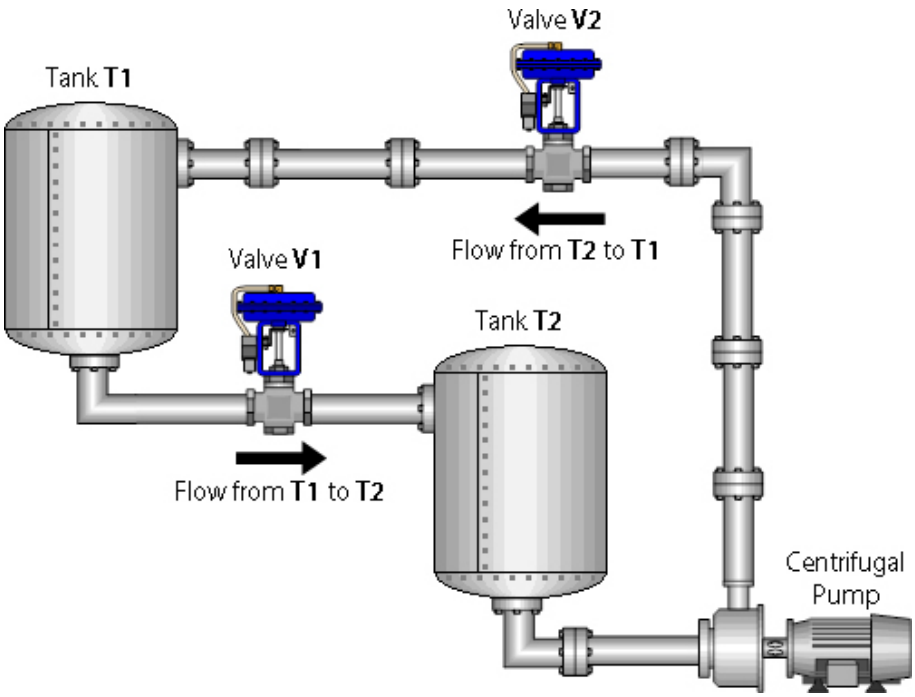

Figura 6.4: Diagrama esquemático da planta

referência (r, *set-point* definido pelo usuário) e *pressão* (u, sinal de controle) na bomba, formando o vetor $x = (r, y, u)$, para $r = 0,5$ (50% da capacidade máxima do tanque), dentro do estado normal de operação.

Deve-se notar que consideraremos aqui apenas o processo em regime estacionário, sendo, então, o regime transitório, novamente ignorado. O nível (variável ob-

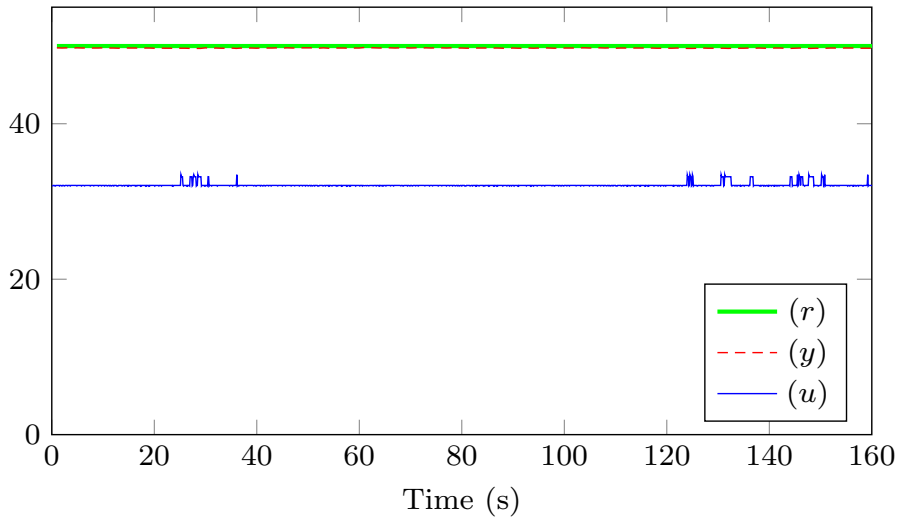

Figura 6.5: Planta piloto no estado normal de operação

servável), atinge a referência e permanece estável, com erro ($e = r - y$) próximo de zero e sem oscilação significativa. Da mesma maneira, o sinal de controle é praticamente constante, ignorando-se as pequenas oscilações ruidosas intrínsecas aos ambientes industriais. A partir de agora o padrão dinâmico apresentado na Figura 6.5 será a referência para a planta em questão, e variações significativas nos sinais podem ser interpretadas como falhas.

O objeto deste estudo é um conjunto de 16 falhas diferentes, sendo a maioria gerada fisicamente na planta piloto. As falhas são divididas em três grupos: atuador, estrutural e perturbação.

Cada grupo contém experimentos com diferentes padrões e níveis. No grupo "atuador", existem 6 níveis (3 positivos e 3 negativos) de *off-set* na bomba; no grupo "estrutural" existem 3 níveis de abertura de dreno, que simulam fisicamente um vazamento no tanque T_1, e 3 níveis de emperramento para cada uma das válvulas; No grupo "perturbação", existe uma perturbação do ambiente através da adição manual de água ao tanque. Todas as falhas geradas são descritas na Tabela 6.2.

O experimento utilizando a planta industrial real foi dividido em duas partes: 1) Detecção de falhas, onde o algoritmo proposto de detecção baseado em EDR é comparado com a abordagem estatística CEP, e 2) classificação de falhas, onde o algoritmo proposto AutoClass é comparado com o seu antecessor, eClass0.

No estágio de detecção, os algoritmos baseados em EDR e CEP foram utilizados para monitoração das variáveis *erro* (e) e *sinal de controle* (u). Uma vez que ambas as abordagens são sensíveis a variações/oscilações nos sinais, a utilização de variáveis representativas é indicada. Por "representativa" entende-se uma variável que permanece constante (ou próximo de constante) no estado normal de operação, e oscilatória na presença de uma falha. Tal indicação explica a escolha das variáveis citadas.

Quanto ao estágio de diagnóstico, os algoritmos AutoClass e eClass0 foram uti-

Tabela 6.2: Conjunto de falhas geradas para o experimento real

ID	**Grupo**	**Tipo**	**Nível**
F_1	Atuador	*Off-set* positivo	+2%
F_2			+4%
F_3			+8%
F_4		*Off-set negativo*	-2%
F_5			-4%
F_6			-8%
F_7	Estrutural	Vazamento no tanque T_1	33%
F_8			66%
F_9			100%
F_{10}		Válvula V_1 emperrada	30%
F_{11}			50%
F_{12}			85%
F_{13}		Válvula V_2 emperrada	25%
F_{14}			50%
F_{15}			75%
F_{16}	Perturbação	Perturbação no ambiente	Baixo

lizados para monitoração dos atributos *período* (Característica 1) e *amplitude* (Característica 2) do sinal de controle. Uma vez que, nas falhas geradas, o sinal assume um comportamento periódico ligeiramente diferente para cada falha em termos de frequência e amplitude, tais atributos tornam-se bastante representativos na diferenciação de diferentes estados de falha.

Capítulo 7

Resultados Obtidos

O experimento foi dividido em dois estágios: (7.1) detecção, e (7.2) classificação, onde conjuntos de diferentes falhas foram analisados separadamente.

7.1 Estágio de detecção

Para fins de comparação, primeiramente, um conjunto de dados de falha foi analisado através do uso de uma aplicação de CEP (Hossain et al., 1996), (Martin et al., 1996), (Cook et al., 1997), que trata-se de um algoritmo bem conhecido para detecção de *outliers* em processos industriais. Os detalhes do procedimento foram exaustivamente apresentados na literatura, com referências no Capítulo 4.

Neste trabalho, o algoritmo de CEP foi implementado na linguagem Java e executado *on-line*, com 100 amostras de dados para cada iteração do algorimo, o que representa uma janela de tempo de 10 segundos do processo (frequência de 10Hz). No estágio de detecção, a comparação dos algoritmos CEP e EDR foi realizada utilizando-se apenas os dados da planta piloto real. As variáveis monitoradas no experimento são os sinais de *controle* (u) e *erro* (e).

Os resultados para a abordagem baseada em CEP são apresentados como gráficos do tipo *X-Bar* (Hossain et al., 1996). Um gráfico X-Bar mostra o comportamento da variável monitorada no decorrer do tempo, os seus limites superior e inferior. A Figura 7.1 ilustra os gráficos X-Bar resultantes para o sinal de controle (imagem superior) e para o erro (imagem inferior) na falha F_9 (vazamento em T_1 de 100%), enquanto a Figura 7.2 apresenta os mesmos sinais para a falha F_{10} (válvula V_1 emperrada em 50%).

Após a primeira rodada de experimentos, os mesmos dados foram analisados pelo algoritmo baseado em EDR. O algoritmo proposto também foi implementado em Java e executato *on-line*. As variáveis monitoradas também são o *sinal de controle* (u) e o *erro* (e). A Figura 7.3 apresenta os gráficos resultantes para o sinal de controle e erro (imagem superior), representado pelos sinais de referência (r) e nível (y), bem como a evolução da densidade calculada a cada iteração (imagem inferior), novamente para a falha F_9. A Figura 7.4, por sua vez, mostra os mesmos sinais da figura anterior, porém, neste caso, para a falha F_{10}. Note que as barras verticais tracejadas preta e cinza indicam, respectivamente, o início e fim de um

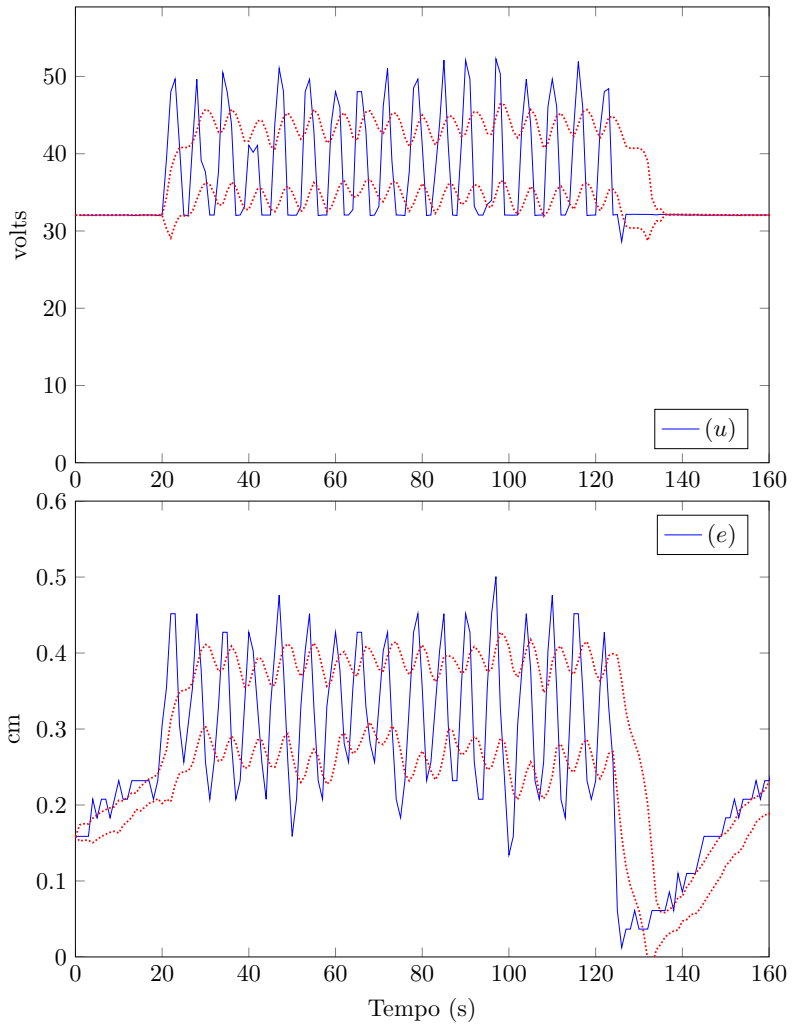

Figura 7.1: Detecção da falha F_9 utilizando CEP

estado de falha, calculados pelo algoritmo proposto nesta tese.

