UNIVERSIDADE FEDERAL DE ITAJUBÁ
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

MODELAGEM DE SISTEMAS MIMO BASEADA EM CONJUNTOS APROXIMADOS

Rubiane Heloisa Oliveira

Itajubá, Dezembro de 2013

UNIVERSIDADE FEDERAL DE ITAJUBÁ

Programa de Pós Graduação em Engenharia Elétrica

Rubiane Heloisa Oliveira

MODELAGEM DE SISTEMAS MIMO BASEADA EM CONJUNTOS APROXIMADOS

Tese submetida ao Programa de Pós-graduação em Engenharia Elétrica como parte dos requisitos para obtenção do Título de Doutor em Ciências em Engenharia Elétrica.

Área de concentração:

Automação e Sistemas Elétricos Industriais.

Orientador:

Prof. Dr. Carlos Alberto Murari Pinheiro.

Dezembro de 2013

Itajubá – MG

Ficha catalográfica elaborada pela Biblioteca Mauá –
Bibliotecária Margareth Ribeiro- CRB_6/1700

O48m

Oliveira, Rubiane Heloisa
Modelagem de Sistemas MIMO baseada em Conjuntos Aproximados / Rubiane Heloisa Oliveira. -- Itajubá, (MG) : [s.n.], 2013.
140 p. : il.

Orientador: Prof. Dr. Carlos Alberto Murari Pinheiro.
Tese (Doutorado) – Universidade Federal de Itajubá.

1. Modelos baseados em regras. 2. Conjuntos Aproximados. 3. Funções não lineares. 4. Sistemas de múltiplas entradas e múltiplas saídas (MIMO). I. Pinheiro, Carlos Alberto Murari, orient. II. Morais, Mabel Scianni, coorient. III. Universidade Federal de Itajubá. IV. Título.

UNIVERSIDADE FEDERAL DE ITAJUBÁ

Programa de Pós Graduação em Engenharia Elétrica

Rubiane Heloisa Oliveira

MODELAGEM DE SISTEMAS MIMO BASEADA EM CONJUNTOS APROXIMADOS

Tese aprovada por banca examinadora em 06 de Dezembro de 2013, conferindo ao autor o título de **Doutor em Ciências em Engenharia Elétrica**.

Banca Examinadora:

Prof. Dr. Carlos Alberto Murari Pinheiro (Orientador) – UNIFEI

Prof. Dr. Ronaldo Rossi – UNESP – FEG

Prof. Dr. Agnaldo José da Rocha Reis – UFOP

Prof. Dr. Antônio Carlos Zambroni de Souza – UNIFEI

Prof. Dr. Otávio Augusto Salgado Carpinteiro – UNIFEI

Prof. Dr. Luis Henrique de Carvalho Ferreira – UNIFEI

Itajubá

2013

A Deus,

aos meus adoráveis pais, Pedro e Cida, aos queridos irmãos Elvis e Luciana, as sobrinhas Lívia e Júlia, a cunhada Karina, por me amarem e apoiarem.

AGRADECIMENTO

Agradeço a Deus, por ter me protegido ao longo dessa árdua caminhada, pela saúde, paciência, persistência e perseverança, sem as quais não teria conseguido e por ter me proporcionado a família maravilhosa que tenho.

A minha querida família (minha mãe Cida, pai Pedro, minha irmã Luciana, minhas sobrinhas amadas Lívia e Júlia, meu irmão Elvis e sua esposa Karina) pelo apoio nas horas difíceis da minha vida, estando sempre ao meu lado, apoiando as minhas decisões e algumas vezes me ajudando a tomá-las, dando amor, carinho, compreensão nas muitas vezes que estava com os nervos à flor da pele, sempre me acalmando, com palavras de ânimo e coragem.

Ao professor Dr. Carlos Alberto Murari Pinheiro, que me aceitou como sua orientada, sempre me orientando de maneira clara e objetiva, pela compreensão de alguns problemas que tive ao longo desses quatro anos e incentivo à realização desse trabalho.

Aos amigos Paula dos Santos e Eduardo Moreira Vicente, pelo apoio, amizade e ajuda nas horas difíceis.

A CAPES pelo apoio financeiro.

RESUMO

A modelagem de sistemas dinâmicos é de grande importância em vários campos das ciências e engenharias. Os métodos clássicos de modelagem de sistemas são baseados em modelos matemáticos acurados. Entretanto, para sistemas complexos com características não lineares e/ou parâmetros variantes no tempo, a obtenção dos modelos correspondentes não é uma tarefa trivial. Os modelos baseados em regras têm uma característica importante em relação aos modelos gerados pelos métodos estatísticos clássicos e os modelos gerados por intermédio de redes neurais artificiais. Essa característica consiste na facilidade com que o conhecimento embutido em cada modelo pode ser interpretado, visto que o conhecimento está todo abstraído em regras cuja sintaxe é muito próxima da linguagem humana. Essa característica confere a esses modelos a capacidade de serem facilmente inseridos em sistemas computacionais gerais. Diferentemente dos métodos de análise de dados convencionais, que frequentemente utilizam procedimentos estatísticos, abordagens com conjuntos aproximados são baseadas em técnicas de mineração de dados visando à extração de conhecimentos (Tay e Shen, 2002). A Teoria dos Conjuntos Aproximados (*Rough Sets*) foi introduzida por Pawlak com dois objetivos principais: revelar estruturas pertinentes em conjuntos de dados e classificar objetos (Pawlak, 1982). Através de uma revisão bibliográfica, verificou-se que não é comum trabalhos sobre conjuntos aproximados abordando questões relacionadas com a modelagem de sistemas estáticos ou dinâmicos que utilizam variáveis contínuas ou amostradas, principalmente em relação a sistemas lineares e não lineares, em especial para sistemas com múltiplas variáveis de entrada e de saída (sistemas MIMO) (Pinheiro 2009, Pinheiro et al., 2010_a). Então, neste trabalho será apresentada uma proposta para construção de modelos baseados em regras de sistema MIMO. Para a validação da modelagem, serão apresentados exemplos de modelos não lineares com duas variáveis de entrada e duas de saída, de um modelo discreto e um modelo contínuo de um sistema de nível real (tanques acoplados). Para fins de comparação com a modelagem proposta, serão utilizados modelos *fuzzy Takagi-Sugeno* com estruturas do tipo *ANFIS* (*Adaptative Network Based Fuzzy Inference System*) (Jang 1993).

Palavras-Chaves: Modelos baseados em regras; conjuntos aproximados; funções não lineares; sistemas de múltiplas entradas e múltiplas saídas; modelos não lineares.

ABSTRACT

The modeling of dynamic systems is of great importance in several fields of science and engineering. The classical methods of system modeling are based on accurate mathematical models. However, for complex systems with non-linear and / or time-varying parameters, obtaining the corresponding models is not a trivial task. The rule-based models have an important feature in relation to models generated by the classical statistical methods and models generated by means of artificial neural networks. This feature is the ease through which knowledge embedded in each model can be interpreted, since knowledge is abstracted in rules whose syntax is very close to human language. This feature gives these models the ability to be easily inserted into computer systems. Unlike the methods of conventional data analysis, often using statistical mechanisms, approaches to rough sets are based on data-mining techniques in order to obtain knowledge (Tay and Shen, 2002). The rough set theory was introduced by Pawlak with two main objectives: to reveal the relevant structures in data sets and classify objects (Pawlak, 1982). Through a literature review, it was found that is not common rough sets work on issues related to the modeling of static or dynamic systems that use continuous or sampled variables, mainly in relation to linear and nonlinear systems, especially for multi input and multi output systems (MIMO systems) (Pinheiro 2009, Pinheiro et al., 2010). So in this work a proposal to build rule based model of MIMO Systems will be presented. To validate the modeling, examples of nonlinear models with two input and two output variables of a discrete model and a continuous model of a real system level (coupled tanks.) will be presented. For purposes of comparison with modeling proposal will be used Takagi-Sugeno fuzzy models with structures of type ANFIS (Adaptive Network based fuzzy Inference System) (Jang 1993)

Key-Words: rule-based models; rough sets; nonlinear functions; multi-input multi-output system; nonlinear models.

Índice

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DOS PRINCIPAIS SÍMBOLOS E ABREVIAÇÕES

ANFIS	*Adaptive Network Based Fuzzy Inference System*
DENFIS	*Dynamic Evolving Neural-Fuzzy Inference System*
FIS	*Sistema de Inferência Fuzzy*
FSOM	*Fuzzy Self-Organization Map*
IA	*Inteligência Artificial*
MIMO	*Multi Input Multi Output*
NARMAX	*Nonlinear AutoRegressive Moving Average with exogenous input*
NARX	*Nonlinear AutoRegressive exogenous model*
NEFClass	*Neuro Fuzzy Classification*
NEFCON	*Neuro-Fuzzy control*
SE	*Sistemas Especialistas*
SISO	*Single Input Single Output*
TCA	*Teoria dos Conjuntos Aproximados*
B_*	*Aproximação inferior*
B^*	*Aproximação superior*
$M_D(B)$	*Matriz de discernibilidade*
$m_D(i,j)$	*Elementos da matriz de discernibilidade*
$q_{i_1}(t)$ *e* $q_{i_2}(t)$	*Fluxos de entrada de um determinado fluido*
v_n	*Valor do consequente da regra* n
$\alpha_B(B_*(O))$	*Coeficiente da qualidade da aproximação inferior*
$\alpha_B(B^*(O))$	*Coeficiente da qualidade da aproximação superior*
$\alpha_B(O)$	*Coeficiente de imprecisão da qualidade da aproximação em relação aos elementos do conjunto* O

$\mu_A(x)$	*Função de pertinência*
T	*Atraso de tempo*
$F(B)$	*Função de discernibilidade*
RF	*Região de fronteira*
$\mu(n)$	*Grau de ativação da regra n.*
MLP	*Multi Layer Perceptron*
KDD	*Knowledge Discovery Data Bases*
SI	*Sistema de Informação*
SOM	*Mapas auto-organizável de Kohonen (Self Organizing Maps)*
IMC	*Internal Model Control*
MIT	*Motor de Indução Trifásico*
SQL	*Structured Query Language*
GRBDE	*Gerador de Regras em Banco de Dados Especialistas*
MCSA	*Motor Current Signature Analysis*

CAPÍTULO 1

1 INTRODUÇÃO

1.1 Introdução Geral

A modelagem de sistemas dinâmicos é de grande importância em vários campos das ciências e engenharias. A modelagem de sistemas pode ser classificada em duas vertentes de modelos: os lineares e os não lineares. Os métodos clássicos de modelagem de sistemas são baseados em modelos matemáticos acurados. Entretanto, para sistemas complexos com características não lineares e/ou parâmetros variáveis no tempo, a obtenção dos modelos correspondentes não é trivial (Aguirre et al., 2000). As dificuldades geralmente aumentam em relação a sistemas com múltiplas entradas e saídas (*MIMO – Multi Input Multi Output*) em relação aos procedimentos para sistemas com entradas e saídas singelas (*SISO – Single Input Single Output*).

Para sistemas com características lineares existem métodos clássicos que apresentam estruturas e procedimentos de identificação bem conhecidos. Para sistemas com características não lineares existem diferentes abordagens segundo as aplicações e os métodos utilizados. Entre os modelos não lineares conhecidos têm-se as estruturas NARX, NARMAX, redes neurais artificiais, sistemas fuzzy etc.

Na prática, as informações de dados de sistemas reais são freqüentemente incertas, imprecisas ou incompletas. Diversas metodologias foram desenvolvidas para tratar tais condições, dentre elas a Teoria dos Conjuntos Difusos (Zadeh, 1965), Teoria de Dempster-Shafer (Dempster, 1967), (Shafer, 1976), e a Teoria dos Conjuntos Aproximados (Pawlak, 1982).

A utilização de técnicas de inteligência artificial (redes neurais, lógica fuzzy, algoritmos evolutivos, entre outras) tem se mostrado promissora na modelagem de sistemas com características não lineares. É conhecido que lógica difusa e redes neurais artificiais são alternativas na obtenção de modelos de sistemas complexos, sem a necessidade de detalhes prévios dos processos estudados.

Os modelos baseados em regras desempenham um papel fundamental na modelagem de sistemas complexos com características não lineares e/ou parâmetros variáveis no tempo. Em geral as regras encapsulam relações entre as variáveis dos modelos e fornecem mecanismos para ligar as representações das informações dos mesmos com seus procedimentos computacionais. Por exemplo, muitos modelos baseados em regras podem aproximar uniformemente funções contínuas com algum grau de precisão em conjuntos fechados e limitados, onde uma questão importante diz respeito a procedimentos de estimação de valores dentro dos conjuntos considerados. Alguns métodos são particularmente úteis para reduzir a complexidade dos modelos (Pedrycz e Gomide, 2007).

Existem dois procedimentos principais para a construção de modelos baseados em regra; os baseados no conhecimento de especialistas e os orientados por dados. No primeiro caso supõe-se que um especialista pode fornecer conhecimento sobre o problema considerado. Um especialista pode quantificar um conhecimento relacionando às variáveis de um problema de forma a associá-las a um conjunto de regras que modelam as informações do processo em questão. No entanto, em muitas aplicações, as informações necessárias para desenvolver as regras podem não estar disponíveis de forma direta, e os seres humanos podem ser incapazes de extrair todo o conhecimento relevante de uma grande quantidade de dados. Nessas circunstâncias, procedimentos computacionais podem extrair os conhecimentos necessários e codificá-los na forma de regras, o que constitui o segundo método descrito. O modelo resultante captura a estrutura existente dos próprios dados.

A conceituação sobre conjuntos aproximados (*Rough Sets*) foi introduzida por Pawlak com dois objetivos principais: revelar estruturas pertinentes em conjuntos de dados e classificar objetos (Pawlak, 1982). Diferentemente dos métodos convencionais de análise de dados, que frequentemente utilizam mecanismos estatísticos, a abordagem de conjuntos aproximados é baseada em técnicas de mineração de dados para extrair conhecimentos (Tay e Shen, 2002). Essa teoria foi desenvolvida visando à manipulação de incertezas e imprecisões em conjuntos de dados, contexto inerente em muitos problemas práticos. A teoria fornece meios sistemáticos para eliminar informações irrelevantes através de um processo de redução dos chamados redutos que, conforme será descrito no Capítulo 3, são conjuntos de atributos capazes de manter as mesmas propriedades da representação de conhecimentos quando se utiliza todos os atributos originais de um determinado sistema de informação. O conceito principal envolvido diz respeito à relação conhecida como indiscernibilidade (Pawlak e Skowron, 2007). Através de relações de indiscernibilidade definem-se atributos redundantes

ou supérfluos em bancos de dados, sistemas de informações etc. permitindo a redução de dados em conjuntos mais consistentes (Sakai e Nakata, 2006, Sankar e Mitra, 2002). Uma das principais vantagens dessa teoria é que ela não necessita de informações preliminares ou adicionais sobre os dados que serão manipulados, tais como distribuição de probabilidade, intervalos de crença, valores de possibilidades (Pawlak, 1991).

Os modelos baseados em regras apresentam uma característica importante em relação aos modelos gerados por técnicas via redes neurais artificiais, por exemplo. Para modelos com redes tipo MLP (*Multilayer Perceptron*), os pesos que unem os neurônios de uma camada com os neurônios de outra (Haykin, 2000) são de difícil interpretação em relação a eventuais conhecimentos explícitos. Já os modelos baseados em regras representam conhecimentos cuja sintaxe é passível de codificação em linguagem humana. Essa característica confere a esses modelos a capacidade de serem facilmente inseridos em sistemas computacionais em geral.

1.2 Motivação

Através da revisão bibliográfica apresentada no Capítulo 2, verificou-se que trabalhos sobre conjuntos aproximados abordando questões relacionadas a modelos baseados em regras de sistemas estáticos ou dinâmicos que utilizam variáveis contínuas ou amostradas não era comum (Pinheiro et al. 2010_a, 2010_b), principalmente em relação a sistemas lineares e não lineares em especial sistemas com múltiplas entradas e saídas. Os poucos trabalhos abordam basicamente variáveis puramente binárias ou simbólicas (Ziarko e Katzberg, 1993, Kusiak e Shah, 2006), e representações de sistemas com entrada e saída singelas (*SISO – Single Input Single Output*). Isso motivou o desenvolvimento desse trabalho para propor a construção de modelos baseados em regras para sistemas MIMO empregando conceitos da Teoria dos Conjuntos Aproximados.

1.3 Objetivo

O objetivo é a obtenção de modelos (estáticos ou dinâmicos) referentes a sistemas lineares e não lineares para sistemas MIMO, utilizando conceitos de conjuntos aproximados.

O método proposto consiste em agrupar os dados relativos à modelagem de sistemas MIMO em sistemas de informações correspondentes, onde os mesmos englobam eventuais acoplamentos nas variáveis intrínsecas a estes modelos.

A fim de validar e avaliar a proposta de metodologia utilizada nesse trabalho, serão apresentados um exemplo de um modelo discreto de um sistema não linear com duas variáveis de entrada e duas variáveis de saída (Wang et al., 2009) e um exemplo de um modelo não linear contínuo também com duas variáveis de entrada e duas de saída de um sistema de nível real (tanques acoplados) de uma planta pertencente ao laboratório de Controle de Processos da Universidade Federal de Itajubá.

Para gerar modelos baseados em regras, duas abordagens podem ser consideradas. A primeira baseada na aplicação de conceitos relacionados aos conjuntos aproximados. E a segunda baseada em conjuntos difusos, que será usada para fins de comparação com a metodologia proposta. Serão utilizados modelos *fuzzy Takagi-Sugeno* que terão estruturas do tipo *ANFIS* (*Adaptative Network Based Fuzzy Inference System*) (Jang 1993).

1.4 Organização

Os próximos capítulos desta tese estão organizados da seguinte maneira:

Capítulo 2 - Revisão Bibliográfica: Nesse capítulo é apresentada uma revisão bibliográfica sobre os principais temas pesquisados para o desenvolvimento deste trabalho.