Para fins de comparação, foram analisados aqui: i) as taxas de acerto/erro de detecção, que são complementares, e calculadas pela soma dos acertos/erros em comparação com a classificação correta das falhas, ambas quando o sistema está operando normalmente ou com falha, e ii) o tempo de execução de ambos os algoritmos na mesma máquina, sob as mesmas condições de *hardware* e sistema operacional. Os resultados para todos os 16 experimentos executados na planta piloto, para as abordagens baseadas em CEP e EDR, são detalhados na Tabela 7.1.

Enquanto ambas as abordagens utilizadas nos experimentos são *on-line* e *data-driven*, a detecção é realizada de maneira bem diferente, com a abordagem baseada em EDR demonstrando uma grande melhoria quando comparada à abordagem baseada em CEP. Enquanto o algoritmo CEP obteve um total de 55.37% de acertos, no algoritmo EDR, tal valor foi de 84.76%. Individualmente, a segunda aplicação também demonstrou melhores resultados, para 15 de 16 diferentes falhas analisadas.

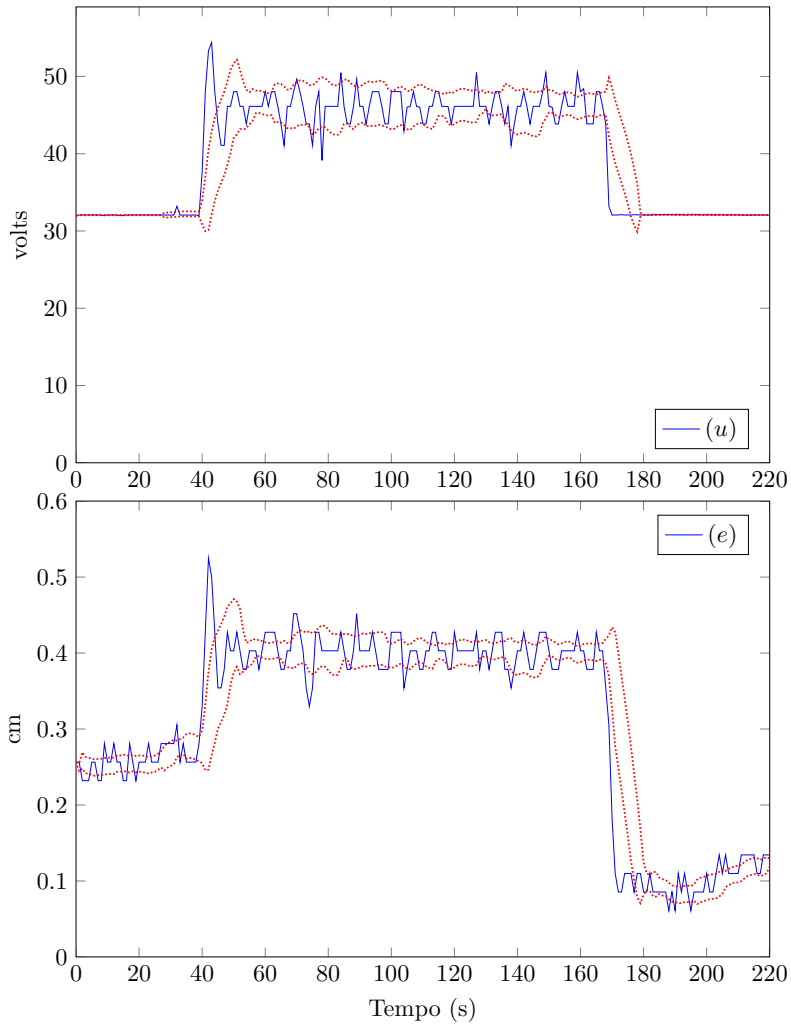

Figura 7.2: Detecção da falha F_{10} utilizando CEP

Da mesma maneira, devemos notar a natureza robusta da proposta. Enquanto é possível identificar visualmente nos gráficos diversos chaveamentos do estado "normal" para "falha" (e vice-versa) durante a execução do algoritmo CEP, a aplicação baseada em EDR é, claramente, mais conservadora na decisão de quando entrar ou sair de um estado de falha. Nesse sentido, o sistema de detecção proposto tenta ignorar sinais transitórios e apresenta um alerta bem mais preciso ao usuário.

Outro aspecto a ser considerado nessa comparação é o tempo de execução dos dois algoritmos. O tempo total de execução para os 16 arquivos de amostras de dados no detector proposto foi 22,23% menor que no algoritmo de CEP. A maior razão para o desempenho otimizado é que o algoritmo EDR não necessita armazenar amostras de dados passadas, nem executar cálculos *off-line*, tais como média e desvio padrão. Nessa abordagem, a densidade e a sua média são calculadas de maneira recursiva, e tais valores são atualizados a cada iteração, sem a necessidade de armazenamento de dados passados.

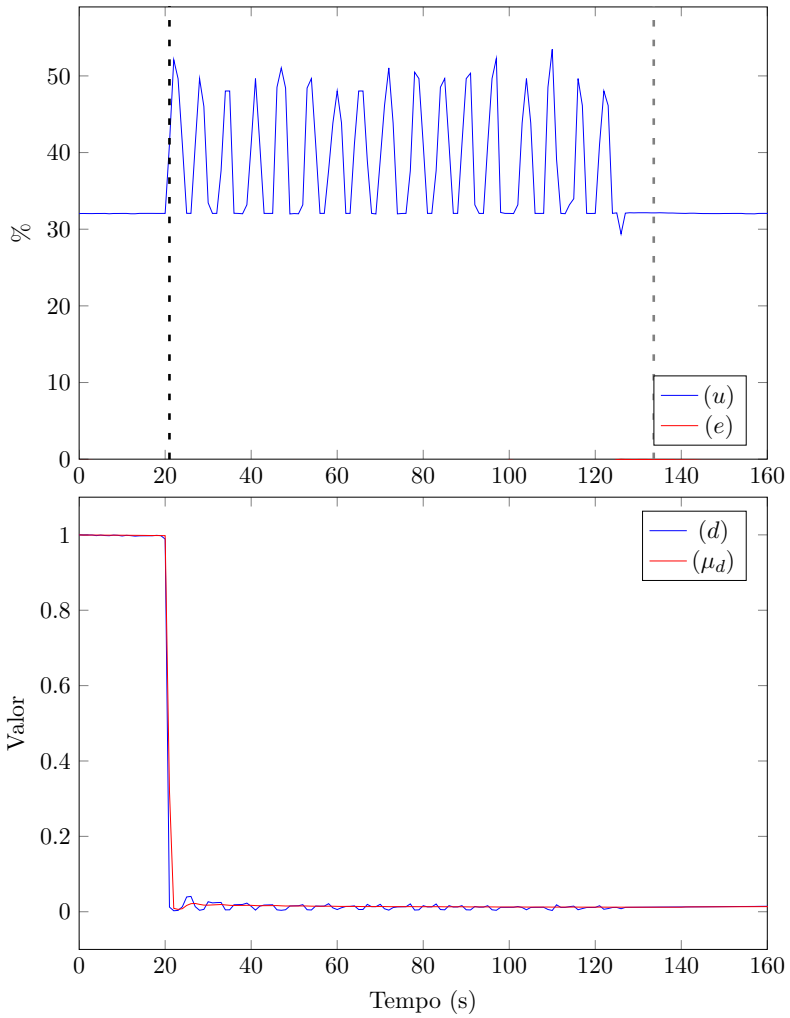

Figura 7.3: Detecção da falha F_9 utilizando EDR

É importante ressaltar que a ideia principal da abordagem proposta, bem como em outras abordagens executadas inteiramente de maneira *on-line* e sem estágios de treinamento, é baseada no conceito de "normalidade". Isso significa que, por padrão, o algoritmo considerará o ponto de operação mais frequente como o estado livre de falhas. É importante considerar o fato de que, normalmente, estados de falha não são densos. Analisando os gráficos da Figura 7.4, por exemplo, é fácil perceber que, mesmo quando um estado de falha ocorre na maior parte do experimento, o sinal de densidade não cresce continuamente. Assim sendo, na prática, uma falha que ocorre, mesmo que prematuramente, pode ser detectada, se os dados da falha apresentam um comportamento oscilatório (não denso), o que ocorre na maioria das vezes.