Capítulo 3 - Conceituação sobre Conjuntos Aproximados: O objetivo deste capítulo é apresentar a conceituação sobre conjuntos aproximados, seus conceitos básicos e fundamentos, fornecendo os subsídios necessários para a compreensão da aplicação dos mesmos na proposição deste trabalho.

Capítulo 4 - Revisão sobre Conceitos de Modelos Fuzzy: O objetivo deste capítulo é apresentar conceitos básicos sobre modelos *fuzzy*. Esses conceitos serão importantes para o entendimento da obtenção dos modelos *fuzzy* que serão usados para realizar comparações com os modelos propostos neste trabalho que utilizam conceitos de conjuntos aproximados.

Capítulo 5 – Proposta de Modelagem Aproximada de Sistemas MIMO: O objetivo deste capítulo é apresentar uma proposta para construção de modelos baseados em regras para sistemas MIMO que empregam conceitos de conjuntos aproximados.

Capítulo 6 - Experimentos: O objetivo deste capítulo é apresentar estudos de casos para a proposta feita no capítulo anterior, exemplificando modelos baseados em regras de sistemas dinâmicos MIMO, incluindo o exemplo de um processo real. Os resultados obtidos e os processamentos correspondentes serão confrontados com dados resultantes de modelos *fuzzy*.

Capítulo 7 - Conclusões: Neste capítulo são apresentadas as conclusões e os desenvolvimentos futuros sugeridos para continuação deste trabalho.

CAPÍTULO 2

2 REVISÃO BIBLIOGRÁFICA

O objetivo deste capítulo é apresentar uma resenha bibliográfica sobre os temas principais pesquisados neste trabalho.

2.1 Introdução

Nos últimos anos tem havido um rápido crescimento de trabalhos relacionados a Teoria dos Conjuntos Aproximados (TCA) e suas aplicações. Uma grande quantidade de artigos, dissertações de mestrado, teses de doutorado e tópicos relacionados a conjuntos aproximados (*rough sets*) foram publicados em uma variedade de periódicos, e conferências internacionais. A fundamentação matemática dessa teoria, que será apresentada com maiores detalhes no Capítulo 3, permite que padrões ocultos nas bases de dados sejam encontrados (Tay e Shen, 2002). Essa característica tem grande utilidade em problemas relacionados às áreas da inteligência artificial e das ciências cognitivas, especialmente em reconhecimento de padrões, classificação de informações, mineração de dados, sistemas de decisão e sistemas especialistas (Ilczuk e Wakulicz, 2007). Previsão de falhas em processos industriais (Kusiak e Shah, 2006), processamento de sinais, agrupamento de dados, aplicações em finanças, química, computação, economia, engenharia elétrica, medicina, biologia molecular, neurologia, robótica e ciências sociais estão entre outras áreas de aplicação (Pawlak e Skowron, 2007).

2.2 Resenha Bibliográfica

Ao propor a conceituação sobre conjuntos aproximados, Pawlak (1982) citou algumas vantagens da sua utilização e exemplificou uma aplicação prática a partir de uma análise de um banco de dados contendo atributos que caracterizam diagnósticos médicos de pacientes com sintomas de gripe. Também foram realizadas algumas comparações com outros métodos utilizados para extrair informações de bancos de dados.

O trabalho de Pawlak e Munakata (1996) cita algumas referências que abordam considerações sobre aplicações da teoria dos conjuntos aproximados em sistemas de controle,

argumentando que esta área tem grande potencial para futuros desenvolvimentos, tanto teórico como prático. O artigo indica que conceitos sobre conjuntos aproximados podem ser usados diretamente (dependendo da aplicação), ou agregados com outras técnicas que utilizam, por exemplo, lógica difusa, redes neurais etc. No caso de processamento simbólico das variáveis, as informações de comando poderiam ser traduzidas em dados numéricos que seriam ponderados por técnicas como "centro de área". O artigo conclui mencionando que muitas questões referentes às aplicações de conjuntos aproximados em sistemas de controle estão em aberto, e dependem de novos desenvolvimentos, testes etc.

Cai e Gong (2002) abordaram a questão sobre a aquisição de conhecimento, que é um problema muito importante em muitos campos, tais como sistemas especialistas, sistema de reconhecimento de padrões, sistemas de comunicações etc. O propósito de aquisição de conhecimento é extrair algumas informações sobre a distribuição dos dados brutos relacionados a uma determinada aplicação. Nesse contexto, os autores verificaram que muitas abordagens, como por exemplo, algoritmos genéticos, redes neurais artificiais, lógica fuzzy e a TCA foram propostas para resolver problemas desta natureza. O método mais usual tem sido a aplicação de sistemas *neuro-fuzzy*, que combina a teoria da lógica *fuzzy* e das redes neurais artificiais. No entanto, existem algumas dificuldades, dependendo da aplicação. Uma delas é o aumento do número de regras de modelagem conforme a dimensão dos dados relacionados. Abordagens baseadas na TCA vêm sendo empregadas com sucesso nesse contexto. Esta teoria permite a caracterização de conjuntos de objetos em termos de valores de atributos, cujas dependências (total ou parcial) possibilitam a redução de informações supérfluas.

Como exemplo de aplicação dos conceitos de conjuntos aproximados em previsões, pode-se citar o trabalho de Shen e Loh (2003). Estes autores analisaram o mercado de ações e determinaram se para uma determinada ação, esta deveria ser vendida, comprada ou retida. A fim de realizar tal procedimento, um sistema híbrido composto por uma rede neural *SOM* (*Self Organizing Map*) juntamente com conceitos de conjuntos aproximados foi proposto, chamado de RoughSOM. O sistema de informação do trabalho era composto por sete atributos de decisão relacionados a indicadores financeiros relevantes ao problema. O procedimento foi dividido em três etapas. Na primeira, o sistema de informação foi reconstruído por meio de uma categorização dos dados realizado pela *SOM*. Na segunda etapa, os dados gerados na primeira etapa eram discretizados. Na terceira e última etapa, aplicou-se conceitos da teoria de conjuntos aproximados para construção dos redutos, sendo escolhido aquele que apresentava

menor número de atributos. No caso de empate no número de atributos, escolhia-se aquele que apresentava melhor qualidade de aproximação. Foram realizados estudos com exemplos de base de dados de aprendizagem de máquina para verificar a eficácia do método desenvolvido RoughSOM em relação ao método dos conjuntos aproximados originais, e verificou-se que o método proposto apresentou maior precisão em torno de 2,5% em relação a aplicação usando o método dos conjuntos aproximados originais. O melhor modelo gerado era composto por 1044 regras e conseguiu classificar corretamente 58% das informações.

Pessoa (2004) apresentou um sistema híbrido baseado na teoria dos conjuntos aproximados e nas redes neurais artificiais na previsão climática, com o objetivo de estimar o comportamento médio atmosférico sazonal com um alcance temporal de uma a três estações. A TCA foi usada com o propósito de redução de variáveis para a realização de previsão climática utilizando redes neurais artificiais, de modo a diminuir o esforço computacional e manter os erros em níveis aceitáveis na previsão climática. As redes neurais foram utilizadas para mapear dados de um período de dezoito anos de uma região da América do Sul, o comportamento sazonal das variáveis de precipitação e temperatura, objetivando realizar estimativas para três anos consecutivos. Os resultados dos experimentos para previsão climática de temperatura e de precipitação mostrou-se eficaz.

No trabalho de Yun e Yuanbin (2004) verificou-se que usando a teoria dos conjuntos aproximados, o conhecimento podia ser extraído a partir dos dados de um inversor de frequência que acionava um motor de corrente alternada de uma bomba de recalque de um reservatório de água. Para conseguir a modelagem do sistema, primeiramente foram coletados conjuntos de dados de entrada e saída. Depois um modelo baseado em regras foi obtido de acordo com um algoritmo baseado em conjuntos aproximados. A validação do modelo foi verificada por comparação com dados do sistema real.

Vieira (2005) propôs um sistema híbrido que utiliza os conceitos de modelo relacional aproximado e de um aproximado *fuzzy* combinados com um método simbólico de aprendizado para viabilizar a extração de conhecimento.

Cerchiari (2006) propôs uma metodologia baseada em inteligência artificial para estimar a curva de demanda de consumidores de baixa tensão em uma empresa de energia elétrica, onde foi utilizado estruturas de Mapas Auto-Organizávies (*SOM*) e de conjuntos aproximados. Uma estrutura SOM foi utilizada para reconhecimento de padrões de comportamentos semelhantes nas curvas de demanda dos consumidores, agrupando-as e construindo curvas típicas para sua representação. A utilização da técnica dos conjuntos

aproximados foi no sentido de extrair conhecimento do banco de dados de consumidores, visando à obtenção de um conjunto de regras que possibilitassem de forma automática classificar um consumidor qualquer a uma das curvas típicas obtidas anteriormente. Diante dos resultados apresentados ficou demonstrado que a metodologia proposta é aplicável para esse tipo de problema, estando apta a se incorporar aos sistemas computacionais das distribuidoras de energia elétrica como mais um instrumento de suporte à decisão de investimento e, também, para análise de consumo de energia elétrica.

Sassi (2006) propôs o desenvolvimento, aplicação e análise de uma arquitetura híbrida formada pela combinação da TCA com uma arquitetura de rede SOM para descoberta de conhecimento. O objetivo foi verificar o desempenho da arquitetura proposta na geração de agrupamentos (*clusters*) em bases de dados. Na arquitetura híbrida, a função dos conjuntos aproximados é de redução dos atributos relacionados a informações do sistema, e a função da rede SOM é a de gerar agrupamentos de dados. Com base nos experimentos realizados foi possível concluir que a arquitetura híbrida teve um desempenho superior ao de uma rede SOM convencional, e que a aplicação da arquitetura híbrida pode ser vantajosa em diversas áreas relacionadas a descoberta de conhecimento (*Knowledge Discovery Data Bases*).

Herbert e Yao (2009) aplicaram a teoria de conjuntos aproximados na análise das ações de mercado da Nova Zelândia. A série temporal usada contemplava observações que começavam em 31 de julho de 1991 e terminavam em 27 de abril de 2007. Os dados disponíveis eram referentes ao preço de abertura e fechamento, e os valores do maior e do menor preço alcançados pela ação ao longo do dia. Para cada uma das observações, adicionou-se um atributo de decisão cuja função era verificar se a ação deveria ser vendida, comprada ou se nada deveria ser feito. No total, dispunha-se de 1665 exemplos para construir um modelo e 555 exemplos para validação. Após aplicação da técnica de conjuntos aproximados, um total de dezoito redutos foi encontrado. Desses redutos, selecionou-se aquele com menor número de atributos. Com um total de dez regras, o modelo de previsão resultante foi capaz de atingir uma precisão de aproximadamente 67% das previsões.

Liuyang et at. (2009) utilizaram a TCA no pré-processamento de dados de redes neurais artificiais aplicadas no diagnóstico de falhas de sistemas industriais. Primeiramente uma tabela de dados de decisão de falhas é formada e os dados são discretizados usando um método de agrupamento híbrido. O objetivo principal era remover informações redundantes e buscar tabelas de decisão reduzidas, visando à obtenção de um subconjunto mínimo de falhas. Depois, uma rede neural era treinada com um algoritmo de retro-propagação (*back-*

propagation). O método reduziu a taxa de falsos alarmes, realizando diagnósticos de forma eficaz, podendo detectar falhas compostas e melhorando a robustez do sistema.

Dun et al. (2010) apresentaram um estudo revisando vinte anos da história dos conjuntos aproximados, incluindo a história do seu desenvolvimento, as teorias do método de redução dos atributos e suas aplicações. Mostraram uma revisão das aplicações da teoria dos conjuntos aproximados em diversas áreas ao longo do tempo, tais como: área médica; processo de inspeção de produtos; classificação de documentos; gestão ambiental; seleção de e-mail, pesquisa em campos de petróleo entre outros. O objetivo do artigo era fazer uma revisão geral do assunto e tornar mais claro os conceitos dessa teoria.

Lotfabadi e Moghadam (2010) verificaram que uma das maneiras para diminuir as características redundantes e inválidas de conjuntos de dados em aplicações de mineração de dados, consiste em utilizar métodos de análise dos componentes principais dos conjuntos aproximados e de conjuntos difusos (*fuzzy sets*). No trabalho esses métodos de redução de conjuntos de dados foram comparados com outras técnicas.

Para concluir parte da resenha bibliográfica relativa a conjuntos aproximados, serão citadas a seguir algumas dissertações, teses defendidas na UNIFEI que utilizaram conceitos da TCA em seus desenvolvimentos.

Rossi (2000) propôs um classificador hierárquico sistêmico para redes elétricas de alta tensão com o objetivo de determinar o estado operativo do sistema, auxiliar o processo de monitoração de redes elétricas de médio e grande porte, onde o número de pontos a serem observados é relativamente grande. Também apresentou uma metodologia genérica para a extração do conhecimento para bases de dados gerais, e uma estratégia para determinar o ponto desejado de operação de um dado sistema elétrico a qual suportará a estrutura do classificador hierárquico proposto. Para a validação das ideias propostas no classificador, foram coletados dados reais de um sistema elétrico, tanto em número de atributos quanto em número de exemplos, que foram submetidos à metodologia proposta e verificada a sua eficácia.

Carvalho (2000) apresentou uma aproximação que usa a TCA para reduzir o tamanho de sistemas de banco de dados mantendo somente as informações essenciais para o processo. Uma ferramenta geradora de regras em banco de dados relacionais foi desenvolvida chamada "Gerador de regras em banco de dados especialistas (GRBDE)", usando os recursos da linguagem SQL. Os principais benefícios desse programa computacional foram: auxiliar na

tomada de decisão de usuários de banco de dados, mesmo diante de informações incompletas, aumentando a disponibilidade e o desempenho do usuário na execução de suas tarefas; viabilizar os processos de automação e viabilizar a análise de grande volume de informações, otimizando a tomada de decisão.

Henriques (2001) desenvolveu um classificador de sinal de voz baseado na TCA que permitiu o tratamento de aspectos como redundância, irrelevância e incerteza encontradas nas informações extraídas de sinais de voz. A teoria foi aplicada para extrair conhecimento relevante das características obtidas de sinais de voz e estabelecer regras capazes de classificar uma nova informação ou padrão de dados, permitindo consequentemente o reconhecimento de um determinado locutor.

Bonaldi (2006) apresentou resultados oriundos da necessidade de redução dos custos de produção e aumento da produtividade em processos de diagnóstico de falhas em máquinas elétricas, principalmente em técnicas preditivas que se utilizam de sistemas de monitoração contínua de equipamentos. As indústrias continuam a procura de métodos de identificação e predição de falhas em equipamentos. Um dos novos métodos que vem ganhando espaço na indústria é a análise do sinal de corrente de uma das fases do motor, conhecida como Motor Current Signature Analysis (MCSA). Um dos problemas dessa técnica era que, quando se falava em MCSA logo se associava ao diagnóstico de barras quebradas e excentricidade do *air gap*. A localização de problemas puramente mecânicos através do espectro de corrente ficava sempre em segundo plano. Por esta razão, o trabalho propôs o estabelecimento de padrões inéditos de falhas na carga acoplada. Neste caso, o trabalho propôs a aplicação da Teoria dos Conjuntos Aproximados ao diagnóstico de avarias em Motores de Indução Trifásicos (MIT) e os resultados obtidos apresentaram-se satisfatórios em relação à classificação das falhas resultantes.

Coutinho (2007) mostrou que a fim de melhorar a segurança dos sistemas SCADA (*Surpervisory Control and Data Acquisition*), a técnica de detecção de anomalia tem sido utilizada para identificar valores corrompidos devido a acessos ou faltas provocadas de forma não autorizada. A aplicação desenvolvida é capaz de realizar o monitoramento *on-line* em subestações, e é baseada na extração de conhecimento das bases de dados do sistema utilizando a TCA. A técnica consistiu em projetar e implementar um classificador para detectar leituras não autorizadas.

Camatta (2009) propôs a utilização de malhas de controle aplicadas em acionamentos de máquinas elétricas, abordando o controle de corrente e velocidade no acionamento de

motores de corrente contínua, onde os controladores empregados foram projetados via conceitos de conjuntos aproximados. A ideia consiste na utilização de conceitos de conjuntos aproximados visando à obtenção de uma classe de controladores denominados aproximados. Utilizou-se topologias em cascata compostas por malhas reguladoras de corrente e velocidade. Os resultados obtidos foram comparados com dados de controladores convencionais. Os resultados alcançados por meio de simulações computacionais e por ensaios práticos foram bons e validaram a proposição do trabalho.

Rissino (2009) apresentou uma metodologia híbrida, baseada em ferramentas matemáticas não convencionais que avalia a relevância dos atributos em grandes bases de dados incompletas. As incompletudes são identificadas e classificadas com o objetivo de disponibilizar o estado dos dados e dos registros para uma descoberta de conhecimento confiável, descobrindo um conjunto mínimo de atributos relevantes que represente o conhecimento embutido na base de dados via aplicação da TCA. A identificação e classificação das incompletudes dos dados foi realizada através da aplicação da Lógica Paraconsistente.

Tajiri (2009) apresentou o estudo e a proposição de um controlador digital para o conversor Buck utilizando a TCA. A teoria foi empregada na obtenção de um conjunto de regras capazes de reproduzir o comportamento dos sistemas de controle originalmente utilizados nos conversores estudados. O estudo desenvolvido mostrou que é possível utilizar a TCA para desenvolver controladores que empregam representações baseadas em regras para aplicações em conversores estáticos.

Santos (2009) apresentou o desenvolvimento de um sistema de reconfiguração automática para painéis fotovoltaicos operando sob condições de sombreamento a fim de maximizar a potência de saída. O sistema proposto busca minimizar os efeitos negativos do sombreamento parcial através de uma reorganização nas conexões elétricas do painel, onde os módulos sombreados são conectados em sequência e agrupados em um número limitado de fileiras. O sistema de reconfiguração automática é construído com o auxílio da TCA. A viabilidade do sistema proposto foi avaliada através de simulações com painéis de quatro e seis módulos fotovoltaicos. Os resultados comprovaram a eficácia do sistema proposto.