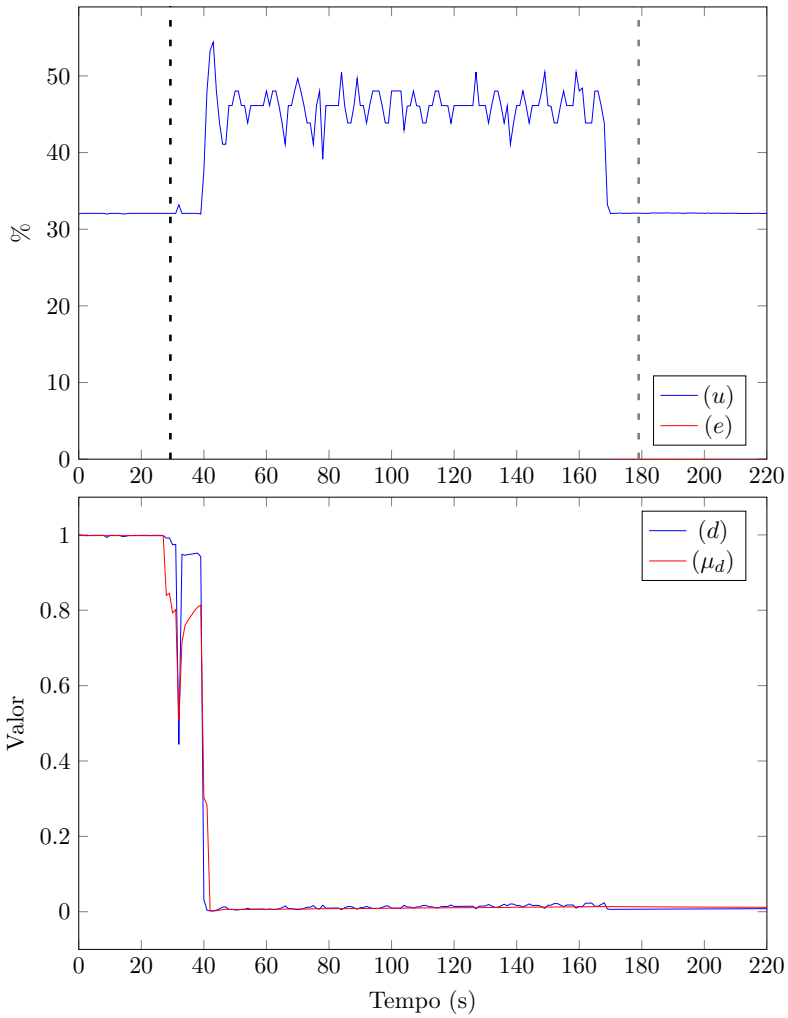

Figura 7.4: Detecção da falha F_{10} utilizando EDR

7.2 Estágio de identificação

O segundo estágio do sistema proposto, que lida com a identificação de falhas, é bastante único no sentido de que é autônomo e de uma maneira não-supervisionada (sem nenhum treinamento ou conhecimento prévio a respeito do processo) identifica/classifica as falhas detectadas. Sendo assim, é difícil compará-la à maioria das abordagens tradicionais existentes.

Nos experimentos deste estágio, utilizamos dados, tanto da planta simulada, quanto da planta real. Em ambas as situações, o algoritmo AutoClass (proposto) é comparado com o eClass0. Foram considerados grandes *streams* de dados de falhas sequenciais. O processo de classificação é realizado a partir "do zero", com a leitura da primeira amostra de dados adquirida, o que significa uma base de regras *fuzzy* vazia, para ambos os algoritmos. Note que a detecção é realizada pelo algoritmo baseado em EDR, já descrito, e os algoritmos AutoClass e eClass0 são ativados somente se o sistema detecta uma falha. O progresso da execução e

Tabela 7.1: Comparação entre os algoritmos baseados em CEP e EDR

Falha	**Execução (ms)**		**Acertos %**		**Erros %**	
	CEP	**EDR**	**CEP**	**EDR**	**CEP**	**EDR**
F_1	728	568	64,13	**97,84**	35,87	2,16
F_2	605	389	71,46	**98,48**	28,54	1,52
F_3	360	271	50,23	**98,63**	49,77	1,37
F_4	284	277	56,66	**96,7**	43,34	3,3
F_5	397	161	61,41	**96,46**	38,59	3,54
F_6	332	308	74,76	**98,48**	25,24	1,52
F_7	221	171	50,29	48,36	49,71	51,64
F_8	570	275	45,46	75,59	54,54	24,41
F_9	293	351	61,64	**95,46**	38,36	4,54
F_{10}	352	247	45,78	**98,93**	54,22	1,07
F_{11}	241	293	62,4	88,61	37,6	11,39
F_{12}	334	173	52,3	77,14	47,7	22,86
F_{13}	302	218	48,25	66,92	51,75	33,08
F_{14}	505	312	33,62	**90,31**	66,38	9,69
F_{15}	290	173	46,21	**98,75**	53,79	35,14
F_{16}	524	145	61,3	64,86	3,7	9,58
Média	**396,1**	**308,1**	**55.37**	**84.76**	**44.63**	**15.24**

comportamento dos sistemas serão ilustrados nos próximos gráficos. Similarmente aos gráficos anteriores, uma barra vertical tracejada indica o momento em que uma falha é detectada e uma barra vertical tracejada cinza indica o momento em que o sistema deixa o estado de falha.

7.2.1 Experimentos com planta simulada

Para os experimentos realizados com a planta simulada, foi utilizado um *stream* de dados contendo amostras de 6 falhas - G_1, G_4, G_2, G_5, G_6 e G_3, respectivamente - , intercaladas por períodos de operação normal.

A Figura 7.5 apresenta o estado do sistema durante o processamento da amostra de dados de número 200 após a detecção da primeira falha. O quadrante superior esquerdo ilustra os sinais de controle e erro, enquanto o quadrante inferior esquerdo apresenta a densidade e sua média. O quadrante superior direito, por sua vez, mostra o gráfico de distribuição espacial em duas dimensões (duas características selecionadas - erro e sinal de controle) e a classificação das amostras de falha através do algoritmo eClass0. Por último, o quadrante inferior direito apresenta a classificação para as mesmas amostras, utilizando, porém, o algoritmo AutoClass. Esse padrão de apresentação é utilizado também nos demais gráficos.

Uma vez que utilizamos aqui um período de amostragem de 100ms, 200 amostras de dados de falha ($k' = 200$, onde k' é a k-ésima amostra de falha do *stream*) signifi-

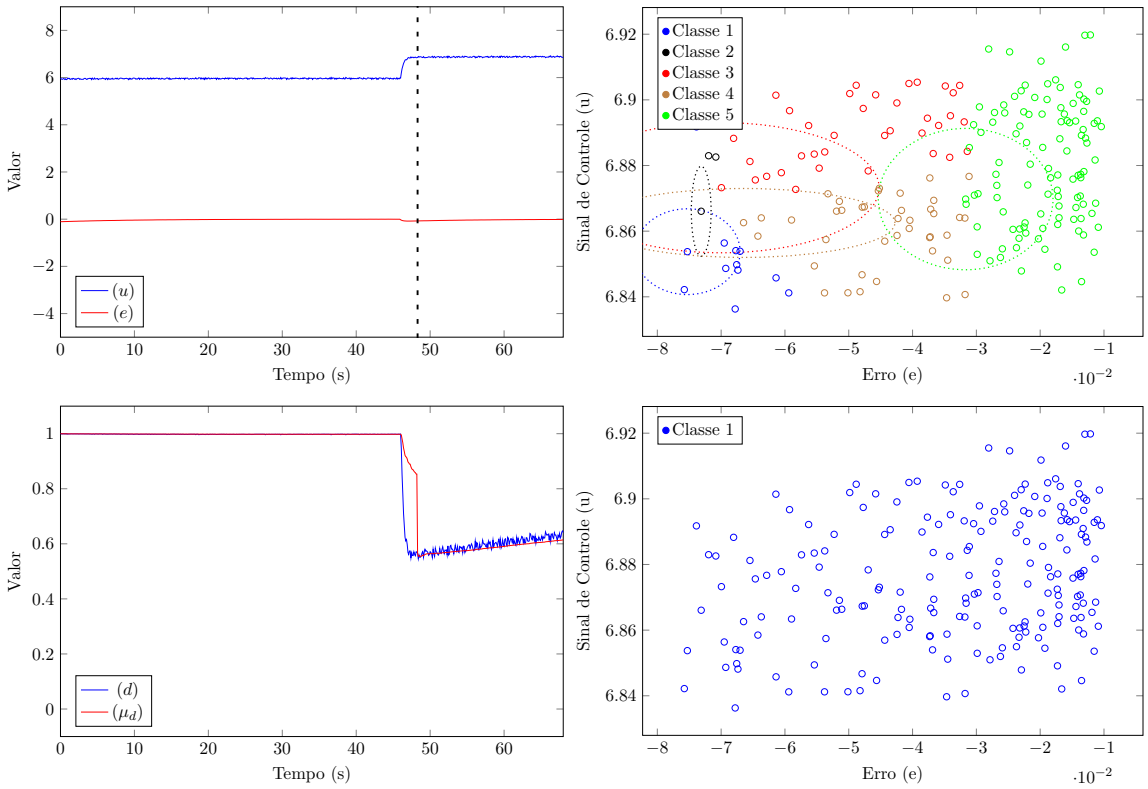

Figura 7.5: Classificação com eClass0 e AutoClass após 200 amostras de falha

cam 20 segundos dentro dessa falha, e assim por diante. Note que a falha G_1 (*off-set* positivo de 1V) é detectada por volta do instante $k = 480$. Enquanto, o algoritmo AutoClass cria apenas uma regra de inferência (uma nuvem de dados) no instante $k' = 1$, o algoritmo eClass0 já tem criado cinco regras de inferência (cinco clusters), nos instantes $k' = 1$, $k' = 4$, $k' = 7$, $k' = 16$ e $k' = 92$, respectivamente. Cada nuvem de dados (para AutoClass) ou cluster (para eClass0) é, automaticamente, nomeada de "Classe i", onde i é o número de classes existentes até o momento mais um. É fácil perceber, então, que a classificação de acordo com o algoritmo eClass0 não representa o real número de falhas identificadas até o momento.

Após o processamento dos primeiros dados de falha, o sistema retorna para o estado "normal" de operação. A próxima falha, G_4 (saturação em 5V), é, então, detectada por volta do instante de tempo $k = 1.450$, como pode ser visto na Figura 7.6.

No instante $k' = 407$, após a detecção de duas novas falhas, G_4 em $k = 1.450$ e G_2 (*off-set* positivo de 2.5V) em $k = 2.200$, uma nova nuvem de dados é criada pelo algoritmo AutoClass, automaticamente nomeada de "Classe 2". Note que, parte das amostras próximas à segunda nuvem ("Classe 2") foram classificadas como pertencentes à nuvem mais antiga ("Classe 1"). Tal fato dá-se em função de, uma vez que a nuvem mais recente ainda não existia, e a quantidade de pontos e densidade local da nuvem candidata ainda não serem suficientes para a sua criação, tais amostras são classificadas como *outliers* temporários, ou ainda, atribuídas à nuvem mais