Carvalho (2010) propôs um método para discretização de atributos contínuos utilizando algoritmos genéticos, que é capaz de realizar uma discretização simultânea de todos os atributos contínuos de um sistema de informação levando em consideração a relação de dependência entre esses atributos. Algoritmos genéticos foram utilizados para determinar os

pontos de corte de valores de atributos de modo a obter uma discretização consistente. Os resultados obtidos mostram a efetividade do método de discretização proposto na aplicação das técnicas da TCA quando comparados com outros métodos de discretização.

Faustino (2011) e Faustino et al. (2011) desenvolveu sistemas computacionais para a previsão de séries temporais, na qual foram utilizados modelos baseados em regras obtidos diretamente da utilização de conceitos relacionados com conjuntos aproximados. Os modelos resultantes foram comparados com outros que utilizavam redes neurais artificiais e modelagem fuzzy. Os resultados obtidos mostraram precisões equivalentes entre os modelos utilizados e as séries empregadas.

Pinheiro et al. (2010a, 2010_b) apresentaram uma metodologia para construção de modelos baseados em regras que emprega conceitos de conjuntos aproximados, cujo objetivo foi obter modelos (estáticos ou dinâmicos) referentes a sistemas lineares, não lineares e com parâmetros variáveis para sistemas SISO (*Single Input Single Output*). Foram apresentados exemplos numéricos para ilustrar a metodologia. Os resultados obtidos apresentaram precisões adequadas e comprovaram a validade da metodologia proposta. Estes trabalhos serviram como base para o desenvolvimento dessa tese, que propõe a extensão da metodologia para modelos (estáticos ou dinâmicos) referentes a sistemas lineares e não lineares com múltiplas variáveis de entrada e de saída (sistemas MIMO).

Rodor (2012) aplicou conceitos de conjuntos aproximados no desenvolvimento de um sistema de controle que utilizava a técnica IMC. O procedimento proposto tinha como objetivo ajustar a constante de tempo do filtro da estrutura do controlador adotado. Os resultados obtidos foram comparados com dados de malhas de controle IMC com o parâmetro do filtro com valor fixo, e também com malhas com controladores convencionais. A abordagem proposta apresentou melhores resultados que as técnicas convencionais utilizadas.

Nos próximos parágrafos deste capítulo são citados alguns trabalhos que utilizaram o *ANFIS* (*Adaptive Network Based Fuzzy Inference System*) para construção de modelos baseados em regras *fuzzy*. O motivo está relacionado com as comparações de resultados que serão realizadas em um capítulo específico deste trabalho, cujos modelos de comparação utilizarão a estrutura *ANFIS*.

Castilho e Melin (2002) consideraram a aplicação do *ANFIS* em processos de previsão do preço de caixas de cebolas e de tomates, e na previsão da variação do peso mexicano em relação ao dólar americano. A comparação foi realizada com os resultados de modelos neurais

obtidos via redes MLP treinadas com o algoritmo "*Levenberg Marquardt*" e o *back-propagation* com termo de momento. Os modelos neurais obtiveram melhores resultados quando aplicados na previsão da variação do peso em relação ao dólar.

Denaï et al. (2004) ilustraram a utilidade das abordagens de inteligência computacional (*soft-computing*) na modelagem de sistemas complexos. Verificaram que pesquisas em inteligência computacional se preocupam com a integração de ferramentas da inteligência artificial (redes neurais, lógica fuzzy e algoritmos evolutivos) em uma estrutura híbrida complementar para resolver os problemas reais. O trabalho concentrou-se no sistema neuro-fuzzy *ANFIS*, que foi usado para a modelagem dinâmica (não linear) da articulações em registros de dados clínicos.

En-ANFIS foi o nome dado ao sistema híbrido proposto por Chen e Zhang (2005) para previsão de séries temporais. O modelo utiliza o paradigma de treinamento "Ensemble" (daí a origem do nome do sistema) que consiste em utilizar vários procedimentos computacionais para a realização de um mesmo objetivo. Cada componente pode ser realizado por meio de conjuntos construídos de forma aleatória ou por meio do método "Bootstrap Sampling" (com ou sem repetições). O valor de previsão é resultado da combinação do resultado de previsão de cada componente isolado, sendo que essa combinação pode ser ponderada ou não. Caso a opção seja por ponderações, se atribui pesos maiores aos valores de previsão de componentes que apresentaram menores erros durante o procedimento de estimação.

Li e Xiong (2005) utilizaram modelos fuzzy gerados via *ANFIS* para prever o mercado de ações de Changai. Para realizar tal procedimento, foram consideradas 244 observações diárias de ações inseridas no intervalo de março de 2004 a março de 2005, para construção dos conjuntos de treinamento e testes, respectivamente. Os resultados indicaram uma modelagem e previsões razoáveis da série, apresentando erro de 1% e 5%, respectivamente.

Nayak et al. (2004) e Firat e Güngör (2007) estudaram, respectivamente, o fluxo de água nos Rios Baitarani na Índia e Great Menderes na Turquia. Nayak et al. (2004) propuseram seis modelos fuzzy, sendo que cada um deles foi gerado por meio de treinamento que considerou de um até seis observações passadas para previsão de um passo à frente. Comparações com modelos com redes neurais e modelos *ARMA* (*AutoRegressive with Moving Average*) foram realizadas. O *ANFIS* obteve melhores resultados. Firat e Güngör (2007) utilizaram quatro subconjuntos de dados a partir das 5844 observações diárias disponíveis, aplicando três dos quatro conjuntos para treinamento do *ANFIS* e um para teste.

Para cada subconjunto de treinamento, foram criados modelos que levaram em consideração até sete observações passadas. A validação dos modelos foi realizada com base na comparação de desempenho de modelos obtidos via redes neurais. Mais uma vez, os modelos fuzzy obtiveram melhores resultados.

Kurian et al. (2006) desenvolveram um modelo fuzzy a partir de treinamento por estrutura *ANFIS* para previsão de um, seis e dez passos à frente de um sistema de iluminação. O conjunto de treinamento foi construído com o auxílio de um software de simulação que simulou as condições de iluminação de um ambiente segundo parâmetros como dimensão da sala, obstruções, posicionamento, condições de iluminação externa, dentre outros.

Liao e Tsao (2007) propuseram um sistema híbrido baseado em lógica nebulosa e algoritmos genéticos para a previsão de demanda de carga elétrica a curto-prazo. A finalidade da aplicação do algoritmo genético consistiu em se estimar os parâmetros de treinamento para um modelo *ANFIS* considerado para determinação do modelo fuzzy de previsão. Como forma de avaliação do modelo obtido, uma comparação foi realizada com modelos obtidos via aplicação de redes neurais. Os modelos baseados em lógica nebulosa obtiveram resultados mais precisos.

Syed-Ahmad et al. (2007) e Ying e Pan (2008) utilizaram-se de modelos fuzzy para prever a demanda de carga elétrica. O primeiro artigo considerou observações horárias de consumo de energia elétrica para compor o conjunto de treinamento usado para obtenção do modelo fuzzy. O modelo obtido foi aplicado na previsão de demanda vinte e quatro horas à frente. O segundo artigo utilizou observações anuais de 1981 até 1996 para previsão de carga para os anos de 1997 até 2000 em diferentes regiões de Taiwan. Em ambos os casos os modelos obtidos via *ANFIS* obtiveram resultados compatíveis com os modelos neurais usados para fins de comparação.

Ming-Bao e Xin-Ping (2008) aplicaram um sistema híbrido *ANFIS* e algoritmo genético para prever o fluxo de tráfego de automóveis. Para obtenção dos dados de treinamento, um software de simulação de propósito específico foi utilizado para gerar novecentas observações de fluxo sob determinadas condições. Das novecentas observações, oitocentas foram usadas para criação do conjunto de treinamento e cem foram usadas para compor o conjunto de testes. Os dados de treinamento foram agrupados por meio do algoritmo "*Subtractive Clustering*" e, com base nesse agrupamento, o processo de

treinamento foi inicializado. A definição de um raio de agrupamento (clusterização) ótimo foi realizada por meio da aplicação de um algoritmo genético. Os resultados foram comparados aos resultados obtidos via aplicação de modelos gerados por redes neurais e modelos *ANFIS*. O sistema híbrido apresentou melhores resultados.

Fahimifard et al. (2009) realizaram um estudo de desempenho entre métodos lineares e não lineares quando aplicados em processos de previsão. A técnica linear considerada foi um modelo *ARIMA* e a não linear foi um modelo *ANFIS*. A fim de comparar ambas as técnicas, uma série que define uma variável agrícola iraniana foi considerada. O processo considerou a previsão em três horizontes distintos com uma, duas e quatro semanas respectivamente. A análise dos resultados permitiu concluir que o modelo *ANFIS* obteve um desempenho consideravelmente maior em todos os experimentos realizados.

Chen et al. (2010), a partir do uso de uma estrutura *ANFIS*, estimaram um modelo fuzzy capaz de prever a quantidade de turistas que chegam a Taiwan vindos dos Estados Unidos, Japão e Hong Kong. Para construção do modelo, um conjunto de treinamento que levava em consideração observações mensais de 1989 a 2000 foi construído. O objetivo consistia em estimar o número de turistas para os anos de 2001 a 2003.

Faustino et al. (2012) desenvolveram um modelo fuzzy via *ANFIS* para alguns exemplos de séries temporais práticas. Os resultados obtidos foram comparados com modelos gerados por redes neurais artificiais com diferentes estruturas. Em geral, os modelos fuzzy apresentaram melhores valores nas estimações das séries em relação aos modelos com apenas a utilização de redes neurais.

CAPÍTULO 3

3 CONCEITUAÇÃO DE CONJUNTOS APROXIMADOS

O objetivo deste capítulo é apresentar os conceitos básicos e os fundamentos sobre conjuntos aproximados, fornecendo os subsídios necessários para a compreensão da aplicação dos mesmos na proposição deste trabalho.

3.1 Introdução

O processamento de informações na presença de incertezas é reconhecido como sendo um procedimento de grande importância em várias áreas, entre elas em sistemas computacionais que empregam técnicas de Inteligência Artificial (*IA*). Neste contexto, entende-se por incerteza, informações incompletas, vagas e/ou imprecisas. Representações computacionais em áreas diversas de *IA* devem ser capazes de lidar com incertezas. Estes sistemas geralmente utilizam modelos apropriados para representar a informação incerta, bem como para controlar sua combinação e propagação nos processamentos correspondentes.

A capacidade de observar uma determinada quantidade de informação e extrair um conhecimento associado é inerente ao ser humano e à sua capacidade de aprendizado. Porém a realização automática desta tarefa por meios computacionais pode ser complexa, principalmente quando as informações são desorganizadas, incompletas ou possuem partes irrelevantes. A Teoria dos Conjuntos Aproximados (*TCA*) é utilizada para facilitar a transformação automática de dados em conhecimento (Pawlak, 1991), na forma de representações computacionais baseadas em conjuntos de atributos ou em regras de decisão, como aquelas definidas por Sistemas Especialistas (*SE*), por exemplo.

A TCA é considerada como uma extensão da teoria clássica dos conjuntos. A mesma foi proposta por Zdzislaw Pawlak (Pawlak, 1982) como uma nova ferramenta matemática para o tratamento de incertezas e imprecisões em Sistemas de Informações (*SI*) em geral.

3.2 A Teoria dos Conjuntos Aproximados

A abrangência de um *SI* construído a partir de conceitos de conjuntos aproximados (*rough sets*) está em sua capacidade de classificar informações adequadamente. A granulação ou classificação do conhecimento é obtida através do estabelecimento de relações entre as

informações disponíveis a respeito do objeto em estudo. O sucesso na elaboração de conexões entre as informações sobre um determinado assunto indica que foi possível extrair o conhecimento disponível dos dados fornecidos.

A estruturação de um conjunto de informações por meio da *TCA* permite que não apenas os fatos presentes em informações completas contribuam para o enriquecimento do conhecimento sobre o assunto, mas que também sejam extraídas parcelas de conhecimento contidas em informações incompletas. A granulação do conhecimento possibilita que se aproveite o conhecimento presente em qualquer parcela de informação. Essa é uma característica desejável especialmente em casos onde as informações disponíveis são incompletas ou não representam a totalidade de situações possíveis sobre um determinado contexto.

Uma das principais vantagens da *TCA* é poder representar as similaridades conceituais entre os dados de um determinado *SI*, agrupando valores que são conceitualmente similares ou equivalentes. Valores que pertencem a um mesmo grupo são considerados *indiscerníveis* e, assim, pode-se levar em consideração o significado intrínseco dos dados e a relação que existe entre eles, e não tratar os seus valores somente de maneira isolada. Um *SE* baseado na *TCA* tem a capacidade de tomar as decisões apropriadas a partir das situações ao qual é submetido. Outra vantagem desta teoria é de não necessitar de qualquer informação adicional ou preliminar a respeito dos dados a serem processados, tais como distribuição de probabilidade, atribuição de valor de crença, ou grau de possibilidade.

Neste contexto, um tipo específico de representação tabular é denominado de "tabela de decisão" (Pawlak et al., 1995). Estas tabelas são representações bidimensionais formadas por objetos no formato atributo-valor. Tais objetos encontram-se dispostos em tabelas de decisão e são agrupados em classes (Pawlak, 2003).

Um sistema de apoio à tomada de decisão geralmente é definido por uma tabela de um *SI*, independentemente do contexto do problema em análise. Contudo, é comum a situação em que nem todos os atributos presentes na tabela de representação correspondente sejam úteis no processo de tomada de decisão. Estas informações desnecessárias e que não modificam os resultados com relação à decisão a ser tomada são denominados de atributos irrelevantes. Dessa forma, para a construção de um modelo de auxílio ao processo de tomada de decisão, é de interesse considerar somente aqueles atributos fundamentais para a tomada da decisão.

Com a *TCA* é possível, portanto, classificar os atributos em duas formas distintas, como *dispensáveis* – aqueles que, se omitidos ou inexistentes, não trazem nenhum problema de classificação – e como *indispensáveis* – que se forem omitidos geram problemas na classificação do sistema de decisão. Esta teoria fornece meios sistemáticos para eliminar atributos irrelevantes por meio de um processo de redução do sistema de informação original. O mecanismo utilizado para esta finalidade é baseado na definição de redutos (Pawlak e Skowron, 2007). Contudo, é comum encontrar nos dados que compõem um *SI*, objetos cujos dados são inconsistentes. Essa inconsistência impede que a discernibilidade entre objetos seja realizada, visto que existem objetos com mesmos valores dos atributos classificados em classes diferentes. A administração dessa imprecisão é realizada por meio dos conceitos de aproximação inferior e aproximação superior (Pawlak e Skowron, 2007).

É apresentada abaixo uma série de vantagens da aplicação da *TCA* na solução de problemas diversos, segundo o trabalho de Tay e Shen (2002).

- A construção de modelos de decisão tratados por esta teoria necessita somente dos dados originais referentes ao problema em estudo e não de informações adicionais, como é o caso dos modelos estatísticos que frequentemente necessitam de informações da distribuição de probabilidade, e dos modelos fuzzy que necessitam de informações como grau de possibilidade;
- O modelo baseado em regras resultante fornece informações capazes de analisar não somente dados quantitativos mais também qualitativos;
- A aplicação da teoria permite a descoberta de informações representativas ocultas nos dados, e expressa esses fatos na forma de regras de decisão cuja sintaxe é bem próxima a linguagem natural;
- O conjunto de regras que compõe uma determinada representação resulta em uma descrição generalizada do conhecimento presente no conjunto de dados, eliminando eventuais redundâncias nos mesmos;
- As regras de decisão obtidas são baseadas em informações pertinentes visto que foram geradas a partir de informações de sistemas práticos;
- As representações obtidas por meio da teoria pertencem a uma classe de modelos cuja característica principal é a facilidade de entendimento e análise, visto que são formados por regras cuja sintaxe é muito próxima da linguagem natural. Diferentemente outras técnicas como as redes neurais artificiais, por exemplo, que constituem modelos cuja interpretação não é explicita em relação

aos pesos associados aos neurônios ou as estruturas das redes empregadas, o que dificulta a extração do conhecimento associado ao contexto da aplicação.

3.3 Conceitos Básicos

Um espaço aproximado é definido por $S = (U, A)$, onde U é um conjunto de objetos ou observações (o_1) chamado de universo e A é um conjunto de atributos de condições (a_j). Seja um sistema de informação representado por uma tabela de atributos-valores, onde se determinam classificações $f: U\ x\ A$. A representação tabular genérica de um *SI* está ilustrada na Tabela 3.1, onde valores de atributos de decisão são definidos na coluna *d.*

Tabela 3.1 - Representação tabular genérica de um *SI*.

	a_1	...	a_j	...	a_n	d
o_1	$f(o_1, a_1)$		$f(o_1, a_j)$		$f(o_1, a_n)$	$f(o_1, d_1)$
⋮	⋮		⋮		⋮	⋮
o_i	$f(o_i, a_1)$	...	$f(o_i, a_j)$	...	$f(o_i, a_n)$	$f(o_i, d_i)$
⋮	⋮		⋮		⋮	⋮
o_m	$f(o_m, a_1)$	...	$f(o_m, a_j)$	...	$f(o_m, a_n)$	$f(o_m, d_m)$

Em conjuntos aproximados trabalha-se geralmente com valores discretos. Para atributos numéricos é necessário aplicar um processo de discretização para torná-los nominais (será apresentado um resumo sobre alguns métodos de discretização no item 3.3.1). Algumas abordagens podem ser utilizadas para minimizar eventuais efeitos da quantização dos dados (Nguyen e Skowron, 1995; Carvalho, 2010).

Outro conceito importante da teoria é o conceito de relação de não discernimento ou indiscernibilidade. Se essa relação existe entre dois objetos, tem-se para um conjunto de atributos desses objetos, onde se os valores desses atributos são idênticos entre si, não se distingue um objeto do outro (Goh e Law, 2003). Para cada subconjunto de atributos $B \subseteq A$ uma relação de equivalência $IND(B)$ é associada, recebendo o nome de relação de não discernimento e sendo definida pela expressão (3.1).

$$IND(B) = \{(O_i, O_j) \in U^2 \big| \forall a_k \in B, f(o_i, a_k) = f(o_j, a_k)\} \quad (3.1)$$

O conjunto de todas as classes de equivalência determinadas por $IND(B)$ é representado pela notação $U / IND(B)$. Das classes de equivalência emergem dois outros conceitos importantes: a aproximação inferior e a aproximação superior. Dado que O é um conjunto tal que $O \subseteq U$, pode se definir:

- ✓ A aproximação inferior B_* representada pela equação (3.2), é definida como o conjunto de possíveis objetos que podem ser classificados com certeza como membros de O utilizando o conjunto de atributos B (Pawlak e Skowron, 2007).

$$B_*(O) = \{o \in U | IND(B) \subseteq O\} \quad (3.2)$$

- ✓ A aproximação superior B^* representada pela equação (3.3), é definida como a região onde existe a possibilidade dos elementos serem parte da classificação em questão.

$$B^*(O) = \{o \in U | IND(B) \cap O \neq \emptyset\} \quad (3.3)$$

Um conjunto O é denominado preciso (*crisp*) se $B_*(O) = B^*(O)$, caso contrário, ele é definido como impreciso, grosseiro (*rough*) ou aproximado. Todos os elementos do conjunto de aproximação inferior fazem parte do conjunto de aproximação superior.

A partir de (3.2) e (3.3) define-se também a região de fronteira expressa por (3.4). A região de fronteira de O representa a região de incerteza, que consiste de objetos impossíveis de serem classificados em O.

$$RF(O) = B^*(O) - B_*(O) \quad (3.4)$$

As aproximações em questão podem ter sua qualidade medida em termos dos próprios elementos que a definem:

- O coeficiente de imprecisão $\alpha_B\ (O)$ define a qualidade da aproximação em relação aos elementos do conjunto O e é definido por (3.5).

$$\alpha_B(O) = \frac{|B_*(O)|}{|B^*(O)|} \quad (3.5)$$

Onde $\left|B_*(X)\right|$ e $\left|B^*(X)\right|$ denotam a cardinalidade das aproximações inferior e superior respectivamente, sendo conjuntos não vazios. O valor $\alpha_B(O)$ pertence ao intervalo [0, 1].

Quanto mais próximo de um, mais preciso é O em relação ao conjunto de atributos de *B*, e quanto mais próximo de zero mais impreciso (*rough*) é O em relação ao conjunto de atributos de *B*.

- O coeficiente da qualidade da aproximação superior $\alpha_B\ (B^*(O))$ pode ser interpretado como sendo o percentual de todos os objetos possivelmente classificados como pertencentes a O e é dado pela equação (3.6), sendo $\left|U\right|$ a cardinalidade do conjunto de objetos de sistema de informação em questão.

$$\alpha_B\big(B^*(O)\big) = \frac{|B^*(O)|}{|U|} \tag{3.6}$$

- O coeficiente da qualidade da aproximação inferior $\alpha_B(B_*(O))$ pode ser interpretado como sendo o percentual de todos os objetos certamente classificados como pertencentes a O e é definido por (3.7).

$$\alpha_B\big(B_*(O)\big) = \frac{|B_*(O)|}{|U|} \tag{3.7}$$

Uma matriz de discernibilidade de um sistema de informação é denotada por $M_D(B)$, constituindo uma matriz simétrica de dimensão $n\ x\ n$ dada por (3.8), cujos elementos são dados por (3.9). Os elementos $m_D(i,j)$ da matriz de discernibilidade constituem o conjunto de atributos condicionais de B que diferenciam os objetos das classes com relação aos seus valores nominais.

$$M_D(B) = [m_D(i,j)\ ]_{n\ x\ n}, i \geq 1, j \leq card(U\ /\ IND(B)\ ) \tag{3.8}$$

$$m_D(i,j) = \{a_k \in B | f(o_i, a_k) \neq f\big(o_j, a_k\big)\} \tag{3.9}$$

Uma função de discernibilidade $F(B)$ é uma função booleana que determina o conjunto mínimo de atributos necessários para diferenciar qualquer classe de equivalência das demais para um determinado *SI*, sendo definida por (3.10) e (3.11). A função de discernibilidade $F(B)$ é obtida da seguinte forma: para os atributos contidos dentro de cada célula da matriz de discernibilidade, aplica-se o operador "soma", "or" ou " ∨ " e , entre as células dessa matriz, utiliza-se o operador "produto", "and" ou "∧", resultando em uma expressão booleana na forma de "*Produto-da-Soma*". Simplificando esta expressão, utilizando

teoremas, propriedades e postulados da Álgebra Booleana, obtém-se a expressão minimizada na forma de "Produto da Soma" ou "Soma de Produto".

$$F(B) = \wedge \{\vee \overline{m}_D(i,j)\} \tag{3.10}$$

$$\overline{m}_D(i,j) = \{\overline{a_k} \mid a_k \in m_D(i,j)\} \tag{3.11}$$

O conjunto formado pelo termo mínimo de $F(B)$ determina os chamados redutos de *B*. Reduto é um conjunto de atributos mínimos necessários para manter as mesmas propriedades de um *SI*, que utiliza todos os atributos originais do sistema. Pode existir mais de um reduto para um mesmo conjunto de atributos. A obtenção dos redutos mínimos de um *SI* de dimensão elevada, geralmente consiste em um problema de complexidade computacional crescente com o volume de dados do processo. Algumas abordagens são utilizadas para tratar este tipo de problema no processamento de redutos, por exemplo, por intermédio de relações de similaridade (Huang et al., 2007).

Para transformar um reduto em regra de decisão, basta agregar os valores dos atributos condicionais da classe de objetos da qual foi originado o reduto, com os atributos correspondentes ao mesmo, e depois completar a regra com os atributos de decisão. Para um determinado reduto, um exemplo de regras de decisão pode ser expresso por (3.12). A utilização da teoria dos conjuntos aproximados possibilita de modo sistemático, que as regras de decisão resultantes apresentem informações concisas em relação a um determinado *SI*, tratando adequadamente eventuais redundâncias, incertezas, ou imprecisões presentes nos dados.

$$IF\ a_1 = f(o_1, a_1) AND \dots AND\ a_k = f(o_m, a_k)\ THEN\ d_1 = f(o_1, d_1)\ OR \dots OR\ f(o_i, d_i) \tag{3.12}$$

3.3.1 Discretização

Como citado anteriormente, a TCA não permite a utilização de atributos numéricos fracionários diretamente, uma vez que o conjunto de valores nominais para esses atributos seriam infinitos. A discretização é um pré-processamento dos dados aplicado a um *SI* de forma a transformar atributos numéricos em valores nominais. A discretização transforma

valores fracionários em valores de intervalos inserindo pontos de corte, que são limiares entre um intervalo e outro.

Os métodos de discretização podem ser divididos em (Carvalho, 2010):

Estáticos ou dinâmicos: Esta classificação da discretização refere-se ao momento em que a discretização é realizada. A discretização estática discretiza os dados antes da extração de padrões do sistema de informação. Na técnica dinâmica, a discretização ocorre ao mesmo tempo em que os padrões são obtidos;

Top-down ou botton-up: Os métodos *top-down* iniciam a discretização sem pontos de corte e, durante a discretização, vão inserindo novos pontos dividindo os valores em intervalos menores. Métodos *botton-up* determinam, no início da discretização, intervalos aos quais os valores contínuos pertencem e esses intervalos são agrupados de acordo com critérios inerentes ao algoritmo;

Direto ou Incremental: Os métodos diretos dividem os valores do atributo em um número pré-determinado de intervalos, enquanto os métodos incrementais definem o número de intervalos por um processo de otimização, com critério de parada pré-definido;

Univariado ou multivariado: Métodos univariados consideram um atributo contínuo por vez, não considerando a relação entre os atributos. Métodos multivariados consideram múltiplos atributos, bem como a relação de dependência entre eles;

Supervisionado ou não supervisionado: A discretização é chamada supervisionada quando a mesma leva em consideração o atributo classe, podendo ser univariado ou multivariado. Nos métodos não supervisionados, os atributos são discretizados desprezando o atributo classe e qualquer outro atributo do sistema de informação.

Vários métodos de discretização foram e continuam sendo desenvolvidos, uma vez que certas características de determinados problemas podem requerer especificidades do algoritmo de discretização que não se classifica entre os métodos existentes. Alguns métodos comumente utilizados são:

***Equal-with Binning*:** Método não supervisionado de discretização simples. Divide o espaço dos valores observados em intervalos de igual tamanho;

***Equal-frequency Binning*:** Método não supervisionado e univariado. Pode produzir intervalos de tamanhos diferentes, porém cada intervalo deve conter, aproximadamente, o mesmo número de exemplos. Nesse método, o número de intervalos a serem criados deve ser previamente definido;

Entropia MDL: Este método é supervisionado e univariado apresentado por Fayyad e Irani (1993). Este método é supervisionado e univariado. É baseado em uma heurística de entropia mínima. O algoritmo não requer nenhum parâmetro, a discretização ocorre automaticamente.

Boolean Reasoning: Método supervisionado e univariado. Faz uso de conceitos de conjuntos aproximados e raciocínio booleano (Skowron and Nguyen, 1999). O algoritmo agrupa os conjuntos de dados da matriz de regressão (atributos de condição) em classes que conseguem "explicar" de maneira consistente a alteração do atributo de decisão. Obtendo-se os redutos e as regras de decisão, determina-se por intermédio do antecedente de cada uma das regras, as partições referentes a todos os atributos.

Neste trabalho na etapa de experimentos, haverá um exemplo de um sistema MIMO discreto 2x2 (Wang et al., 2009) com a discretização *Equal frequency binning*, e outro exemplo MIMO 2x2 de um sistema de nível real, que além da discretização *Equal frequency binning*, também usará o método *Boolean Reasoning* para fins de comparações. Para a discretização com os dois métodos, mais a geração de regras de decisão, empregou-se o aplicativo Rosetta (Ohrn e Komorowski, 1997).

Exemplo de Aplicação dos Conceitos de Conjuntos Aproximados:

Decisão sobre tipos de exames clínicos para diagnósticos.

Este exemplo tem como objetivo determinar qual é a menor combinação de exames clínicos necessários para classificar determinados diagnósticos que estão sendo avaliados em um experimento clínico. A Tabela 3.2 mostra exemplos de determinados exames (Exame X) clínicos que podem classificar três tipos de diagnósticos ($D1, D2\ e\ D3$). Os resultados dos exames são classificados como positivo (P) ou negativo (N), e em faixas de valores como 1, 2, 3 e 4, ou < 12, 12-23, 24-32 e > 32.

Tabela 3.2 - Sistema de informação do Exemplo.

	$\boldsymbol{Exame_1}$	$\boldsymbol{Exame_2}$	$\boldsymbol{Exame_3}$	$\boldsymbol{Exame_4}$	**Diag**
o_1	1	P	2	> 32	D1
o_2	2	P	3	12-23	D2
o_3	4	N	3	24-32	D3
o_4	1	N	2	< 12	D1
o_5	3	N	1	24-32	D2
o_6	1	N	4	< 12	D1

Como em conjuntos aproximados geralmente trabalha-se com valores discretos (e não contínuos), os valores das faixas referentes ao atributo "$Exame_4$" podem ser expressos como $\boldsymbol{F1} \rightarrow < 12$, $\boldsymbol{F2} \rightarrow 12-23$, $\boldsymbol{F3} \rightarrow 24-32$ e $\boldsymbol{F4} \rightarrow > 32$. Usando as definições sobre conjuntos aproximados é possível estabelecer as informações básicas referentes a este sistema de informação em questão:

$U = \{o_1, o_2, o_3, o_4, o_5, o_6\}$; U é o conjunto de observações.

$A = \{\text{Exame}_1, \text{Exame}_2, \text{Exame}_3, \text{Exame}_4\}$; A é o conjunto de atributos de condição.

$V_{\text{Exame}_1} = V_{\text{Exame}_3} = \{1,2,3,4\}$;

$V_{\text{Exame}_2} = \{N, P\}$;

$V_{\text{Exame}_4} = \{F1, F2, F3, F4\}$

$V_{\text{Diag}} = \{D1, D2, D3\}$;

Da equação (3.1), tem-se:

$f(o_1, \text{Exame}_1) = 1; \quad f(o_1, \text{Exame}_2) = P.$

$IND(\{\text{Exame}_2\}) = \{\{o_1, o_2\}, \{o_3, o_4\}, \{o_3, o_5\}, \{o_3, o_6\}, \{o_4, o_5\}, \{o_4, o_6\}, \{o_5, o_6\}\}.$

$U/IND(\{\text{Exame}_2\}) = \{\{o_1, o_2\}, \{o_3, o_4, o_5, o_6\}\}.$

Das equações (3.2) e (3.3), vem:

$-$ Para $B = \{\text{Exame}_1, \text{Exame}_3\} \rightarrow IND(B) = \{\{o_1, o_4\}\}$;

$U/IND(B) = \{\{o_1, o_4\}, \{o_2\}, \{o_3\}, \{o_5\}, \{o_6\}\}.$

$-$ Para $O = \{o_1, o_2, o_3\} \rightarrow B_*(O) = \{o_2, o_3\}; \quad B^*(O) = \{o_1, o_2, o_3, o_4\};$

De (3.5), (3.6) e (3.7) tem-se:

$$\rightarrow \alpha(O) = \frac{2}{4} = 0.5; \quad \alpha(B^*(O)) = \frac{2}{6} = 0.33; \quad \alpha(B_*(O)) = \frac{4}{6} = 0.67.$$

A Tabela 3.3 ilustra a matriz de discernibilidade do sistema de informação correspondente (equações (3.8) e (3.9)).

Tabela 3.3 - Matriz de discernibilidade do exemplo.

	O_1	O_2	O_3	O_4	O_5	O_6
O_1	-					
O_2	Ex1, Ex3, Ex4	-				
O_3	Ex1, Ex2, Ex3, Ex4	Ex1, Ex2, Ex4				
O_4	Ex2, Ex4	Ex1, Ex2, Ex3, Ex4	Ex1, Ex3, Ex4			
O_5	Ex1, Ex2, Ex3, Ex4	Ex1, Ex2, Ex3, Ex4	Ex1, Ex3	Ex1, Ex3, Ex4		
O_6	Ex2, Ex3, Ex4	Ex1, Ex2, Ex3, Ex4	Ex1, Ex3, Ex4	Ex3	Ex1, Ex3, Ex4	

Com as informações da matriz de discernibilidade, obtém-se a função de discernibilidade associada (equações (3.10) e (3.11)).

$$\begin{aligned} F(B) = {} & (E_{x1} \vee E_{x3} \vee E_{x4}) \wedge (E_{x1} \vee E_{x2} \vee E_{x3} \vee E_{x4}) \wedge (E_{x2} \vee E_{x4}) \\ & \wedge (E_{x1} \vee E_{x2} \vee E_{x3} \vee E_{x4}) \wedge (E_{x2} \vee E_{x3} \vee E_{x4}) \wedge (E_{x1} \vee E_{x2} \vee E_{x3}) \\ & \wedge (E_{x1} \vee E_{x2} \vee E_{x3} \vee E_{x4}) \wedge (E_{x1} \vee E_{x2} \vee E_{x3} \vee E_{x4}) \\ & \wedge (E_{x1} \vee E_{x2} \vee E_{x3} \vee E_{x4}) \wedge \ (E_{x1} \vee E_{x3} \vee E_{x4}) \wedge (E_{x1} \vee E_{x3}) \\ & \wedge (E_{x1} \vee E_{x3} \vee E_{x4}) \wedge (E_{x1} \vee E_{x3} \vee E_{x4}) \wedge (E_{x3}) \wedge (E_{x1} \vee E_{x3} \vee E_{x4}); \end{aligned}$$

$$F(B) = (Ex1 \vee Ex3 \vee Ex4) \wedge (Ex1 \vee Ex2 \vee Ex3 \vee Ex4) \wedge \ldots \wedge (Ex1 \vee Ex3 \vee Ex4).$$

Simplificando e colocando na forma da "Soma de Produto", vem:

$$F(B) = (Ex1) \vee (Ex2 \wedge Ex3) \vee (Ex3 \wedge Ex4).$$

Assim, têm-se os Redutos = {{Exame1}, {Exame2, Exame3}, {Exame3, Exame4}}. Escolhendo o reduto com menor número de atributos Reduto = {Exame1}, têm-se as regras de decisão correspondentes:

IF Exame1 = 1 THEN Diag = D1;
IF Exame1= 2 THEN Diag = D2;
IF Exame1= 4 THEN Diag = D3;
IF Exame1= 3 THEN Diag = D2.

Logo, conclui-se que para a classificação dos diagnósticos (D1, D2 e D3) estudados é suficiente o exame do tipo 1 (Exame_1).

CAPÍTULO 4

4 REVISÃO DE CONCEITOS SOBRE MODELOS FUZZY

O objetivo deste capítulo é apresentar conceitos básicos sobre modelos fuzzy. Esses conceitos serão importantes para o entendimento de comparações entre os resultados obtidos por modelos fuzzy, e a modelagem proposta neste trabalho que utiliza conceitos de conjuntos aproximados.

4.1 Introdução

A Lógica Fuzzy e a Teoria dos Conjuntos Nebulosos (*Fuzzy Sets*) podem ser utilizadas para traduzir em termos linguísticos informações imprecisas, vagas ou com incertezas. O marco inicial está relacionado à publicação do artigo denominado *Fuzzy Sets* (Zadeh, 1965), onde foi proposta uma forma para expressar e operar matematicamente conceitos subjetivos, permitindo o tratamento de problemas que envolvem conceitos abstratos e subjetivos. O enfoque linguístico no processamento de problemas complexos foi proposto por (Zadeh, 1973).