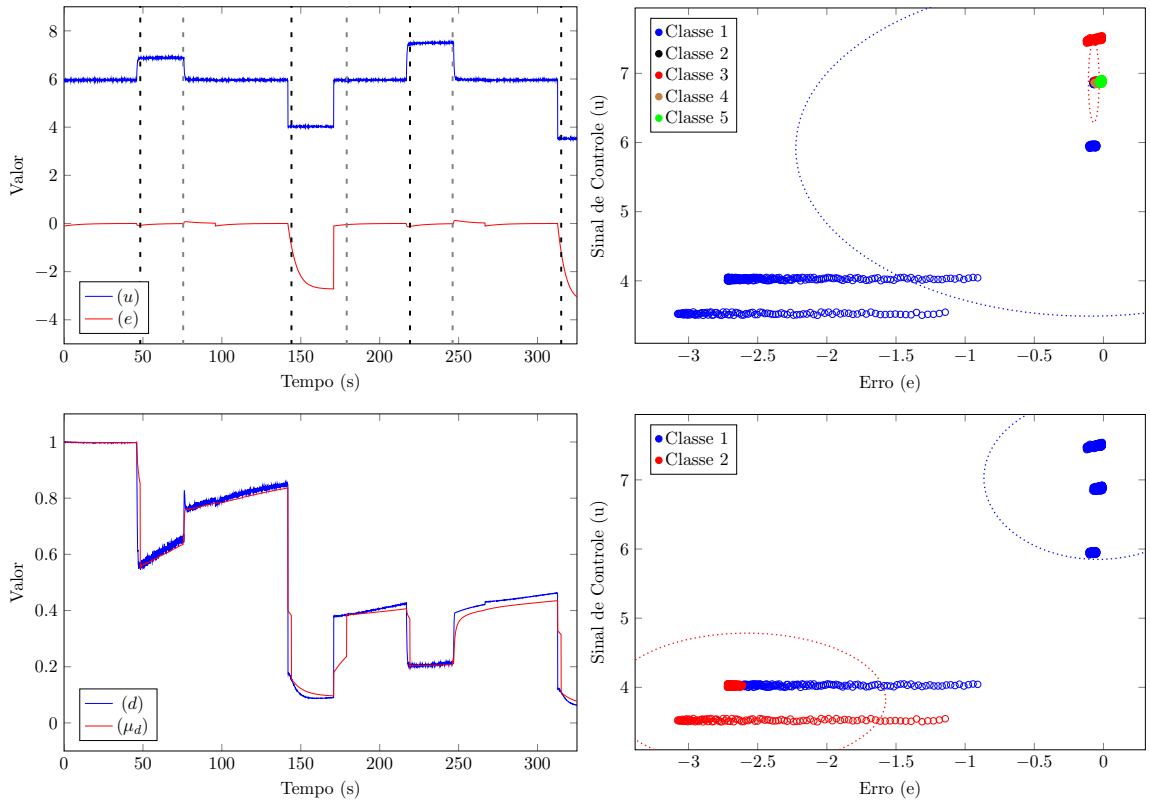

Figura 7.6: Classificação com eClass0 e AutoClass após 1.000 amostras de falha

próxima, mesmo que fora da sua zona de influência. Note também que as falhas G_1 e G_2, apesar de pertencerem a conjuntos de amostras diferentes, e apresentarem-se com níveis diferentes, pertencem à mesma classe, uma vez que ambas representam a falha *off-set* positivo. O algoritmo eClass0, por sua vez, não indica a criação de novos clusters até a presente amostra lida.

Avançando ao fim da leitura do stream, que ocorre por volta do instante $k = 5.000$, percebemos que três novas falhas são detectadas, G_5 (saturação em 4V) em $k = 3.150$, G_6 (saturação em 3V) em 3.880 e G_3 (*off-set* positivo de 4V) em $k = 4.620$, respectivamente, como ilustrado na Figura 7.7.

Nesse caso, nem o algoritmo AutoClass nem o eClass0 indicam a criação de novas regras, trabalhando apenas na atualização da base existente, como pode ser identificado visualmente. É importante mostrar novamente que as falhas G_1, G_2 e G_3 pertencem ao tipo *off-set* positivo, enquanto as falhas G_4, G_5 e G_6 pertencem ao tipo saturação. Sendo assim, o algoritmo AutoClass realizou a classificação correta para a grande maior parte das amostras. O algoritmo eClass0, por sua vez, não apresentou resultados tão precisos.

A base de regras final gerada pelo algoritmo AutoClass, após a execução de 5.000

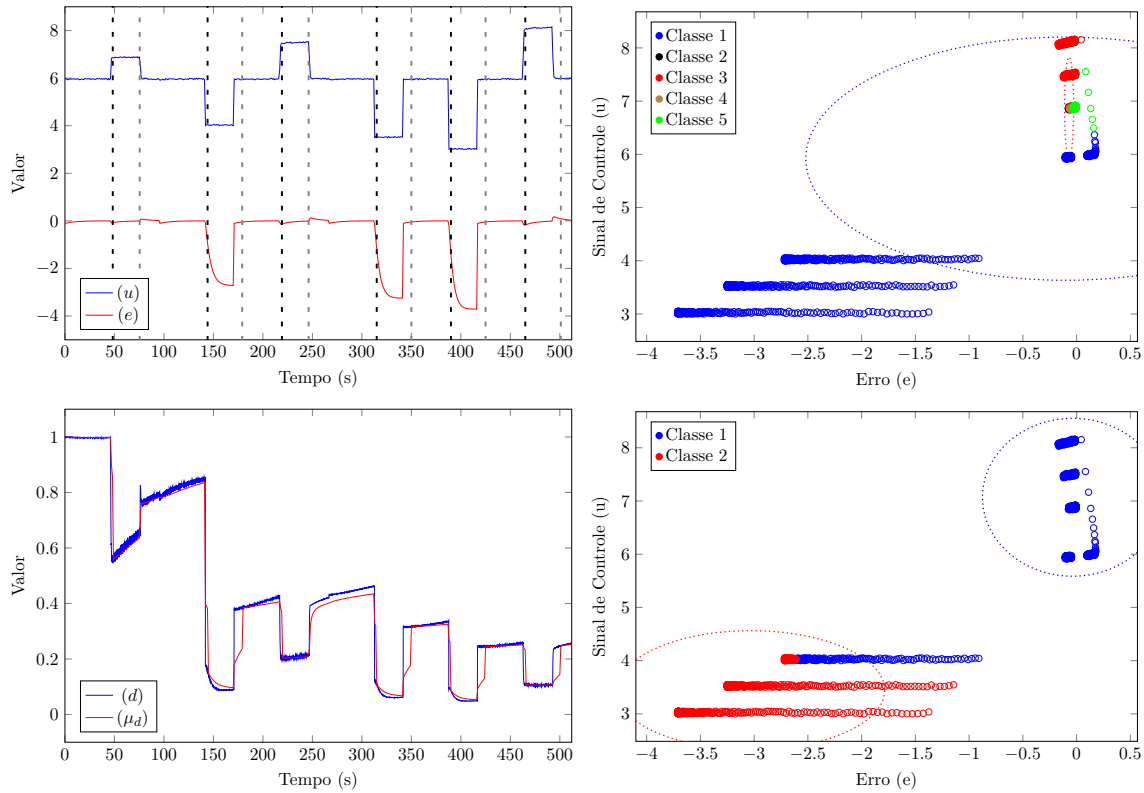

Figura 7.7: Classificação com eClass0 e AutoClass após 1.950 amostras de falha

amostras de dados, é detalhada abaixo.

$$\mathcal{R}^1 : \textbf{Se } \left(\vec{x} \sim \aleph^1\right) \textbf{ Então } \left(\text{Classe}^1\right)$$
$$\mathcal{R}^2 : \textbf{Se } \left(\vec{x} \sim \aleph^2\right) \textbf{ Então } \left(\text{Classe}^2\right)$$

onde

$$X^{1*} = \text{[-0,037; 7,069] e } r^1 = \text{[0,837; 1,483]}$$
$$X^{2*} = \text{[-3,030; 3,442] e } r^2 = \text{[1,244; 1,1205]}$$

sendo X^{i*} o ponto focal e r^i a zona de influência da nuvem $\aleph^i$.

7.2.2 Experimentos com planta real

Para os experimentos executados utilizando-se a planta piloto real, foi utilizado um *stream* de dados contendo amostras de 4 falhas - F_2, F_4, F_1 e F_9, respectivamente - , intercaladas por períodos de operação normal.

A Figura 7.8 apresenta o estado do sistema durante o processamento da amostra de dados de número 300 após a detecção da primeira falha. O quadrante superior esquerdo ilustra os sinais de controle e erro, enquanto o quadrante inferior esquerdo apresenta a densidade e sua média. O quadrante superior direito, por sua vez, mostra o gráfico de distribuição espacial em duas dimensões (duas características

selecionadas - período e amplitude) e a classificação das amostras de falha através do algoritmo eClass0. Por último, o quadrante inferior direito apresenta a classificação para os mesmas amostras, utilizando, porém, o algoritmo AutoClass. Esse padrão de apresentação é utilizado também nos demais gráficos.

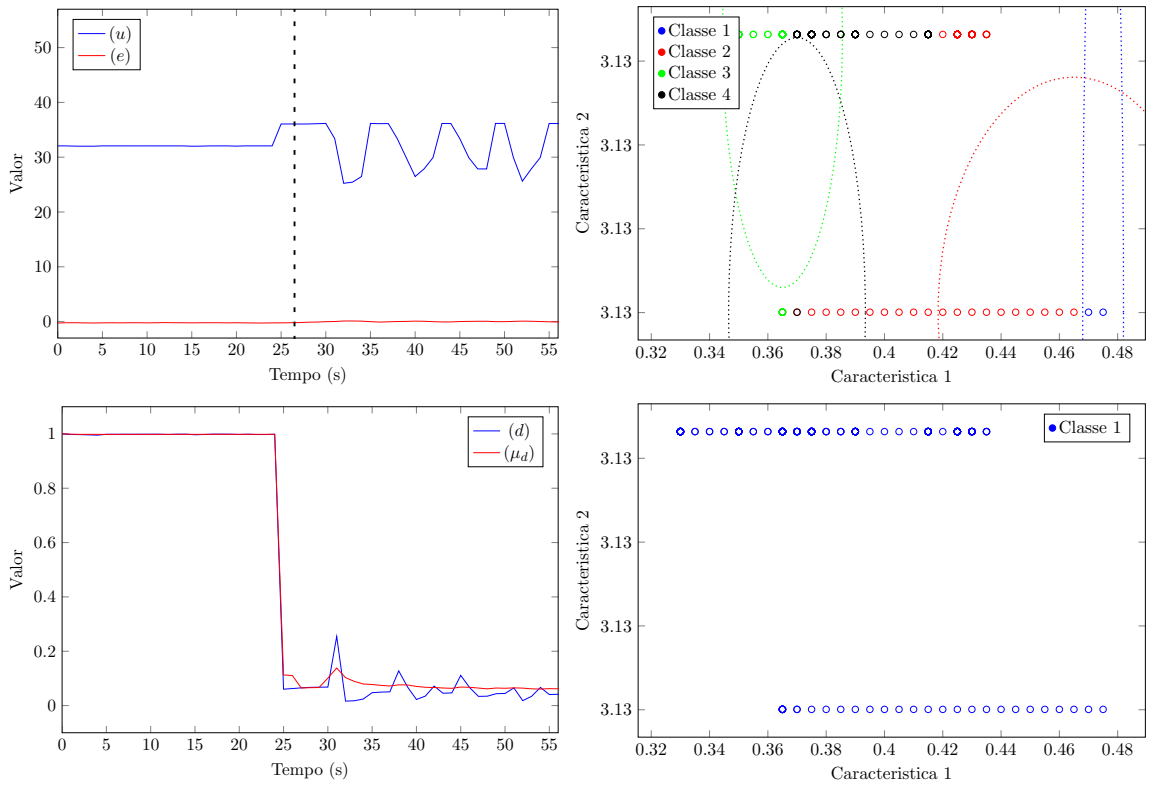

Figura 7.8: Classificação com eClass0 e AutoClass após 300 amostras de falha

Aqui, novamente, utilizamos um período de amostragem de 100ms. Note que a falha F_2 (*off-set* positivo +4%) é detectada por volta do instante $k = 265$. Enquanto, o algoritmo AutoClass cria apenas uma regra de inferência (uma nuvem de dados) no instante $k' = 1$, o algoritmo eClass0 já tem criado quatro regras de inferência (quatro clusters), nos instantes $k' = 1$, $k' = 3$, $k' = 26$ e $k' = 33$, respectivamente. Cada nuvem de dados (para AutoClass) ou cluster (para eClass0) é, automaticamente, nomeada de "Classe i", onde i é o número de classes existentes até o momento mais um. É fácil perceber então, que a classificação de acordo com o algoritmo eClass0, mais uma vez, não representa o real número de falhas identificadas até o momento.