4.2 Conjuntos Nebulosos

O que distingue a lógica difusa em relação à lógica clássica é a associação de conjuntos com valores numéricos intermediários entre o zero e o um. Na teoria clássica dos conjuntos, o conceito de pertinência de um elemento *x* a um dado conjunto *A*, em um universo de discurso *X* (que relaciona um conjunto de valores permitidos para o elemento *x* em questão), associa a cada elemento deste universo *X* um valor binário que determina se o elemento *x* pertence ou não pertence àquele conjunto *A*. Esse procedimento pode ser representado pela função característica f_A, definida por (4.1).

$$f_A(x) = \begin{cases} 1 \; se \; x \; \in A \\ 0 \; se \; x \; \notin A \end{cases} \tag{4.1}$$

Em conjuntos nebulosos a função de pertinência correspondente possui um grau de pertinência (Klir, Yuan, 1995) definida por (4.2) que pode possuir infinitos valores no

intervalo [0, 1]. A Figura 4.1 ilustra as diferenças entre conjuntos da lógica clássica e a lógica difusa em uma representação de informações de temperatura.

$$\mu_A(x): x \rightarrow [0,1]; x \in A \tag{4.2}$$

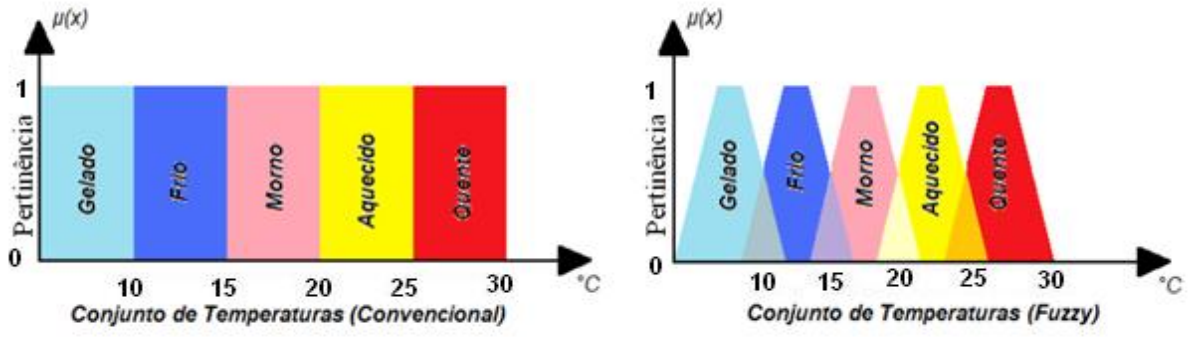

Figura 4.1 - Representação comparativa de conjuntos convencionais e conjuntos fuzzy.

A cada conjunto nebuloso pode ser associado uma variável linguística que pode possuir termos, terminologias, nomes ou rótulos, ao invés de números, permitindo uma modelagem similar ao pensamento humano (Zadeh, 1996). A quantidade de valores linguísticos define a especificação e distribuição dos termos e, por conseguinte, a partição nebulosa do universo de discurso correspondente. Um número pequeno de termos linguísticos define uma partição esparsa ou grossa, ao passo que um número maior resulta numa partição fina.

As funções de pertinência associadas aos conjuntos nebulosos correspondentes podem ser de diversos tipos (Zimmermann, 2001). Os tipos mais usados são: singleton; crisp; triangular; trapezoidal; gaussiana. As mesmas são representadas respectivamente pelas equações (4.3), (4.4), (4.5), (4.6) e (4.7), onde *a* e *b* são os valores limites das funções, *m* e *n* são os valores modais e *k* é uma constante numérica.

.

$$A(x) = \begin{cases} 1, se\ x = a \\ 0, se\ x \neq a \end{cases} \tag{4.3}$$

$$A(x) = \begin{cases} 0\ , se\ x < a \\ 1, se\ a \leq x \leq b \\ 0\quad , se\ x > b \end{cases} \tag{4.4}$$

$$A(x) = \begin{cases} 0, se\ x < a \\ \dfrac{x-a}{m-a}, se\ x\ \in\ [a,m] \\ \dfrac{b-x}{b-m}, se\ \ x \in [m,b] \\ 0, se\ x > b \end{cases} \tag{4.5}$$

$$A(x) = \begin{cases} 0, se\ x < a \\ \dfrac{x-a}{b-a}, se\ x\ \in\ [a,b] \\ 1, se\ x\ \in\ [b,c] \\ \dfrac{d-x}{d-c}, se\ \ x \in [d,c] \\ 0, se\ x > d \end{cases} \tag{4.6}$$

$$A(x) = e^{-k(x-m)^2} \tag{4.7}$$

O processo de mapear um valor real de uma determinada variável para valores de funções de pertinência correspondentes recebe o nome de *fuzificação* (Zimmermann, 2001). Em outras palavras, *fuzificação* é a atribuição de um valor do universo de discurso em um grau de pertinência correspondente.

Um sistema difuso é composto por um conjunto de regras fuzzy que podem assumir a seguinte forma IF $x = A\ And\ y = B\ And\ \ Then\ z = C$, onde A, B e C são rótulos linguísticos que representam cada um dos conjuntos nebulosos considerados. Frequentemente, $x = A\ And\ y = B\ And\ \cdots$ é denominado de antecedente da regra enquanto $z = C$ é denominado de consequente da regra. Um conjunto de regras fuzzy tem como objetivo representar o conhecimento de um especialista, ou modelo do problema abordado (Dubois, Prade e Yanger, 1996).

Entende-se por regras ativadas, aquelas cujo processamento do antecedente para suas entradas gerou graus de pertinência não nulos. A associação das informações das pertinências não nulas pode ser realizada por meio da aplicação de um operador *T - Norma*. Os dois operadores mais utilizados são o *Produto* (*PROD(A,B,...)*) e o *Mínimo* (*MIN(A,B,...)*) cujos resultados consistem, respectivamente, no produto dos valores de pertinência das variáveis de entrada nos conjuntos definidos nos termos do antecedente da regra, e no menor valor de pertinência dentre as variáveis de entrada para os conjuntos representados nos termos dos antecedentes das regras (Klir e Yuan, 1995).

Existem vários modelos de regras que podem ser usados em sistemas fuzzy. Os dois mais utilizados são o *Modelo Mamdani* e o *Modelo Takagi-Sugeno*.

No Modelo *Mamdani*, as regras do sistema assumem o seguinte formato:

$$Se\ x_1 = X_1^{(i)}\ E\ x_2 = X_2^{(i)}\ E\ \ldots E\ x_k = X_k^{(i)}\ Então\ Y_n = Y^m.$$

Já no Modelo *Takagi-Sugeno*, as regras do sistema assumem o seguinte formato:

$$Se\ x_1 = X_1^{(i)}\ E\ x_2 = X_2^{(i)}\ E\ \ldots E\ x_k = X_k^{(i)}\ Então\ Y_n = c_{oi} + c_{1i}x_1 + c_{2i}x_2 + \cdots + c_{ki}x_k.$$

Em ambos os casos, o antecedente das regras relaciona o valor de uma variável de entrada com um valor linguístico definido por uma função de pertinência. Com relação ao consequente, no *Modelo Mamdani* o valor de saída constitui uma ponderação da composição das regras ativadas. No Modelo *Takagi-Sugeno* o valor do termo consequente de cada regra será o valor resultante da função polinomial presente no consequente da regra ativada (Zimmermann, 2001).

Uma vez determinado quais regras foram ativadas, pode-se realizar a "defuzificação". *Defuzificação* é o processo de mapear valores do conjunto *fuzzy* para valores das variáveis do problema abordado, consistindo em um método de ponderação (Zimmermann, 2001). Existem vários métodos de defuzificação sendo que o mais empregado é o "método de centro de área", definido pela equação (4.8), onde v_n é o valor do consequente da regra n e $\mu(n)$ é o grau de ativação da regra n.

$$z = \frac{\sum_{n=1}^{N} v_n \mu(v_n)}{\sum_{n=1}^{N} \mu(v_n)} \tag{4.8}$$

Uma vez definido o modelo de regras a ser aplicado e as funções de pertinência para os conjuntos fuzzy que irão compor a base de dados, um "*Sistema de Inferência Fuzzy*" (*FIS*), pode ser proposto. A Figura 4.2 ilustra a arquitetura de um *FIS* genérico.

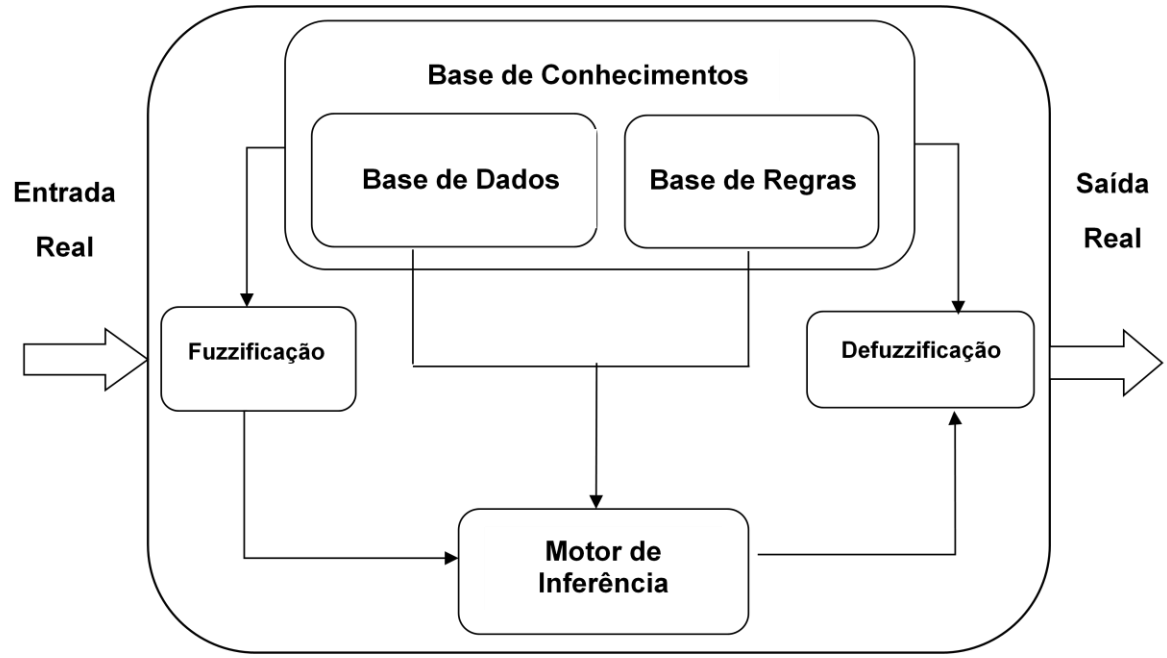

Figura 4.2 - Arquitetura geral de um FIS.

Um *FIS* recebe valores numéricos e transforma esses valores em interferências linguísticas (*fuzificação*) (Dubois et al., 1996). Em seguida, a variável *fuzzy* é processada por um motor de inferência que faz uso das regras nebulosas do sistema *fuzzy*. No final do processo, a variável *fuzzy* resultante é novamente transformada em uma variável real associada ao problema abordado, podendo então ser utilizada em processos externos à computação do modelo *fuzzy* em questão.

4.3 Sistemas Neuro-Fuzzy

A ideia básica de um sistema neuro-fuzzy é realizar um *FIS* com auxílio de redes neurais artificiais caracterizando um sistema híbrido, de forma que os paradigmas de aprendizado usuais às redes neurais possam ser aplicados. Dentre os vários *Sistemas Neuro-Fuzzy* propostos, pode-se destacar o *ANFIS* (*Adaptive Network Based Fuzzy Inference System*), o *FSOM* (*Fuzzy Self-Organization Map*), o *NEFClass* (*Neuro Fuzzy Classification*), o *NEFCON* (*Neuro-Fuzzy control*) e o *DENFIS* (*Dynamic Evolving Neural-Fuzzy Inference System*). O *FSOM*, proposto por Mitra e Pal (1994), é um sistema baseado no modelo de mapas auto-organizáveis de Kohonen. O *NEFClass*, proposto por Nauck e Kruse (1995), é um Sistema Neuro-Fuzzy cuja aplicação básica é em sistemas de classificação (4.9). A diferença principal entre o *NEFClass* e o *FSOM* é que o primeiro apresenta valores *crisp* na camada de

saída que generaliza o formato das regras fuzzy que compõem os modelos gerados pelo *NEFClass*.

$$SE\ x_1 \in \mu_1\ E\ x_2 \in \mu_2\ \ \ldots ENTÃO\ padrao(x_1, x_2, \ldots)\ pertence\ a\ i \tag{4.9}$$

O *NEFCON* é dedicado a aplicações em sistemas de controle utilizando o modelo de Mandami. Kasabov e Song (2001) propuseram o *Denfis*, que consiste em um sistema capaz de evoluir de forma dinâmica adicionando novas regras ao *FIS* antes ou durante o processo de treinamento, e removendo regras durante ou após o processo de treinamento.

O *ANFIS* (Jang, 1993), é um sistema neuro-fuzzy aplicado na construção de modelos baseados em lógica nebulosa. Constitui um sistema híbrido que funciona como um *FIS*, o qual é gerado a partir do treinamento de uma rede neural adaptativa, e realizado a partir da propagação de sinal aplicada à rede adaptativa (Jang, 1993).

Uma rede neural adaptativa é uma rede multicamadas com propagação de informações e com funções de ativação. O resultado processado por cada função fornece o valor do neurônio de saída associado à camada correspondente. As características das funções de ativação podem variar e suas escolhas dependem do contexto onde o problema a ser resolvido está inserido (Jang, 1993).

Os neurônios adaptativos têm como característica a presença de um ou mais parâmetros que, juntamente com os valores de entrada, definem a sua função de ativação. O processo de treinamento de uma rede adaptativa consiste em determinar um valor para os parâmetros dos neurônios adaptativos, de forma a minimizar o máximo possível o erro entre um padrão de treinamento fornecido e o resultante da rede em treinamento.

O modelo *ANFIS*, será usado neste trabalho em comparação ao método de modelagem proposto que usa conceitos de conjuntos aproximados. Modelos *ANFIS* são utilizados usualmente em aplicações de previsão e aproximações de funções. O $MatLab^{®}$ possui um *toolbox* para o desenvolvimento de modelos *ANFIS* (Jang et al.1997). Existem diferentes arquiteturas, cada qual baseada nos modelos de regras fuzzy existentes (Jang e Sun, 1995). Neste trabalho, considerou-se a arquitetura que utiliza modelos com regras do tipo *Takagi-Sugeno*. A Figura 4.3 exemplifica, com o intuito de elucidação, a estrutura de uma arquitetura *ANFIS* composta por duas regras e que recebe duas variáveis de entrada (*x* e *y*), mapeando cada uma delas para dois conjuntos fuzzy distintos.

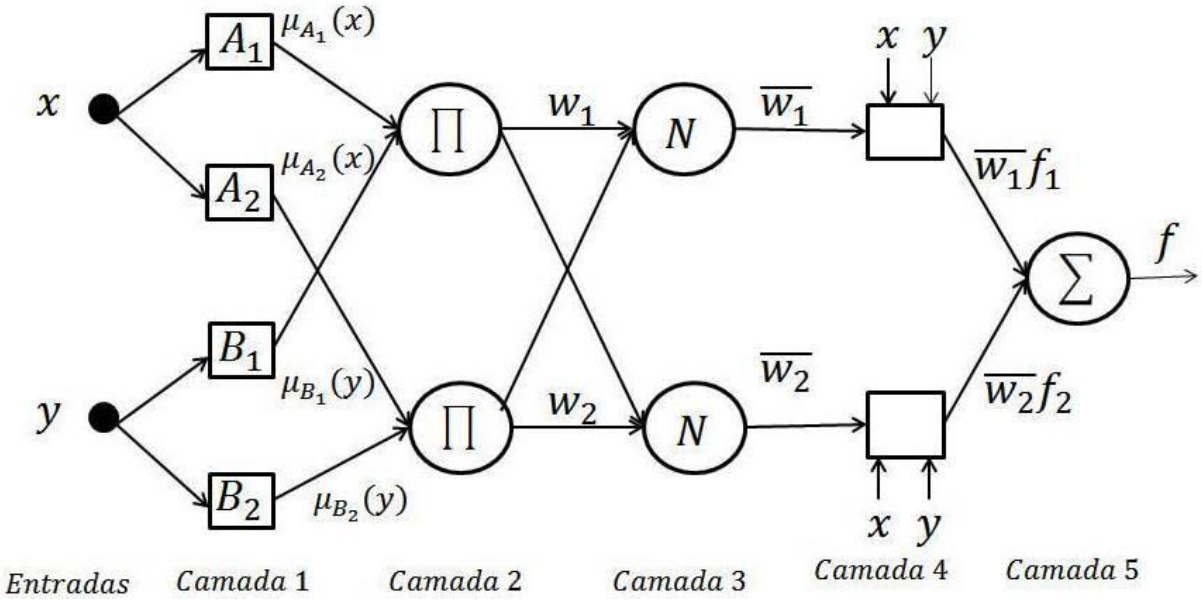

Figura 4.3 - Exemplo de estrutura *ANFIS* com duas entradas (Faustino, 2011).

O modelo recebe as variáveis de entrada e, através de interações em suas camadas, realizando as etapas de fuzificação e defuzificação, fornece ao final o valor de saída correspondente ao padrão de entrada. Cada camada da figura tem sua finalidade descrita a seguir.

- Camada 1: Ocorre o mapeamento das variáveis de entrada do universo de discurso para os conjuntos fuzzy, ou seja, é realizado o processo de *fuzificação* por meio da aplicação de uma função de pertinência. Os neurônios da primeira camada são do tipo adaptativo, sendo assim, os parâmetros, que nesse caso dizem respeito às constantes das funções de pertinência, serão estimados via processo de treinamento.

- Camada 2: Representa os antecedentes das regras fuzzy que compõem o *FIS*. Cada nó desta camada corresponde a uma regra. O valor resultante (w_i), onde i representa o índice dos neurônios dessa camada, é obtido por meio da aplicação de um operador T-Norma como o *Min*(A,B) ou *Prod*(A,B). Para o exemplo, usando o operador *Prod*, têm-se (4.9) e (4.10).

$$w_1 = \mu_{A_1}(x).\mu_{B_1}(x) \tag{4.9}$$

$$w_2 = \mu_{A_2}(y).\mu_{B_2}(y) \tag{4.10}$$

- Camada 3: É responsável pela normalização dos valores gerados na segunda camada. A saída dos neurônios que compõem essa camada indica o grau de ativação de uma regra em relação às outras. O valor de saída desses neurônios é dado pela equação (4.11).

$$\overline{w_i} = \frac{w_i}{\sum_{k=1}^{k} w_k} \tag{4.11}$$

Para o exemplo citado, vem (4.12) e (4.13).

$$\overline{w_1} = \frac{w_1}{w_1 + w_2} \tag{4.12}$$

$$\overline{w_2} = \frac{w_2}{w_1 + w_2} \tag{4.13}$$

- Camada 4: Nessa camada encontram-se as composições dos consequentes das regras *Takagi-Sugeno* cujo formato é dado por uma equação polinomial. No caso da Figura 4.3, tem-se a equação $(p_i x + q_i y + r_i)$. Os valores de saída dos neurônios dessa camada são calculados por meio da equação (4.14). Convém ressaltar que os neurônios dessa camada são do tipo adaptativo e, sendo assim, seus parâmetros serão determinados por meio do processo de treinamento. Nesse caso, os parâmetros a serem estimados correspondem aos valores de $p_i, q_i \; e \; r_i$.

$$f_i = \overline{w_i} g_i = \overline{w_i}(p_i x + q_i y + r_i) \tag{4.14}$$

- Camada 5: Finaliza o processo de *defuzificação* através da aplicação da equação (4.15) que consiste da soma ponderada dos valores obtidos na quarta camada.

$$z_i = \sum_i f_i = \sum_i \overline{w_i}\, g_i \tag{4.15}$$

O processo de treinamento do *ANFIS* consiste em se estimar os parâmetros das funções de pertinência encapsuladas nos neurônios da primeira camada e os parâmetros que compõem as funções polinomiais lineares que representam os consequentes das regras (representadas pelos neurônios da quarta camada). O processo de treinamento pode ser realizado adotando-se um dentre os três procedimentos que se seguem (Jang, 1993):

• *Backprogation:* Usa o método de *backpropagatio*n para ajustar os parâmetros dos neurônios por meio do cálculo do gradiente descendente para as camadas anteriores, ajustando-se os parâmetros dos nós adaptativos da primeira e quarta camadas;

• *Backpropagation + Método dos Mínimos Quadrados:* Usa o método de retropropagação para ajustar os parâmetros dos neurônios da primeira camada, e o método dos mínimos quadrados para ajustar os da quarta camada;

• *Backpropagation + Método dos Mínimos Quadrados* uma única vez: Usa o *backpropagation* para ajustar os parâmetros dos neurônios da primeira camada e o método dos mínimos quadrados para ajustar os da quarta camada somente na primeira interação.

Por ser o método empregado na obtenção de todos os modelos baseados em regras *fuzzy* desse trabalho, a próxima seção descreve como o algoritmo "*Backpropagation* (etapa"*backward*") e o *Método dos mínimos quadrados* (etapa"*forward*") são combinados com a finalidade de gerar o processo de treinamento híbrido de um modelo *ANFIS*.

4.3.1 Aplicação do Método dos Mínimos Quadrados no Processo de Treinamento (etapa *Forward*)

Considerando-se a estrutura do *ANFIS* representada pela Figura 4.3 pode-se definir a saída f_{out} do i-ésimo conjunto de treinamento, como uma combinação linear dos parâmetros dos consequentes das regras (p_i, q_i, r_i) como segue em (4.16) (Chen, Ying e Pan, 2010).

$$
\begin{aligned}
f_{out_i} &= \sum \overline{w_i} f_j \\
&= \overline{w_1} f_1 + \overline{w_2} f_2 \\
&= \overline{w_1}(p_1 x_i + q_1 y_i + r_1) + \overline{w_2}(p_2 x_i + q_2 y_i + r_2) \\
&= \overline{w_1}(p_1 x_i + q_1 y_i + r_1) + \overline{w_2}(p_2 x_i + q_2 y_i + r_2) \\
&= (\overline{w_1} x_i) p_1 + (\overline{w_1} y_i) q_1 + (\overline{w_1}) r_1 + (\overline{w_2} x_i) p_2 + (\overline{w_2} y_i) q_2 + (\overline{w_2}) r_2
\end{aligned}
\tag{4.16}
$$

$\langle (x_i, y_i), f_{outi} \rangle$, constitui o i-ésimo conjunto de treinamento. Sendo *n* o número total de pares de treinamento disponíveis, consideram-se as matrizes seguintes:

$$f = \begin{bmatrix} f_{out_1} \\ f_{out_2} \\ . \\ . \\ . \\ f_{out_n} \end{bmatrix}, \quad \theta = \begin{bmatrix} p_1 \\ q_1 \\ r_1 \\ p_2 \\ q_2 \\ r_2 \end{bmatrix} \quad e \quad B = \begin{bmatrix} \overline{w_1}x_1 & \overline{w_1}y_1 & \overline{w_1} & \overline{w_2}x_1 & \overline{w_2}y_1 & \overline{w_2} \\ \overline{w_1}x_2 & \overline{w_1}y_2 & \overline{w_1} & \overline{w_2}x_2 & \overline{w_2}y_2 & \overline{w_2} \\ : & : & : & : & : & : \\ . & . & . & . & . & . \\ \overline{w_1}x_n & \overline{w_1}y_n & \overline{w_1} & \overline{w_2}x_n & \overline{w_2}y_n & \overline{w_2} \end{bmatrix}.$$

Pode-se então, expressar a equação (4.16) em forma matricial por meio da equação (4.17). Onde θ é uma matriz desconhecida cujos elementos representam o conjunto de parâmetros dos consequentes das regras. Através do *Método dos Mínimos Quadrados*, determinam-se os parâmetros do modelo por meio da equação (4.18) (Denaï, Palis e Zeghbib, 2007).

$$f = B\theta \tag{4.17}$$

$$\theta^* = (B^T B)^{-1} B^T f \tag{4.18}$$

Após esta etapa de processamento dos dados do modelo *fuzzy*, conclui-se a etapa "*forward*" de treinamento. A rede neural é então utilizada para determinar os parâmetros dos antecedentes das regras de modelagem.

4.3.2 Aplicação do *Backpropagation* no Processo de Treinamento (etapa "*Backward*")

Denomina-se *E*, o erro médio quadrático obtido por meio da aplicação da equação (4.19).

$$E = \frac{1}{2}(d - f)^2 \tag{4.19}$$

Onde *d* é o resultado estimado e *f* é o valor esperado. Nesta etapa do treinamento, o erro *E* é propagado para as demais camadas do *ANFIS* e, com base nesse valor, os parâmetros dos nós adaptativos da primeira camada vão sendo ajustados.

Sendo assim, o valor de correção para um determinado parâmetro ρ de uma função de pertinência da variável *fuzzy* de entrada pode ser obtido por $\Delta\rho$, que por sua vez, é definido pela equação (4.20) (Mitra et al., 2004), onde η é a taxa de aprendizagem.

$$\Delta\rho = \eta \frac{\partial E}{\partial a_{ij}} = \eta \frac{\partial E}{\partial f} \frac{\partial f}{\partial f_i} \frac{\partial f_i}{\partial \omega_i} \frac{\partial \omega_i}{\partial \mu_{ij}} \frac{\partial \mu_{ij}}{\partial \rho} \tag{4.20}$$

Da equação (4.19) pode-se resolver $\frac{\partial E}{\partial f}$ conforme a equação (4.21).

$$\frac{\partial E}{\partial \rho} = f - d \tag{4.21}$$

Da equação (4.15) ($z_i = \sum_i^n f_i$), resolve-se o termo $\frac{\partial f}{\partial f_i}$ resultando na equação (4.22)

$$\frac{\partial f}{\partial f_i} = 1 \tag{4.22}$$

Sabe-se que $f_i = \frac{\omega_i}{\sum_{i=1}^n \omega_i}(p_i x + q_i y + r_i)$, portanto, o termo $\frac{\partial f_i}{\partial \omega_i}$ resulta na equação (4.23).

$$\frac{\partial f_i}{\partial \omega_i} = \frac{(p_i x + q_i y + r_i) - f_i}{\sum_{n=1}^1 \omega_i} \tag{4.23}$$

Da segunda camada, considerando que a T-Norma *Prod(A,B...)* seja o operador aplicado, tem-se $\omega_i = \prod_{j=1}^M \mu_{A_{ji}}$. Sendo assim, uma vez resolvido o termo $\frac{\partial \omega_i}{\partial \mu_{ij}}$ obtém-se a equação (4.24).

$$\frac{\partial \omega_i}{\partial \mu_{ij}} = \frac{\omega_i}{\mu_{ij}} \tag{4.24}$$

Por fim a, tem-se a última derivada parcial da função de pertinência propriamente dita, representada pelo primeiro termo da equação (4.20). Considerando uma função trapezoidal, como aquela representada pela equação (4.6), pode-se estimar o valor dos parâmetros *a*, *b* e *c* após resolver as equações (4.25), (4.26) e (4.27).

$$\frac{\partial \mu_{ij}}{\partial a} = -\frac{\mu_{ij}(x_i)^2}{a}\left(\frac{(x_i - c)^2}{a}\right)^b \tag{4.25}$$

$$\frac{\partial \mu_{ij}}{\partial b} = -b\mu_{ij}(x_i)^2 \left(\frac{(x_i - c)^2}{a}\right)^{b-1} \tag{4.26}$$

$$\frac{\partial \mu_{ij}}{\partial c} = \frac{2b\mu_{ij}(x_i)^2}{x_i - c}\left(\frac{(x_i - c)^2}{a}\right)^{b} \tag{4.27}$$

O procedimento anteriormente descrito ocorre para todos os pares de dados disponíveis para treinamento. Uma vez que todos os pares de dados de treinamento são apresentados, uma interação ou época é caracterizada. O processo de treinamento segue até que o número de épocas determinada a priori seja atingido ou até que um erro objetivo, dado pela equação (4.19), calculado no final de cada interação seja alcançado.

Exemplo de Aplicação do *ANFIS* na Previsão de uma Série Temporal Caótica.

Será apresentado um exemplo do *toolbox ANFIS* do $MatLab^{®}$ na obtenção de um modelo *fuzzy* de uma série temporal caótica (foi escolhido essa série por apresentar não linearidades e comportamento caótico dependendo das condições iniciais do modelo). A equação (4.28) ilustra o modelo matemático de uma série de Mackey-Glass.

$$\dot{x}(t) = \frac{a.x(t-\tau)}{b + x^h(t-\tau)} - c.x(t) \tag{4.28}$$

Considerando os parâmetros $a = 0{,}2;\ b = 1;\ c = 0{,}1$ e $h = 10$ para a série, vem (4.29).

$$\dot{x}(t) = \frac{0{,}2x(t-\tau)}{1+x^{10}(t-\tau)} - 0{,}1x(t). \tag{4.29}$$

Esta série apresenta um comportamento caótico dependente dos valores do atraso de tempo (τ) e da condição inicial ($x(0)$) do modelo. Como o comportamento resultante da série é caótico, a mesma constitui um exemplo usado como referência para modelagem de sistemas complexos. Considerando os valores: $x(0) = 1{,}2$ e $\tau = 17$, a Figura 4.4 ilustra o resultado da simulação da série para os valores considerados. Em sequência encontra-se o código em MatLab do programa de obtenção do modelo *fuzzy* correspondente.

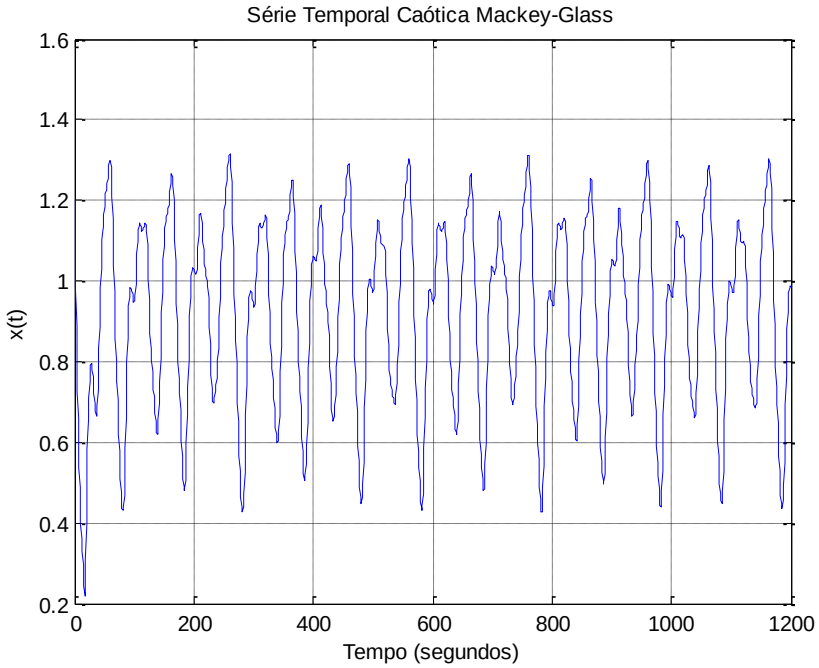

Figura 4.4 - Dados da série temporal caótica de Mackey-Glass.

```
%Previsão da Série temporal caótica da série de Mackey-Glass
%Exemplo de modelo ANFIS


load mgdata.dat
time = mgdata(:, 1); x = mgdata(:, 2);
figure(1), plot(time, x);
title('Série Temporal Caótica de Mackey Glass')
xlabel('Tempo (segundos)')
for t=118:1117,
Data(t-117,:)=[x(t-18) x(t-12) x(t-6) x(t) x(t+6)];
end
trnData=Data(1:500, :);
chkData=Data(501:end, :);
fismat = genfis1(trnData);

figure(2)
subplot(2,2,1)
plotmf(fismat, 'input', 1)
subplot(2,2,2)
plotmf(fismat, 'input', 2)
subplot(2,2,3)
plotmf(fismat, 'input', 3)
subplot(2,2,4)
plotmf(fismat, 'input', 4)

[fismat1,error1,ss,fismat2,error2] = ...
      anfis(trnData,fismat,[],[],chkData);
figure(3)
subplot(2,2,1)
plotmf(fismat2, 'input', 1)
subplot(2,2,2)
plotmf(fismat2, 'input', 2)
```

```
subplot(2,2,3)
plotmf(fismat2, 'input', 3)
subplot(2,2,4)
plotmf(fismat2, 'input', 4)
figure(4)
plot([error1 error2]);grid on
hold on; plot([error1 error2], 'o');
xlabel('Épocas');
ylabel('Erro médio quadrático');
title('Curva de Erros');
figure(5)
anfis_output = evalfis([trnData(:,1:4); chkData(:,1:4)], ...
    fismat2);
index = 125:1124;
subplot(211), plot(time(index), [x(index) anfis_output]);grid on
xlabel('Tempo (segundos)');
title('Série Original e a prevista através do ANFIS');
subplot(212), plot(time(index), x(index) - anfis_output);grid on
xlabel('Tempo (segundos)');
title('Erros Previstos');
```

Em previsão de séries temporais, necessita-se utilizar valores conhecidos da série no tempo *t*, para poder prever valores futuros, tal como $t + P$. O método padrão utilizado para este tipo de previsão é criar um mapeamento a partir de *D* pontos de dados de amostragem, para todas Δ de unidade de tempo amostradas, tal como $(x(t-(D-1)\Delta), \dots, x(t-\Delta), \dots, x(t))$ para um valor futuro previsto de $x(t+P)$. Seguindo as definições convencionais para a previsão de séries temporais consideremos $D = 4$, $\Delta = P = 6$. Para cada t, os dados de entrada de treinamento para o *ANFIS* é um vetor de quatro dimensões da seguinte forma:

$$w(t) = [x(t-18), x(t-12), x(t-6), x(t)].$$

Os dados de treinamento de saída correspondem à previsão da trajetória dada por:

$$s(t) = x(t+6).$$

Para valores de t variando no intervalo [118, 1117], os dados de treinamento do modelo é uma estrutura cujos primeiros quatro vetores são de entrada e o último é o de saída. Dessa base de dados, uma parte dos mesmos forma o conjunto de treinamento (*trnData*) do *ANFIS*, enquanto que outra parte é usada como verificação ou validação (*chkData*) do modelo identificado.

Para começar o treinamento é necessário especificar a estrutura e os parâmetros iniciais para o processo de aprendizagem. A função *genfis1* trabalha com esse tipo de especificação. Se não for especificado o número e nem o tipo da função de pertinência que será utilizada no modelo, determinados valores padrões serão assumidos. O modelo considerado possui duas funções de pertinência para cada uma das quatro entradas do sistema.

O modelo fuzzy resultante contém dezesseis regras. Para alcançar uma boa capacidade de generalização, é importante que o número de dados usados para treinamento seja maior que o número de parâmetros estimados. A função *genfis1* gera inicialmente funções de pertinência igualmente espaçadas de modo a cobrir todo o universo de discurso da entrada considerada (Figura 4.5).