Após o processamento dos primeiros dados de falha, o sistema retorna para o estado "normal" de operação. A próxima falha, F_4 (*off-set* negativo -2%), é, então, detectada por volta do instante de tempo $k = 1.670$, como pode ser visto na Figura 7.9.

No instante $k' = 981$, após a detecção da falha F_4, uma nova nuvem de dados é criada pelo algoritmo AutoClass, automaticamente nomeada de "Classe 2". Note que parte das amostras próximas à segunda nuvem ("Classe 2") foram classificadas como pertencentes à nuvem mais antiga ("Classe 1"). Tal fato dá-se em função de,

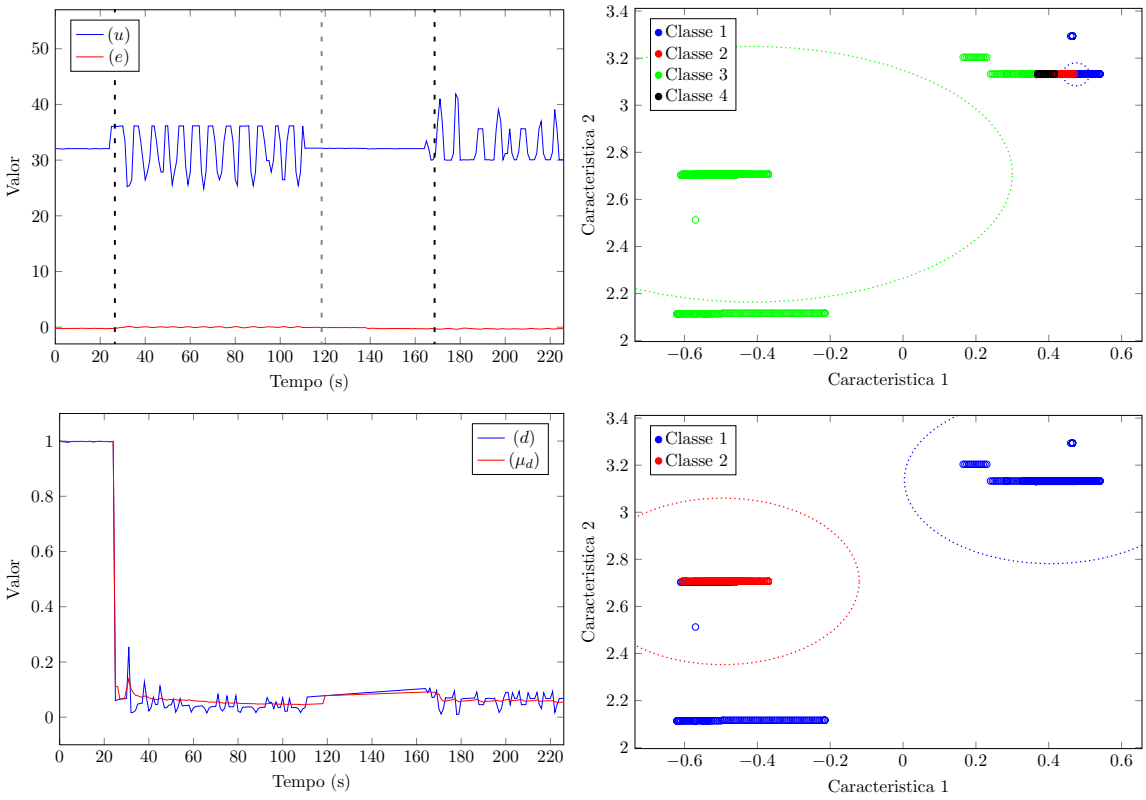

Figura 7.9: Classificação com eClass0 e AutoClass após 1.500 amostras de falha

uma vez que a nuvem mais recente ainda não existia, e a quantidade de pontos e densidade local da nuvem candidata ainda não serem suficientes para a sua criação, tais amostras são classificadas como *outliers* temporários, ou ainda, atribuídas à nuvem mais próxima, mesmo que fora da sua zona de influência. O algoritmo eClass0, por sua vez, não indica a criação de novos clusters até a presente amostra lida.

Avançando ao fim da leitura do stream, que ocorre por volta do instante $k = 5.125$, percebemos que duas novas falhas são detectadas, F_1 (*off-set* positivo +2%) em $k = 3.220$ e F_9 (vazamento de 100%) em 4.260, respectivamente, como ilustrado na Figura 7.10.

No instante $k' = 2.388$, após o após a detecção da falha F_4, uma nova nuvem de dados é criada pelo algoritmo AutoClass, automaticamente nomeada de "Classe 3". Novamente, parte das amostras próximas à terceira nuvem ("Classe 3") foram classificadas como pertencentes à segunda nuvem ("Classe 2"), o que é rapidamente corrigido após a indicação de que o novos novos dados, de fato, representam um novo ponto de operação do sistema. O algoritmo eClass0, por sua vez, não indica a criação de novas regras, trabalhando apenas na atualização da base existente, como pode ser identificado visualmente. É importante mostrar novamente que as falhas F_1 e F_2 pertencem ao mesmo tipo, *off-set* positivo. Sendo assim, o algoritmo AutoClass realizou a classificação correta para a maior parte das amostras. O algoritmo

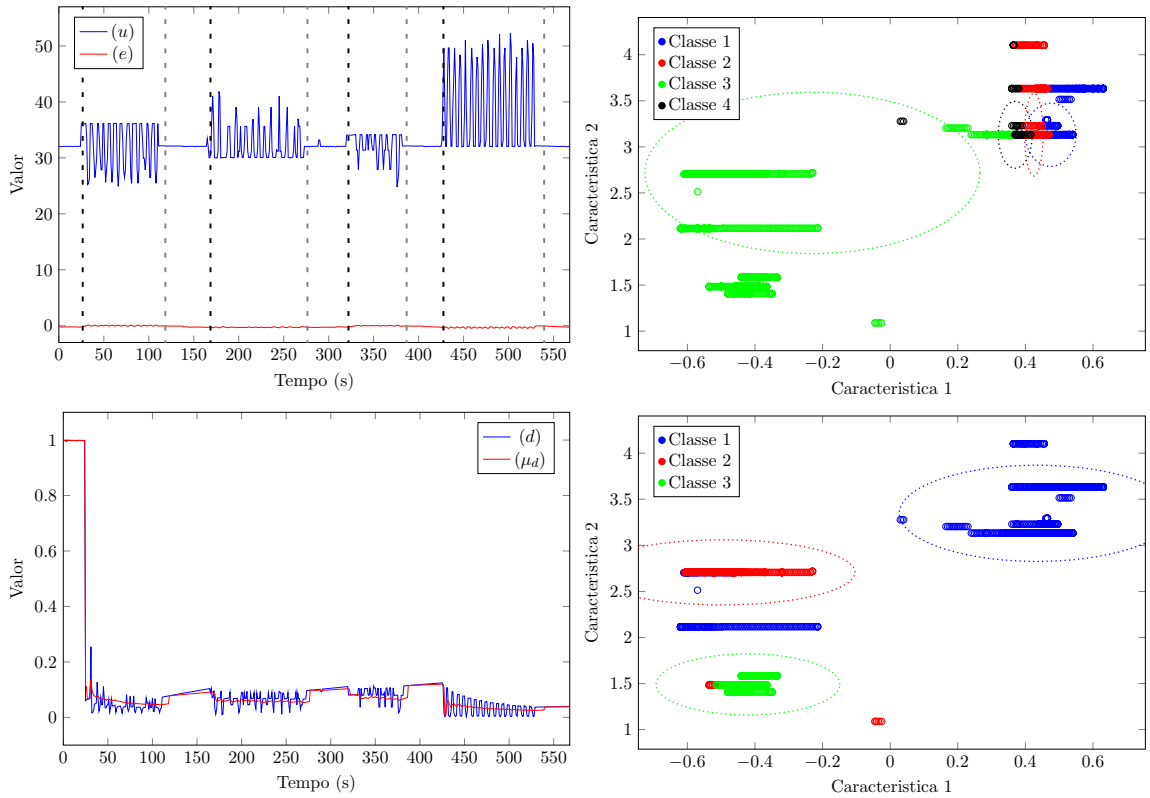

Figura 7.10: Classificação com eClass0 e AutoClass após 3.253 amostras de falha

eClass0, mais uma vez, não apresentou resultados precisos.

A base de regras final gerada pelo algoritmo AutoClass, após a execução de 5.125 amostras de dados, é detalhada abaixo.

$$\mathcal{R}^1 : \textbf{Se } \left(\vec{x} \sim \aleph^1\right) \textbf{ Então } \left(\text{Classe}^1\right)$$
$$\mathcal{R}^2 : \textbf{Se } \left(\vec{x} \sim \aleph^2\right) \textbf{ Então } \left(\text{Classe}^2\right)$$
$$\mathcal{R}^3 : \textbf{Se } \left(\vec{x} \sim \aleph^3\right) \textbf{ Então } \left(\text{Classe}^3\right)$$

onde

$$X^{1*} = [0{,}416;\ 3{,}316] \text{ e } r^1 = [0{,}251;\ 0{,}756]$$
$$X^{2*} = [\text{-}0{,}513;\ 2{,}706] \text{ e } r^2 = [0{,}250;\ 0{,}601]$$
$$X^{3*} = [\text{-}0{,}416;\ 1{,}491] \text{ e } r^3 = [0{,}197;\ 0{,}451]$$

sendo X^{i*} o ponto focal e r^i a zona de influência da nuvem $\aleph^i$.

É importante perceber na Figura 7.10 que, embora o classificador proposto tenha sido capaz de distinguir as falhas F_1 e F_2, que são *off-sets* positivos do atuador, da falha F_4, que é um *off-set* negativo, os seus respectivos pontos ainda estão dispostos de maneira próxima no gráfico de dispersão bi-dimensional, uma vez que ambas as falhas dizem respeito ao atuador. Note também que as falhas F_2 e F_4 também estão próximas uma da outra, embora uma tenha valores negativos para a Característica

1, enquanto a outra tenha valores positivos. A falha F_9, por sua vez, diz respeito a alterações estruturais e, no gráfico em questão, está disposta distante das falhas F_1 e F_2, mas, uma vez que um vazamento é logicamente próximo a uma alteração negativa, F_9 está próximo de F_4.