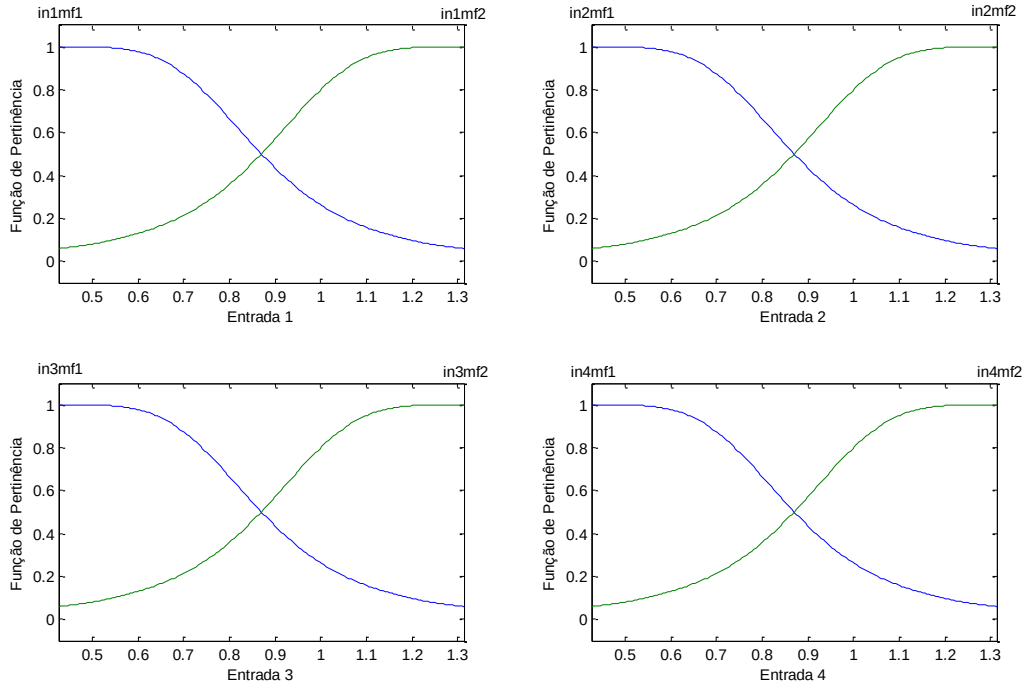

Figura 4.5 - Funções de pertinência igualmente espaçadas.

Para iniciar o treinamento é utilizado o comando ilustrado abaixo.

$$[fismat1, error1, ss, fismat2, error2] = \cdots$$

$$anfis(trnData, fismat, [\,], [\,], chkData);$$

A Figura 4.6 ilustra as funções de pertinência obtidas após o treinamento. A Figura 4.7 contém as informações dos erros obtidos na etapa de treinamento (em cor azul) e com os dados de validação (em cor verde). O primeiro gráfico da Figura 4.8 mostra a comparação entre os dados originais da série (em cor verde) e os previstos pelo modelo *fuzzy* resultante (em cor azul), e o segundo gráfico da Figura 4.8, ilustra os erros instantâneos resultantes. O treinamento foi com apenas dez épocas, e se fosse ampliado para um valor maior poderia resultar em uma melhor exatidão na modelagem.

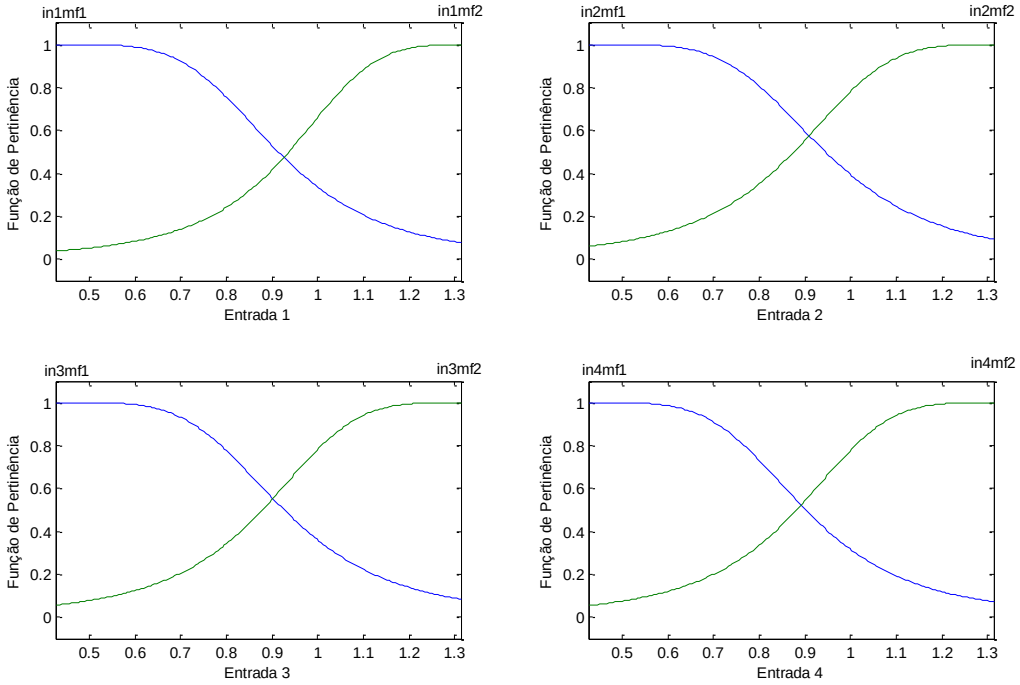

Figura 4.6 - Função de pertinência resultantes do treinamento.

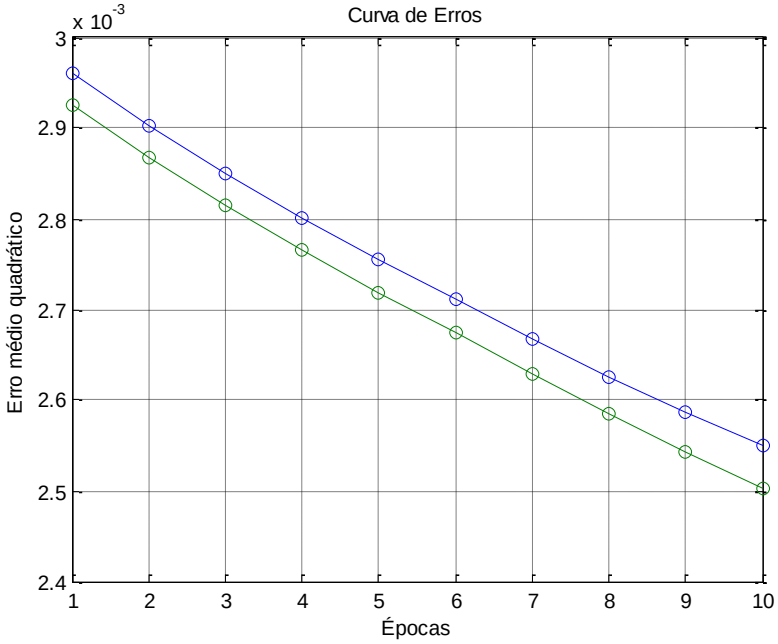

Figura 4.7 - Erro médio quadrático.

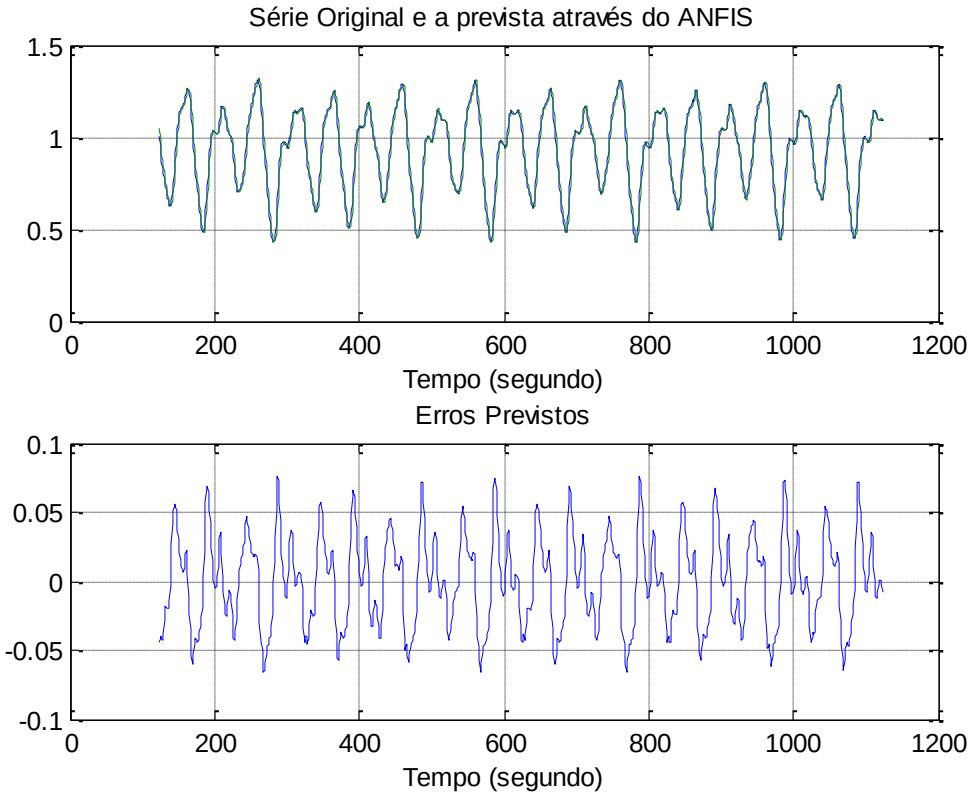

Figura 4.8 - Comparação entre a série original e a prevista pelo modelo fuzzy.

CAPÍTULO 5

5 PROPOSTA DE MODELAGEM APROXIMADA DE SISTEMAS MIMO

O objetivo deste capítulo é apresentar uma proposta para construção de modelos baseados em regras para sistemas com múltiplas variáveis de entrada e de saída (MIMO – Multi Input Multi Output), na qual se emprega conceitos de conjuntos aproximados.

5.1 Introdução

Este capítulo, apresenta uma proposta para construção de modelos baseados em regras para sistemas MIMO que emprega conceitos da teoria dos conjuntos aproximados. O objetivo é a obtenção de modelos (estáticos ou dinâmicos) referentes a sistemas lineares e não lineares. O método proposto consiste em agrupar os dados relativos à modelagem de sistemas MIMO em sistemas de informações correspondentes, onde os mesmos englobam eventuais acoplamentos nas variáveis intrínsecas a estes modelos.

5.2 Metodologia

Objetivando uma representação mais adequada para as aplicações numéricas, será adotada a forma ilustrada na Tabela 5.1 para os sistemas de informação abordados nesse trabalho.

Tabela 5.1 - Representação Tabular Numérica de um *SI*.

x_1	x_2	x_3	⋯	x_N	y
${x_1}^{(1)}$	${x_2}^{(1)}$	${x_3}^{(1)}$	…	${x_N}^{(1)}$	$y^{(1)}$
${x_1}^{(2)}$	${x_2}^{(2)}$	${x_3}^{(2)}$	…	${x_N}^{(2)}$	$y^{(2)}$
…	…	…	…	…	…
${x_1}^{(k)}$	${x_2}^{(k)}$	${x_3}^{(k)}$	…	${x_N}^{(k)}$	$y^{(k)}$
…	…	…	…	…	…
${x_1}^{(m)}$	${x_2}^{(m)}$	${x_3}^{(m)}$	…	${x_N}^{(m)}$	$y^{(m)}$
…	…	…	…	…	…
${x_1}^{(v)}$	${x_2}^{(v)}$	${x_3}^{(v)}$	…	${x_N}^{(v)}$	$y^{(v)}$

Os atributos de condições ($\boldsymbol{x_i}$) serão variáveis cujos dados ($x_N{}^{(k)}$) estarão relacionados a funções gerais definidas pelos vetores ($y^{(k)}$) do atributo de decisão ($\boldsymbol{y}$). Do *SI* em questão derivam as sentenças (5.1). Por exemplo, para $x_1 = x_1{}^{(k)}, x_2 = x_2{}^{(k)}, x_3 = x_3{}^{(k)}$ e $x_N = x_N{}^{(k)}$ tem-se $y = y^{(k)}$ expresso por s_k.. Para $x_1 = x_1{}^{(m)}$, $x_2 = x_2{}^{(m)}$, $x_3 = x_3{}^{(m)}$, e $x_N = x_N{}^{(m)}$ tem-se $y = y^{(m)}$ definido por s_m.

$$
\begin{aligned}
&s_1: IF\ x_1 = x_1{}^{(1)}\ AND\ x_2 = x_2{}^{(1)}\ AND\ \cdots AND\ x_N = x_N{}^{(1)}\ THEN\ y = y^{(1)};\\
&s_2: IF\ x_1 = x_1{}^{(2)}\ AND\ x_2 = x_2{}^{(2)}\ AND\ \cdots AND\ x_N = x_N{}^{(2)}\ THEN\ y = y^{(2)};\\
&\qquad\qquad \cdots \qquad\qquad \cdots \qquad\qquad \cdots\\
&s_k: IF\ x_1 = x_1{}^{(k)}\ AND\ x_2 = \ x_2{}^{(k)}\ AND\ \cdots AND\ x_N = x_N{}^{(k)}\ THEN\ y = y^{(k)};\\
&\qquad\qquad \cdots \qquad\qquad \cdots \qquad\qquad \cdots\\
&s_m: IF\ x_1 = x_1{}^{(m)} AND\ x_2 = \ x_2{}^{(m)}\ AND\ \cdots AND\ x_N = x_N{}^{(m)}\ THEN\ y = y^{(m)};\\
&\qquad\qquad \cdots \qquad\qquad \cdots \qquad\qquad \cdots\\
&s_v: IF\ x_1 = x_1{}^{(v)}\ AND\ x_2 = \ x_2{}^{(v)}\ AND\ \cdots AND\ x_N = x_N{}^{(v)}\ THEN\ y = y^{(v)}
\end{aligned}
\tag{5.1}
$$

Considerando-se valores intermediários aos dados representados na Tabela 5.1, ou seja, $x_1{}^{(k)} \leq x_1 \leq x_1{}^{(m)}, x_2{}^{(k)} \leq x_2 \leq x_2{}^{(m)}$, $x_3{}^{(k)} \leq x_3 \leq x_3{}^{(m)}$ e $x_N{}^{(k)} \leq x_N \leq x_N{}^{(m)}$, as sentenças acima podem ser representadas pela combinação de s_k *e* s_m por meio da regra genérica (5.2), ou na forma simplificada (5.3).

$$
\begin{aligned}
&r_i: IF\ x_1{}^{(k)} \leq x_1 \leq x_1{}^{(m)} AND\ x_2{}^{(k)} \leq x_2 \leq x_2{}^{(m)} AND \cdots AND\ x_N{}^{(k)} \leq x_N \leq x_N{}^{(m)}\\
&THEN\ \min\left\{y^{(k)}, \cdots, y^{(m)}\right\} \leq y \leq \max\left\{y^{(k)}, \cdots, y^{(m)}\right\}
\end{aligned}
\tag{5.2}
$$

$$
r_i: \text{IF}\ x_1 = \alpha^{(g)}\ AND\ \ x_2 = \beta^{(g)}\ AND\ \cdots AND\ \ x_N = \gamma^{(g)}\ \text{THEN Y} = \sigma^{(g)} \tag{5.3}
$$

Onde $\alpha^{(g)} = \left[x_1{}^{(k)}, x_1{}^{(m)}\right]$, $\beta^{(g)} = \left[x_2{}^{(k)}, x_2{}^{(m)}\right]$, $\gamma^{(g)} = \left[x_N{}^{(k)}, x_N{}^{(m)}\right]$ e $\sigma^{(g)} = \left[y^{(k)}, y^{(m)}\right]$, considerando que $y^{(k)} < y^{(m)}$.

A obtenção das regras é realizada por meio das expressões (3.8), (3.9), (3.10) e (3.11), conforme ilustrado no exemplo da página 25. O processamento das expressões em questão pode ser realizado em linguagens de programação genéricas.

Dessa forma, caso o *SI* considerado for composto por um único atributo condicional (x_1) de valores numéricos, a equação de interpolação linear (5.4) pode ser aplicada para estimar os valores intermediários correspondentes.

$$y = y^{(k)} + (y^{(m)} - y^{(k)})\left(\frac{(x_1 - {x_1}^{(k)})}{({x_1}^{(m)} - {x_1}^{(k)})}\right) \tag{5.4}$$

No caso em que o *SI* for composto por dois atributos condicionais $(x_1\ e\ x_2)$ de valores numéricos, a equação (5.5), caso particular de (5.6) para *N* = 2, pode ser aplicada para determinar os valores intermediários correspondentes (Pinheiro et al. 2010_a, 2010_b).

$$y = y^{(k)} + \frac{(y^{(m)} - y^{(k)})}{2}\left(\frac{(x_1 - {x_1}^{(k)})}{({x_1}^{(m)} - {x_1}^{(k)})} + \frac{(x_2 - {x_2}^{(k)})}{({x_2}^{(m)} - {x_2}^{(k)})}\right) \tag{5.5}$$

Para modelos com *n* variáveis de saída, são definidos *n* sistemas de informações correspondentes, onde cada variável de decisão do *SI* associado corresponde a uma variável de saída do modelo em questão.

Para um *SI* composto por um número *n* de atributos condicionais numéricos, a forma (5.6) constitui uma expressão que produz valores iguais à fórmula de interpolação linear de Lagrange, sendo que o índice *j* simboliza saídas adicionais do modelo.

$$y_j = (x_n, x_n^{(i)}, y_j^{(i)})_{\substack{i=k,m \\ n=1,N}} = {y_j}^{(k)} + \frac{({y_j}^{(m)} - {y_j}^{(k)})}{N}\sum_{n=1}^{N}\frac{(x_n - x_n^{(k)})}{(x_n^{(m)} - x_n^{(k)})} \tag{5.6}$$

Uma justificativa teórica referente à capacidade de aproximação de funções da abordagem proposta via regras aproximadas está ilustrada no Anexo E.