Tais verificações são importantes para consolidar o argumento de que a abordagem de classificação de falhas proposta é bastante clara ao operador. A estrutura da base de regras, o formato das nuvens e o gráfico de dispersão dão uma ideia real e direta do comportamento do sistema ao usuário, tornando a leitura do estado de operação da planta uma tarefa trivial. Tal característica vai de encontro à estrutura de outras técnicas tradicionais, tais como as redes neurais, redes neuro-*fuzzy*, algoritmos genéticos, uma vez que essas abordagens contam com o fator "caixa-preta", o que dificulta o acompanhamento de todos os passos do processo de inferência pelo usuário, caso este não seja especialista na técnica em questão.

Capítulo 8

Conclusões

Uma inteira e nova abordagem para detecção e diagnóstico de falhas em processos industriais foi apresentada nesta tese. Através de dois estágios independentes e bem definidos, a abordagem proposta é capaz de realizar a detecção e classificação de falhas de diferentes tipos e tamanhos, de uma maneira não-supervisionada e *on-line*, sem a necessidade de conhecimentos prévios avançados a respeito do processo.

Estimativa de densidade recursiva, que foi recentemente introduzida, foi utilizada no primeiro estágio para a detecção de *outliers*/anomalias sobre *streams* de dados. Essa técnica não requer modelos pré-definidos ou parâmetros complexos definidos pelo usuário, como as técnicas tradicionais requerem, e é completamente baseada nos dados.

No estágio de classificação/identificação, uma nova abordagem, chamada AutoClass, foi apresentada nesta tese. AutoClass pode ser utilizado em problemas de classificação, e trabalha com um procedimento de rotulação automática, de maneira similar ao classificador de aprendizagem autônoma eClass0, também recentemente introduzido e utilizado em diversas áreas. AutoClass, diferentemente das abordagens tradicionais, trabalha com o conceito de nuvens de dados, que são estruturas sem formas, limites, centros ou funções paramétricas específicas que as descrevem, sendo modeladas a partir da distribuição real dos dados.

Nesta tese, o sistema de DDF proposto foi aplicado com sucesso a um processo real e um simulado (diferentes plantas de controle de nível), com diversas falhas geradas fisicamente e através de *software*. O estágio de detecção proposto foi comparado à técnica de controle estatístico de processo, e o segundo estágio comparado ao classificador evolutivo eClass0. Os resultados de ambos os estágios demonstraram a superioridade da abordagem proposta no problema de detecção e classificação, bem como o fato de que um agrupamento com estrutura aberta e rotulação automática pode ser gerado de maneira *on-line*, a partir de *streams* de dados, alcançando altas taxas de classificação e utilizando recursos computacionais limitados.

Além do procedimento em dois estágios, composto pelos algoritmos de detecção e classificação aqui propostos, das aplicações experimentais, reais e simuladas, realizadas para validação da proposta, e do estudo comparativo entre as abordagens apresentadas e algumas das técnicas existentes na literatura, esta tese veio a contribuir no sentido de introduzir um procedimento para detecção e diagnóstico de falhas genérico (não orientado a tipos específicos de aplicações), transparente (li-

vre de elementos "caixa-preta"), autônomo (com intervenção mínima do usuário) e de estrutura flexível (adaptável mudanças não previstas no sistema e pontos de operação desconhecidos).

8.1 Publicações

O trabalho de pesquisa que levou ao desenvolvimento desta tese de Doutorado deu origem às seguintes publicações (em ordem cronológica inversa):

- COSTA, Bruno S. J. "Fuzzy Fault Detection and Diagnosis (capítulo de livro)". Em: *Handbook in Computational Intelligence*, editado por Plamen Angelov. World Scientific, USA (a ser publicado em 2015).
- COSTA, Bruno S. J. ; ANGELOV, Plamen P. ; GUEDES, Luiz Affonso. "A New Unsupervised Approach to Fault Detection and Identification". In: *Neural Networks (IJCNN), The 2014 International Joint Conference on*, 6-11 de Julho de 2014, Pequim, China (aceito para publicação).
- COSTA, Bruno S. J. ; ANGELOV, Plamen P. ; GUEDES, Luiz Affonso. "Fully Unsupervised Fault Detection and Identification Based on Recursive Density Estimation and Self-evolving Cloud-based Classifier". *Journal Neurocomputing*, Elsevier, special issue on Data Streams (aceito para publicação).
- COSTA, Bruno S. J. ; ANGELOV, Plamen P. ; GUEDES, Luiz Affonso. "Real-Time Fault Detection Using Recursive Density Estimation". *Journal of Control, Automation and Electrical Systems*, Springer, D.O.I: 10.1007/s40313-014-0128-4.
- COSTA, Bruno S. J. ; ŠKRJANC, Igor ; BLAŽIČ, Sašo ; ANGELOV, Plamen P. "A Practical Implementation of Self-evolving Cloud-based Control of a Pilot Plant". *Cybernetics (CYBCONF), 2013 IEEE International Conference on*, pp.7,12, 13-15 de Junho de 2013, Lausanne, Suíça. [1]
- COSTA, Bruno S. J. ; BEZERRA, Clauber G. ; GUEDES, Luiz Affonso. "A multistage fuzzy controller: Toolbox for industrial applications". In: *2012 IEEE International Conference on Industrial Technology (ICIT 2012)*, p.p. 1142-1147, 19-21 de Março de 2012, Atenas, Grécia.
- COSTA, Bruno S. J. ; BEZERRA, Clauber G. ; GUEDES, Luiz Affonso. "Desenvolvimento de um Software para Automação de Processos Utilizando Lógica Fuzzy". In: *Congresso Brasileiro de Automática (CBA'2012)*, vol. 1, p.p. 4130-4137, 2012, Campina Grande, PB, Brasil.
- COSTA, Bruno S. J. ; BEZERRA, Clauber G. ; GUEDES, Luiz Affonso. "JFuzZ: Uma Ferramenta para Desenvolvimento de Aplicações Fuzzy Industriais". In: *II Congresso Brasileiro de Sistemas Fuzzy*, p.p. 16-17, 2012, Natal, Brasil.
- COSTA, Bruno S. J. ; BEZERRA, Clauber G. ; GUEDES, Luiz Affonso. "Java Fuzzy Logic Toolbox for Industrial Process Control". In: *Congresso Brasileiro de Automática (CBA'2010)*, p.p. 207-214, 2010, Bonito-MS, Brasil.

[1] Ganhador do prêmio de melhor artigo discente da conferência.

8.2 Trabalhos atuais e futuros

O algoritmo AutoClass está sendo generalizado em um procedimento chamado *AutoCloud*, podendo ser, então, utilizado em aplicações diversas, tais como clusterização e processamento de imagens. Testes preliminares já foram realizados para aplicação do algoritmo AutoCloud a problemas de controle, sendo os resultados bastante promissores (Costa et al., 2013). Como trabalhos futuros, estão a inclusão de consequentes de primeira ordem (lineares) e procedimento de extração automática de características, tornando a abordagem ainda mais poderosa e aumentando o seu escopo de aplicação.

Referências Bibliográficas

J. Abonyi. *Fuzzy Model Identification for Control*. Birkhäuser, Boston, USA, 2003.

A. Akhenak, M. Chadli, J. Ragot, & D. Maquin. Design of fuzzy observers for Takagi-Sugeno fuzzy models for fault detection and isolation. In *Proceedings of the 7th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes*, volume 9 of *SAFEPROCESS 2009*, pages 1109–1114, 2009.

P. Angelov. *Evolving Rule-based Models: A Tool for Design of Flexible Adaptive Systems*. Studies in Fuzziness and Soft Computing. Springer Verlag, 2002.

P. Angelov. An approach for fuzzy rule-base adaptation using on-line clustering. *International Journal of Approximate Reasoning*, 35(3):275 – 289, 2004. ISSN 0888-613X.

P. Angelov. *Autonomous Learning Systems: From Data to Knowledge in Real Time*. John Willey and Sons, 2012a.

P. Angelov. Anomalous system state identification, patent gb1208542.9, priority date 15 may 2012, 05 2012b.

P. Angelov & R. Buswell. Identification of evolving fuzzy rule-based models. *Fuzzy Systems, IEEE Transactions on*, 10(5):667–677, 2002. ISSN 1063-6706.

P. Angelov & N. Kasabov. Evolving intelligent systems, eis. *IEEE SMC eNewsLetter*, 15:1–13, 2006.

P. Angelov & R. Yager. Simplified fuzzy rule-based systems using non-parametric antecedents and relative data density. In *Evolving and Adaptive Intelligent Systems (EAIS), 2011 IEEE Workshop on*, pages 62–69, 2011.

P. Angelov & R. Yager. A new type of simplified fuzzy rule-based systems. *International Journal of General Systems*, 41(2):163–185, 2012. ISSN 0308-1079.

P. Angelov, V. Giglio, C. Guardiola, E. Lughofer, & J. M. Luján. An approach to model-based fault detection in industrial measurement systems with application to engine test benches. *Measurement Science and Technology*, 17(7), 2006.

P. Angelov, X. Zhou, & F. Klawonn. Evolving fuzzy rule-based classifiers. In *Computational Intelligence in Image and Signal Processing, 2007. CIISP 2007. IEEE Symposium on*, pages 220–225, April 2007.

P. Angelov, R. Ramezani, & X. Zhou. Autonomous novelty detection and object tracking in video streams using evolving clustering and Takagi-Sugeno type neuro-fuzzy system. In *Neural Networks, 2008. IJCNN 2008. (IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, pages 1456–1463, 2008.

P. Angelov, R. D. Baruah, & J. Andreu. Simpl_eclass: simple potential-free evolving fuzzy rule-based on-line classifiers. In *Proceedings of 2011 IEEE International Conference on Systems, Man and Cybernetics, SMC 2011, Anchorage, Alaska, USA, 7-9 Oct, 2011*, pages 2249–2254. IEEE, 2011.

P. P. Angelov & D. P. Filev. Flexible models with evolving structure. In *Intelligent Systems, 2002. Proceedings. 2002 First International IEEE Symposium*, volume 2, pages 28–33 vol.2, 2002.

P. P. Angelov & D. P. Filev. An approach to online identification of Takagi-Sugeno fuzzy models. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 34(1):484–498, 2004. ISSN 1083-4419.

P. P. Angelov & X. Zhou. Evolving fuzzy-rule-based classifiers from data streams. *Fuzzy Systems, IEEE Transactions on*, 16(6):1462–1475, 2008. ISSN 1063-6706.

H. Bae, S. Kim, J. M. Kim, & K.B. Kim. Development of flexible and adaptable fault detection and diagnosis algorithm for induction motors based on self-organization of feature extraction. In *Proceedings of the 28th Annual German Conference on AI*, KI 2005, pages 134–147, 2005.

R. D. Baruah & P. Angelov. Evolving local means method for clustering of streaming data. In *Fuzzy Systems (FUZZ-IEEE), 2012 IEEE International Conference on*, pages 1–8, 2012.

R. D. Baruah & P. P. Angelov. Online learning and prediction of data streams using dynamically evolving fuzzy approach. In *FUZZ-IEEE*, pages 1–8, 2013.

J. Calado & J. S. da Costa. Fuzzy neural networks applied to fault diagnosis. In Vasile Palade, Lakhmi Jain, & CosminDanut Bocaniala, editors, *Computational Intelligence in Fault Diagnosis*, Advanced Information and Knowledge Processing, pages 305–334. Springer London, 2006. ISBN 978-1-84628-343-7.

G. Canureci, M. Vinatoru, C. Maican, & E. Iancu. Additive faults detection and level control in coupled tanks. *WSEAS Trans. Sys. Ctrl.*, 3(11):907–916, November 2008.

M. F. Carulo. *Desenvolvimento e Implementação de Controlodor Fuzzy Aplicado à Produção de Biodiesel*. PhD thesis, Universidade Estadual de Campinas, Campinas, SP, 2008.

J. Casillas, O. Cordon, F. Herrera, & L. Magdalena. *Interpretability Issues in Fuzzy Modeling*. Springer Verlag, Berlin Heidelberg, 2003.

J.L. Castro & M. Delgado. Fuzzy systems with defuzzification are universal approximators. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 26(1):149–152, Feb 1996. ISSN 1083-4419.

Y. A. Cengel, R. H. Turner, & J. M. Cimbala. *Fundamentals of Thermal-Fluid Sciences*. McGraw Hill,, USA, 4th edition, 2012.

C.J. Chen & R.J. Patton. *Robust Model-Based Fault Diagnosis For Dynamic Systems*. Kluwer International Series on Asian Studies in Computer and Information Science, 3. Kluwer, 1999. ISBN 9780792384113.

Haifeng Chen, Guofei Jiang, & K. Yoshihira. Fault detection in distributed systems by representative subspace mapping. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 4, pages 912–915, 2006.

L.H. Chiang, R.D. Braatz, & E.L. Russell. *Fault Detection and Diagnosis in Industrial Systems*. Advanced Textbooks in Control and Signal Processing. Springer London, 2001. ISBN 9781852333270.

G. E. Cook, J. E. Maxwell, R. J. Barnett, & A. M. Strauss. Statistical process control application to weld process. *Industry Applications, IEEE Transactions on*, 33(2):454–463, 1997. ISSN 0093-9994.

B. Costa, I. Škrjanc, S. Blažič, & P. Angelov. A practical implementation of self-evolving cloud-based control of a pilot plant. In *2013 IEEE International Conference on Cybernetics*, Lausanne, Switzerland, 2013.

B. S. J. Costa, C. G. Bezerra, & L. A. Guedes. Java fuzzy logic toolbox for industrial process control. In *Brazilian Conference on Automatics (CBA)*, Bonito-MS, Brazil, 2010. Brazilian Society for Automatics (SBA).

B. S. J. Costa, C. G. Bezerra, & L. A. Guedes. A multistage fuzzy controller: Toolbox for industrial applications. In *Industrial Technology (ICIT), 2012 IEEE International Conference on*, pages 1142–1147, 2012.

S. Dash, R. Rengaswamy, & V. Venkatasubramanian. Fuzzy-logic based trend classification for fault diagnosis of chemical processes. *Computers & Chemical Engineering*, 27(3):347 – 362, 2003. ISSN 0098-1354.

R. B. M. de Souza. Projeto de reguladores fuzzy Takagi-Sugeno utilizando as condições iniciais da planta. Master's thesis, Universidade Estadual Paulista, UNESP, Ilha Solteira, SP, August 2006.

DeLorenzo. Dl 2314br - Didactic process control pilot plant. Catalog, DeLorenzo Italy, Italy, 2009.

A. L. Dexter & M. Benouarets. Model-based fault diagnosis using fuzzy matching. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 27(5):673–682, Sep 1997. ISSN 1083-4427.

S. Donders. Fault detection and identification for wind turbine systems: a closed-loop analysis. Master's thesis, Faculty of Applied Physics, Systems and Control Engineering, University of Twente, The Netherlands, 2002.

C. Edwards, T. Lombaerts, & H. Smaili. *Fault Tolerant Flight Control: A Benchmark Challenge*. Lecture Notes in Control and Information Sciences. Springer, 2010. ISBN 9783642116896.

D. Efimov, A. Zolghadri, & T. RaÃ¯ssi. Actuator fault detection and compensation under feedback control. *Automatica*, 47(8):1699 – 1705, 2011. ISSN 0005-1098.

P. M. Frank & B. Köppen-Seliger. Fuzzy logic and neural network applications to fault diagnosis. *International Journal of Approximate Reasoning*, 16(1):67 – 88, 1997. ISSN 0888-613X. Fuzzy Logic Applications.

E. Frisk. *Residual Generation for Fault Diagnosis*. PhD thesis, Department of Electrical Engineering, Linköping University, Sweden, 2001.

M.J. Gacto, R. Alcalá, & F. Herrera. Interpretability of linguistic fuzzy rule-based systems: An overview of interpretability measures. *Information Sciences*, 181(20): 4340 – 4360, 2011. ISSN 0020-0255. Special Issue on Interpretable Fuzzy Systems.

J. Gama & P. Kosina. Learning decision rules from data streams. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, volume 2 of *IJCAI'11*, pages 1255–1260, 2011. ISBN 978-1-57735-514-4.

A. S. Glass, P. Gruber, M. Roos, & J. Todtli. Qualitative model-based fault detection in air-handling units. *Control Systems, IEEE*, 15(4):11–22, Aug 1995. ISSN 1066-033X.

P. Gmytrasiewicz, J. A. Hassberger, & J. C. Lee. Fault tree based diagnostics using fuzzy logic. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(11):1115–1119, Nov 1990. ISSN 0162-8828.

B. P. Graham & R. B. Newell. Fuzzy adaptive control of a first-order process. *Fuzzy Sets and Systems*, 31(1):47 – 65, 1989. ISSN 0165-0114.

A. Hossain, Z. A. Choudhury, & S. Suyut. Statistical process control of an industrial process in real time. *Industry Applications, IEEE Transactions on*, 32(2):243–249, 1996. ISSN 0093-9994.

Q. Hu, Z.J. He, Y. Zi, Z.S. Zhang, & Y. Lei. Intelligent fault diagnosis in power plant using empirical mode decomposition, fuzzy feature extraction and support vector machines. *Key Engineering Materials*, 293-294:373–382, 2005.

A. H. F. Insfran, A. P. A. da Silva, & G. L. Torres. Fault diagnosis using fuzzy sets. *Expert Systems with Applications*, 7(4):177–182, 1999.

R. Isermann. Model-based fault-detection and diagnosis - status and applications. *Annual Reviews in Control*, 29(1):71 – 85, 2005. ISSN 1367-5788.

R. Isermann. *Fault-Diagnosis Systems: An Introduction from Fault Detection to Fault Tolerance*. Springer, 2006. ISBN 9783540241126.

J. S. R. Jang & C. T. Sun. Neuro-fuzzy modeling and control. In *Proceedings of the IEEE*, 1995.

M. Kano, T. Sakata, & S. Hasebe. Just-in-time statistical process control for flexible fault management. In *SICE Annual Conference 2010, Proceedings of*, pages 1482–1485, 2010.

S. Katipamula & M. R. Brambley. Methods for fault detection, diagnostics, and prognostics for building systems - a review, part i. *HVAC&R RESEARCH*, 11(1): 3–25, 2005.

E. E. Kerre & M. Nachtegael. *Fuzzy Techniques in Image Processing*. Physica Verlag, Heidelberg New York, 2000.

H. Khalajzadeh, C. Dadkhah, & M. Mansouri. A review on applicability of expert system in designing and control of autonomous cars. In *Fourth International Workshop on Advanced Computational Intelligence*, China, 2011.

D. Kolev, P. Angelov, G. Markarian, M. Suvorov, & S. Lysanov. Arfa: Automated real-time flight data analysis using evolving clustering, classifiers and recursive density estimation. In *Proceedings of the IEEE Symposium Series on Computational Intelligence SSCI-2013*, pages 91–97, Singapore, 2013.

J. Korbicz. *Fault Diagnosis: Models, Artificial Intelligence, Applications*. Engineering online library. U.S. Government Printing Office, 2004. ISBN 9783540407676.

P. Kosina & J. Gama. Handling time changing data with adaptive very fast decision rules. In *Machine Learning and Knowledge Discovery in Databases*, volume 7523 of *Lecture Notes in Computer Science*, pages 827–842. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-33459-7.

R. Kruse, J. Gebhardt, & R. Palm. *Fuzzy Systems in Computer Science*. Verlag Vieweg, Wiesbaden, Germany, 1994.

M. Kulkarni, S. C. Abou, & M. Stachowicz. Fault detection in hydraulic system using fuzzy logic. In *Proceedings of the World Congress on Engineering and Computer Science 2009*, volume 2 of *WCECS'2009*, 2009.

E. G. Laukonen, K. M. Passino, V. Krishnaswami, G. C. Luh, & G. Rizzoni. Fault detection and isolation for an experimental internal combustion engine via fuzzy identification. *Control Systems Technology, IEEE Transactions on*, 3(3):347–355, 1995. ISSN 1063-6536.

C. C. Lee. Fuzzy logic in control systems: fuzzy logic controller - part 1. *IEEE Transactions on Systems, Man and Cybernetics*, 20(2):404 – 418, 1990.

A. Lemos, W. Caminhas, & F. Gomide. Adaptive fault detection and diagnosis using an evolving fuzzy classifier. *Information Sciences*, 220(0):64 – 85, 2013. ISSN 0020-0255. Online Fuzzy Machine Learning and Data Mining.

Chun-Liang Lin & Chun-Te Liu. Failure detection and adaptive compensation for fault tolerable flight control systems. *Industrial Informatics, IEEE Transactions on*, 3(4):322–331, Nov 2007. ISSN 1551-3203.

F. J. Lin, C. H. Lin, & P. H. Shen. Self-constructing fuzzy neural network speed controller for permanent-magnet synchronous motor drive. *IEEE Transactions on Fuzzy Systems*, 9(3):751–759, 2001.

J. Liu, K. W. Lim, W. K. Ho, K. C. Tan, A. Tay, & R. Srinivasan. Using the opc standard for real-time process monitoring and control. *IEEE Software*, 22(6): 54–59, 2005.

T. Liukkonen & A. Tuominen. A case study of spc in circuit board assembly: statistical mounting process control. In *Microelectronics, 2004. 24th International Conference on*, volume 2, pages 445–448 vol.2, 2004.

C. H. Lo, P. T. Chan, Y. K. Wong, A. B. Rad, & K. L. Cheung. Fuzzy-genetic algorithm for automatic fault detection in {HVAC} systems. *Applied Soft Computing*, 7(2):554 – 560, 2007. ISSN 1568-4946.

E. Lughofer. On-line evolving image classifiers and their application to surface inspection. *Image and Vision Computing*, 28(7):1065 – 1079, 2010. ISSN 0262-8856.

E. Lughofer. *Evolving Fuzzy Systems - Methodologies, Advanced Concepts and Applications*. Springer, Berlin Heidelberg, 2011.

E. Lughofer. On-line assurance of interpretability criteria in evolving fuzzy systems - achievements, new concepts and open issues. *Information Sciences*, 251(0):22 – 46, 2013. ISSN 0020-0255.

E. Lughofer & C. Guardiola. On-line fault detection with data-driven evolving fuzzy models. *Control and Intelligent Systems*, 36(4):307–317, 2008.

E. H. Mamdani & S. Assilian. An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies*, 7:1–13, 1975.

A. Marins. Continuous process workbench. Technical manual, DeLorenzo Brazil, Brazil, 2009.

E. B. Martin, A. J. Morris, & J. Zhang. Process performance monitoring using multivariate statistical process control. *Control Theory and Applications, IEE Proceedings* -, 143(2):132–144, 1996. ISSN 1350-2379.

L. Mendonça, J. Sousa, & J. S. da Costa. Fault detection and isolation of industrial processes using optimized fuzzy models. In *Fuzzy Systems, 2005. FUZZ '05. The 14th IEEE International Conference on*, pages 851–856, May 2005.

L. Mendonça, J. Sousa, & J. S. da Costa. Fault detection and isolation of industrial processes using optimized fuzzy models. In V. Palade, L. Jain, & C. D. Bocaniala, editors, *Computational Intelligence in Fault Diagnosis*, Advanced Information and Knowledge Processing, pages 81–104. Springer London, 2006. ISBN 978-1-84628-343-7.

L. F. Mendonça, J. M. C. Sousa, & J. M. G. Sá da Costa. An architecture for fault detection and isolation based on fuzzy methods. *Expert Systems with Applications*, 36(2, Part 1):1092 – 1104, 2009. ISSN 0957-4174.

F. Meneghetti. Mathematical modeling of dynamic systems. Laboratory Keynotes, n. 2, Federal University of Rio Grande do Norte (UFRN), 2004.

S. Mitra & Y. Hayashi. Neuro-fuzzy rule generation: Survey in soft computing framework. *IEEE Transactions on Neural Networks*, 11(3):748–768, 2000.

L. A. Mozelli. Controle fuzzy para sistema Takagi-Sugeno: Condições aprimoradas e aplicações. Master's thesis, Universidade Federal de Minas Gerais, UFMG, Belo Horizonte, MG, October 2008.

Y. L. Murphey, J. Crossman, & Z. Chen. Developments in applied artificial intelligence. In *Proceedings of the 16th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, IEA/AIE 2003, pages 83–92, 2003.

O. Nelles. *Nonlinear System Identification*. Springer, Berlin, Germany, 2001.

S. Oblak, I. Škrjanc, & S. Blažič. Fault detection for nonlinear systems with uncertain parameters based on the interval fuzzy model. *Engineering Applications of Artificial Intelligence*, 20(4):503 – 510, 2007. ISSN 0952-1976.

K. Patan. *Artificial Neural Networks for the Modelling and Fault Diagnosis of Technical Processes*. Lecture Notes in Control and Information Sciences. Springer, 2008. ISBN 9783540798712.

R. J. Patton, R. N. Clark, & P. M. Frank. *Issues of Fault Diagnosis for Dynamic Systems*. Springer, 2000. ISBN 9783540199687.

W. Pedrycz. *Fuzzy control and fuzzy systems*. Electronic & electrical engineering research studies: Control theory and applications studies series. Research Studies Press, 1993. ISBN 9780863801310.

W. Pedrycz & F. Gomide. *Fuzzy Systems Engineering: Toward Human-Centric Computing*. John Wiley & Sons, Hoboken, New Jersey, USA, 2007.

Z. Peng, M. Xiaodong, Y. Zongrun, & Y. Zhaoxiang. An approach of fault diagnosis for system based on fuzzy fault tree. In *MultiMedia and Information Technology, 2008. MMIT '08. International Conference on*, pages 697–700, Dec 2008.

Quanser. Coupled tanks user manual. Technical manual, Quanser, 2004.

S. A. S. A. Rahman, F. A. M. Yusof, & M. Z. A. Bakar. The method review of neuro-fuzzy applications in fault detection and diagnosis system. *International Journal of Engineering & Technology*, 10(3):50–52, june 2010.

R. Ramezani, P. Angelov, & X. Zhou. A fast approach to novelty detection in video streams using recursive density estimation. In *Intelligent Systems, 2008. IS '08. 4th International IEEE Conference*, volume 2, pages 14–2–14–7, 2008.

P. Sadeghi-Tehran, A. B. Cara, P. Angelov, H. Pomares, I. Rojas, & A. Prieto. Self-evolving parameter-free rule-based controller. In *Proceedings of 2012 IEEE World Congress on Computational Intelligence*, Brisbane, Australia, June 2012. IEEE.

A. K. Samantaray & B. O. Bouamama. *Model-based Process Supervision: A Bond Graph Approach*. Springer Publishing Company, Incorporated, 1st edition, 2008. ISBN 1848001584, 9781848001589.

M. H. Schwarz & J. Boercsoek. A survey on ole for process control (opc). In *Proceedings of the 7th Conference on 7th WSEAS International Conference on Applied Computer Science*, pages 186–191, Stevens Point, Wisconsin, USA, 2007. World Scientific and Engineering Academy and Society (WSEAS).

F. Serdio, E. Lughofer, K. Pichler, T. Buchegger, & H. Efendic. Residual-based fault detection using soft computing techniques for condition monitoring at rolling mills. *Information Sciences*, 259:304 – 320, 2014. ISSN 0020-0255.

D. R. C. Silva. *Sistema de Detecção e Isolamento de Falhas em Sistemas Dinâmicos Baseado em Identificação Paramétrica*. Doutorado em Engenharia de Computação, Departamento de Engenharia de Computação e Automação - Universidade Federal do Rio Grande do Norte (UFRN), Brasil, 2008.

I. N. Silva. Avanços em tecnologias de sistemas inteligentes e aplicações. In C. Minussi A. Feltrin, editor, *Tutoriais do XVIII Congresso Brasileiro de Automática*, pages 65–95, Bonito, MS, 2010. Sociedade Brasileira de Automática.

S. Simani & R. J. Patton. Fault diagnosis of an industrial gas turbine prototype using a system identification approach. *Control Engineering Practice*, 16(7):769 – 786, 2008. ISSN 0967-0661.

Silvio Simani, C. Fantuzzi, & R.J. Patton. *Model-based Fault Diagnosis in Dynamic Systems Using Identification Techniques*. Advances in Industrial Control. Springer, 2002. ISBN 9781852336851.

H. Sneider & P. M. Frank. Observer-based supervision and fault detection in robots using nonlinear and fuzzy logic residual evaluation. *Control Systems Technology, IEEE Transactions on*, 4(3):274–282, 1996. ISSN 1063-6536.

M. Suvorov, S. Ivliev, G. Markarian, D. Kolev, D. Zvikhachevskiy, & P. Angelov. Osa: One-class recursive svm algorithm with negative samples for fault detection. In *Artificial Neural Networks and Machine Learning - ICANN 2013*, volume 8131 of *Lecture Notes in Computer Science*, pages 194–207. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-40727-7.

T. Takagi & M. Sugeno. Fuzzy identification of system and its applications to modeling and control. *IEEE Transactions on Systems, Man and Cybernetics*, 15 (1):116–132, 1985a.

T. Takagi & M. Sugeno. Fuzzy identification of system and its applications to modeling and control. *IEEE Transactions on Systems, Man and Cybernetics*, 15 (1):116–132, 1985b.

H. Tanaka, L. T. Fan, F. S. Lai, & K. Toguchi. Fault-tree analysis by fuzzy probability. *Reliability, IEEE Transactions on*, R-32(5):453–457, Dec 1983. ISSN 0018-9529.

V. Venkatasubramanian, R. Rengaswamy, & S. N. Kavuri. A review of process fault detection and diagnosis: Part ii: Qualitative models and search strategies. *Computers & Chemical Engineering*, 27(3):313 – 326, 2003a. ISSN 0098-1354.

V. Venkatasubramanian, R. Rengaswamy, S. N. Kavuri, & K. Yin. A review of process fault detection and diagnosis: Part iii: Process history based methods. *Computers & Chemical Engineering*, 27(3):327 – 346, 2003b. ISSN 0098-1354.

V. Venkatasubramanian, R. Rengaswamy, K. Yin, & S. N. Kavuri. A review of process fault detection and diagnosis: Part i: Quantitative model-based methods. *Computers & Chemical Engineering*, 27(3):293 – 311, 2003c. ISSN 0098-1354.

L. X. Wang. Fuzzy systems are universal approximators. In *IEEE Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1163–1170, San Diego, CA, USA, Mar 1992.

P. Wang & C. Guo. Based on the coal mine's essential safety management system of safety accident cause analysis. *American Journal of Environment, Energy and Power Research*, 1(3):62 – 68, 2013.

M. Witczak. *Fault Diagnosis and Fault-Tolerant Control Strategies for Non-Linear Systems: Analytical and Soft Computing Approaches*. Springer International Publishing, 2013. ISBN 9783319030135.

H. Yang, J. Mathew, & L. Ma. Vibration feature extraction techniques for fault diagnosis of rotating machinery : a literature survey. In *Asia-Pacific Vibration Conference*, pages 801–807, Gold Coast, Australia, 2003.

L. A. Zadeh. Fuzzy sets. *Information and Control 8*, pages 338–353, 1965.

L. A. Zadeh. Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Transactions on Systems, Man and Cybernetics SMC-3*, pages 28–44, 1973.

Y. Zhao, J. Lam, & H. Gao. Fault detection for fuzzy systems with intermittent measurements. *Fuzzy Systems, IEEE Transactions on*, 17(2):398–410, April 2009. ISSN 1063-6706.

Kemin Zhou & Zhang Ren. A new controller architecture for high performance, robust, and fault-tolerant control. *Automatic Control, IEEE Transactions on*, 46 (10):1613–1618, Oct 2001. ISSN 0018-9286.

Shang-Ming Zhou & John Q. Gan. Low-level interpretability and high-level interpretability: a unified view of data-driven interpretable fuzzy system modelling. *Fuzzy Sets and Systems*, 159(23):3091 – 3131, 2008. ISSN 0165-0114.