A Figura 5.1 ilustra o procedimento para obtenção e computação de modelos baseados em regras como proposto por Pinheiro et al. 2010_a, 2010_b, usado por Faustino, 2011, e que também será utilizada nesse trabalho. A definição do sistema de informação consiste em dados numéricos das informações de entradas e saídas do sistema a se modelar. A

discretização dos dados pode ser realizada de forma manual ou utilizando determinados procedimentos conforme a ferramenta computacional utilizada. A proposição de modelos baseados em regras (obtidos via processamento de conjuntos aproximados) não constitui um problema computacional em relação à obtenção dos redutos na geração de regras de modelagem. A explicação é simples, pois todas as variáveis numéricas (de um determinado *SI*) que definem a estrutura de um determinado modelo constituem os próprios redutos (que não precisam ser calculados) que definem as regras de modelagem. A definição de quais variáveis e quantas de suas amostras passadas constituem a estrutura de um determinado modelo, pode ser obtida por métodos conhecidos (por exemplo, com a avaliação da correlação de subconjuntos de dados que constituem as medições do sistema a ser modelado).

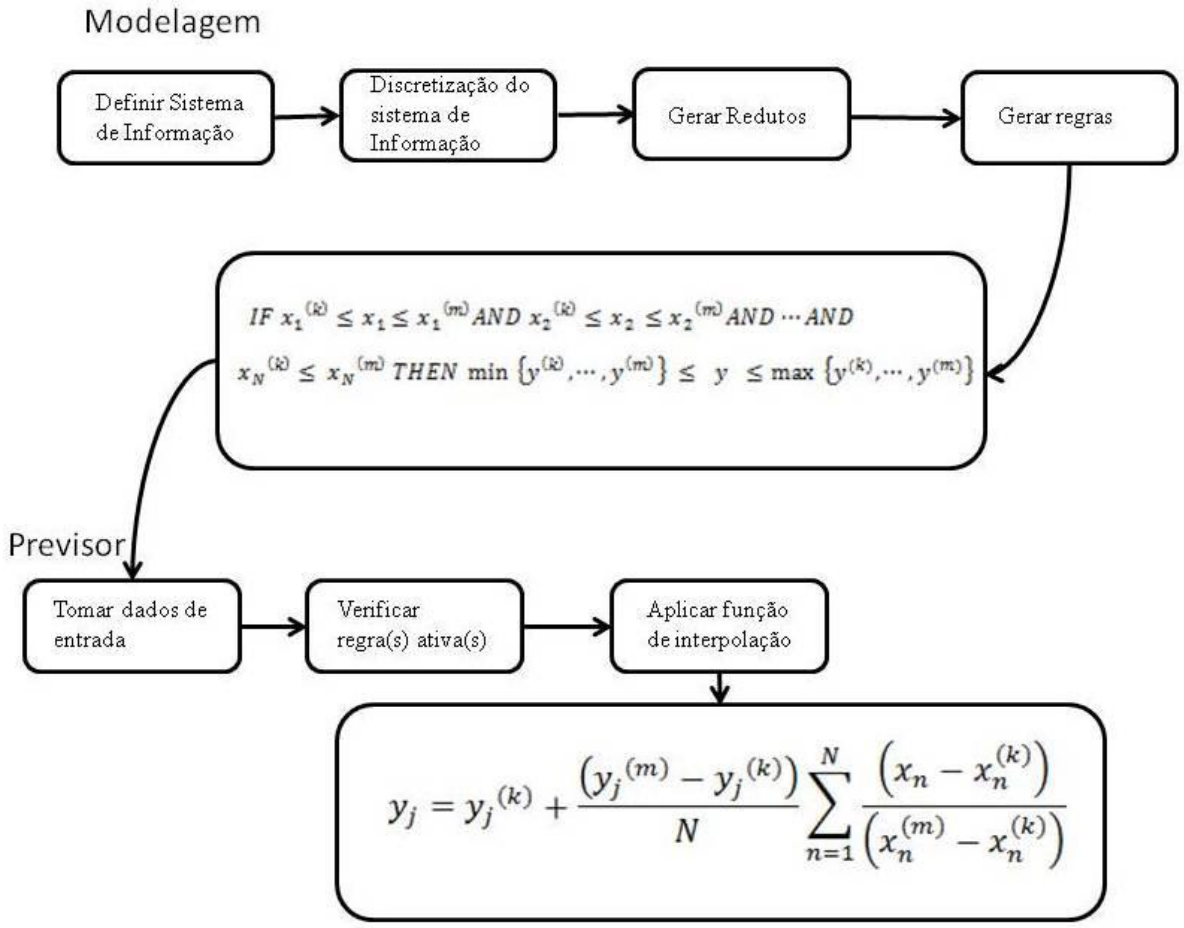

Figura 5.1 - Esquema geral da metodologia proposta (Faustino, 2011).

A geração das regras de decisão correspondentes é facilitada através da utilização de ferramentas computacionais específicas. Existem ferramentas computacionais (várias de acesso gratuito) desenvolvidas especificamente para o processamento de conjuntos aproximados: RSL (Rough Sets Library) desenvolvida por M. Gawrys e J. Sienkiewicz da

Universidade Warsaw de Tecnologia, Polônia; Rough Enough desenvolvida por Anders Torvill Bjorvand; CI (Column Importance facility) e Rosetta (Ohrn e Komorowski, 1997). Neste trabalho será empregada a ferramenta Rosetta para o processamento dos dados relacionados a sistemas de informação em geral. Esta ferramenta é de utilização simples, tem acesso gratuito (http://www.idi.ntnu.no/~aleks/rosetta/), e atende às necessidades da proposta neste trabalho. Um breve tutorial sobre a utilização da ferramenta Rosetta encontra-se no Anexo A, no qual é ilustrada uma modelagem de uma função não linear.

A obtenção da representação (5.2) é realizada através de uma interface que está ilustrada no Anexo B. A computação das regras resultantes e da expressão de interpolação (5.6) está presente no Anexo C, referentes aos exemplos a serem apresentados no Capítulo 6.

Os modelos resultantes neste trabalho serão denominados a partir de agora de modelos aproximados. Os exemplos numéricos a seguir ilustrarão a metodologia adotada. Inicialmente serão exemplificadas funções estáticas lineares e não lineares. No Capítulo 6 sistemas dinâmicos não lineares serão abordados, incluindo o exemplo de um processo real.

5.3 Exemplos

Exemplo 5.3.1

Este exemplo ilustra a modelagem, por intermédio de modelos baseados em regras, de duas funções estáticas não lineares com uma variável de entrada e duas de saída. A Tabela 5.2 ilustra os dados das funções $y_1 = \sin(x_1)$, $y_2 = \sqrt{x_1}$ com $x_1 \in [0, \frac{\pi}{2}]$.

A variável independente (x_1), que é o atributo de condição do *SI* correspondente, possui valores fracionários que serão quantizados neste exemplo em três faixas igualmente espaçadas: $\alpha^{(1)} = [0,\ 0.5236]$; $\alpha^{(2)} = [0.5236,\ 1.0472]$; $\alpha^{(3)} = [1.0472,\ 1.5708]$. Neste exemplo temos um único atributo de condição (x_1) que constitui o próprio reduto do *SI* correspondente, e dois atributos de decisão (y_1 e y_2). Assim, as regras de decisão relativas às funções originais ficam expressas por (5.7).

Tabela 5.2 - Dados referentes ao Exemplo 1.

x_1	y_1	y_2
0.0000	$y_1^{(a)} = 0.0000$	$y_2^{(a)} = 0.0000$
0.2618	$y_1^{(b)} = 0.2588$	$y_2^{(b)} = 0.5117$
0.5236	$y_1^{(c)} = 0.5000$	$y_2^{(c)} = 0.7236$
0.7854	$y_1^{(d)} = 0.7071$	$y_2^{(d)} = 0.8862$
1.0472	$y_1^{(e)} = 0.8660$	$y_2^{(e)} = 1.0233$
1.3090	$y_1^{(f)} = 0.9659$	$y_2^{(f)} = 1.1441$
1.5708	$y_1^{(g)} = 1.0000$	$y_2^{(g)} = 1.2533$

IF $x_1 = \alpha^{(1)}$ THEN $y_1 = y_1^{(a)}$ OR $y_1 = y_1^{(b)}$ OR $y_1 = y_1^{(c)}$; $y_2 = y_2^{(a)}$ OR $y_2 = y_2^{(b)}$ OR $y_2 = y_2^{(c)}$;

IF $x_1 = \alpha^{(2)}$ THEN $y_1 = y_1^{(c)}$ OR $y_1 = y_1^{(d)}$ OR $y_1 = y_1^{(e)}$; $y_2 = y_2^{(c)}$ OR $y_2 = y_2^{(d)}$ OR $y_2 = y_2^{(e)}$;

IF $x_1 = \alpha^{(3)}$ THEN $y_1 = y_1^{(e)}$ OR $y_1 = y_1^{(f)}$ OR $y_1 = y_1^{(g)}$; $y_2 = y_2^{(e)}$ OR $y_2 = y_2^{(f)}$ OR $y_2 = y_2^{(g)}$

(5.7)

Usando a forma (5.3), pode-se escrever o modelo baseado em regras (5.8), onde: $\sigma_1^{(1)} = [0, 0.5]$; $\sigma_1^{(2)} = [0.5, 0.866]$; $\sigma_1^{(3)} = [0.866, 1]$; $\sigma_2^{(1)} = [0.0,7236]$; $\sigma_2^{(2)} = [0.7236, 1.0233]$; $\sigma_2^{(3)} = [1.0233, 1.2533]$.

IF $x_1 = \alpha^{(1)}$ THEN $y_1 = \sigma_1^{(1)}$; $y_2 = \sigma_2^{(1)}$;

IF $x_1 = \alpha^{(2)}$ THEN $y_1 = \sigma_1^{(2)}$; $y_2 = \sigma_2^{(2)}$;

IF $x_1 = \alpha^{(3)}$ THEN $y_1 = \sigma_1^{(3)}$; $y_2 = \sigma_2^{(3)}$ (5.8)

Para estimar valores intermediaries deste modelo baseado em regras será utilizada a fórmula de interpolação linear proposta em (5.4)

Exemplificando: para $x_1 = 0.5$ tem-se $y_1 = 0.4775$ e $y_2 = 0.691$. Para $x_1 = 1.2$ tem-se $y_1 = 0.9051$ e $y_2 = 1.0904$. Se mais de uma regra resultar em valores estimados para

uma determinada variável de saída (por exemplo, para dados nas extremidades das faixas dos atributos de condição), o valor resultante é dado pela média aritmética dos valores estimados.

A fim de verificar o efeito do nível de discretização na modelagem do *SI*, dobrou-se as amostras da tabela de informação (5.2) que resultou nos valores representados na Tabela 5.3.

Tabela 5.3 - Dados referentes ao Exemplo 1 com uma maior discretização dos dados.

x_1	y_1	y_2
0.0000	$y_1^{(a)} = 0.0000$	$y_2^{(a)} = 0.0000$
0.0873	$y_1^{(b)} = 0.0872$	$y_2^{(b)} = 0.2955$
0.1745	$y_1^{(c)} = 0.1736$	$y_2^{(c)} = 0.4177$
0.2618	$y_1^{(d)} = 0.2588$	$y_2^{(d)} = 0.5117$
0.3491	$y_1^{(e)} = 0.3421$	$y_2^{(e)} = 0.5908$
0.4363	$y_1^{(f)} = 0.4226$	$y_2^{(f)} = 0.6605$
0.5236	$y_1^{(g)} = 0.5000$	$y_2^{(g)} = 0.7236$
0.6109	$y_1^{(h)} = 0.5736$	$y_2^{(h)} = 0.7816$
0.6981	$y_1^{(i)} = 0.6428$	$y_2^{(i)} = 0.8355$
0.7854	$y_1^{(j)} = 0.7071$	$y_2^{(j)} = 0.8862$
0.8727	$y_1^{(l)} = 0.7661$	$y_2^{(l)} = 0.9342$
0.9599	$y_1^{(m)} = 0.8191$	$y_2^{(m)} = 0.9797$
1.0472	$y_1^{(n)} = 0.8660$	$y_2^{(n)} = 1.0233$
1.1345	$y_1^{(o)} = 0.9063$	$y_2^{(o)} = 1.0651$
1.2217	$y_1^{(p)} = 0.9397$	$y_2^{(p)} = 1.1053$
1.3091	$y_1^{(q)} = 0.9660$	$y_2^{(q)} = 1.1442$
1.3963	$y_1^{(r)} = 0.9848$	$y_2^{(r)} = 1.1817$
1.4835	$y_1^{(s)} = 0.9962$	$y_2^{(s)} = 1.2180$
1.5707	$y_1^{(t)} = 1.0000$	$y_2^{(t)} = 1.2533$

Quantizando a variável independente (x_1) em seis faixas igualmente espaçadas, vem: $\alpha^{(1)} = [0,\ 0.2618]$; $\alpha^{(2)} = [0.2618,\ 0.5236]$; $\alpha^{(3)} = [0.5236,\ 0.7854]$; $\alpha^{(4)} = [0.7854,\ 1.0472]$; $\alpha^{(5)} = [1.0472,\ 1.3091]$; $\alpha^{(6)} = [1.3091,\ 1.5707]$.

Assim, as regras de decisão relativas às funções ficam expressas por (5.9).

IF $x_1 = \alpha^{(1)}$ THEN $y_1 = y_1^{(a)}$ OR… OR $y_1 = y_1^{(d)}$; $y_2 = y_2^{(a)}$ OR… OR $y_2 = y_2^{(d)}$;

IF $x_1 = \alpha^{(2)}$ THEN $y_1 = y_1^{(d)}$ OR… OR $y_1 = y_1^{(g)}$; $y_2 = y_2^{(d)}$ OR… OR $y_2 = y_2^{(g)}$;

IF $x_1 = \alpha^{(3)}$ THEN $y_1 = y_1^{(g)}$ OR… OR $y_1 = y_1^{(j)}$; $y_2 = y_2^{(g)}$ OR… OR $y_2 = y_2^{(j)}$;

IF $x_1 = \alpha^{(4)}$ THEN $y_1 = y_1^{(j)}$ OR… OR $y_1 = y_1^{(n)}$; $y_2 = y_2^{(j)}$ OR… OR $y_2 = y_2^{(n)}$;

IF $x_1 = \alpha^{(5)}$ THEN $y_1 = y_1^{(n)}$ OR… OR $y_1 = y_1^{(q)}$; $y_2 = y_2^{(n)}$ OR… OR $y_2 = y_2^{(q)}$;

IF $x_1 = \alpha^{(6)}$ THEN $y_1 = y_1^{(q)}$ OR… OR $y_1 = y_1^{(t)}$; $y_2 = y_2^{(q)}$ OR… OR $y_2 = y_2^{(t)}$;

(5.9)

Usando a forma (5.3), pode-se escrever o modelo baseado em regras (5.10), onde: $\sigma_1^{(1)} = [0,\ 0.2588]$; $\sigma_1^{(2)} = [0.2588,\ 0.5000]$; $\sigma_1^{(3)} = [0.5000,\ 0.7071]$; $\sigma_1^{(4)} = [0.7071,\ 0.8660]$; $\sigma_1^{(5)} = [0.8660, 0.9660]$; $\sigma_1^{(6)} = [0.9660, 1.0000]$; $\sigma_2^{(1)} = [0,\ 0.5117]$; $\sigma_2^{(2)} = [0.5117,\ 0.7236]$; $\sigma_2^{(3)} = [0.7236, 0.8862]$; $\sigma^{(4)} = [0.8862, 1.0233]$; $\sigma^{(5)} = [1.0233, 1.1442]$; $\sigma^{(6)} = [1.1442, 1.2533]$.

IF $x_1 = \alpha^{(1)}$ THEN $y_1 = \sigma_1^{(1)}$; $y_2 = \sigma_2^{(1)}$;

IF $x_1 = \alpha^{(2)}$ THEN $y_1 = \sigma_1^{(2)}$; $y_2 = \sigma_2^{(2)}$;

IF $x_1 = \alpha^{(3)}$ THEN $y_1 = \sigma_1^{(3)}$; $y_2 = \sigma_2^{(3)}$;

IF $x_1 = \alpha^{(4)}$ THEN $y_1 = \sigma_1^{(4)}$; $y_2 = \sigma_2^{(4)}$;

IF $x_1 = \alpha^{(5)}$ THEN $y_1 = \sigma_1^{(5)}$; $y_2 = \sigma_2^{(5)}$;

IF $x_1 = \alpha^{(6)}$ THEN $y_1 = \sigma_1^{(6)}$; $y_2 = \sigma_2^{(6)}$;

(5.10)

Para estimar valores intermediários deste modelo baseado em regras será utilizada a fórmula de interpolação definida em (5.4). Exemplificando: para $x_1 = 0.5$ tem-se

$y_1 = 0.4782$ e $y_2 = 0.7045$. Para $x_1 = 1.2$ tem-se $y_1 = 0.9243$ e $y_2 = 1.0938$. Para efeito de comparação, alguns valores reais das funções originais são mostrados na Tabela 5.4 juntamente com os valores estimados pelas regras de modelagem.

Valores exatos das funções originais:

$$x_1 = 0.5 \begin{cases} y_1 = 0.4794 \\ y_2 = 0.7071 \end{cases} \quad ; x_1 = 1.2 \begin{cases} y_1 = 0.9320 \\ y_2 = 1.0954 \end{cases}$$

Tabela 5.4 - Erros entre os dados das funções originais e os dados estimados pelas regras de modelagem.

Discretização	$x_1 = 0.5$	Erro (%)	$x_1 = 1.2$	Erro (%)
3 faixas	$y_1 = 0.4775$ $y_2 = 0.6910$	0,40 2,28	$y_1 = 0.9051$ $y_2 = 1.0904$	2,89 0,46
6 faixas	$y_1 = 0.4782$ $y_2 = 0.7045$	0,25 0,37	$y_1 = 0.9243$ $y_2 = 1.0938$	0,83 0,15

Exemplo 5.3.2

Este exemplo ilustra a modelagem de duas funções lineares estáticas com duas variáveis de entrada e duas de saída. A Tabela 5.5 ilustra os dados das funções $y_1 = x_1 + x_2$ e $y_2 = x_1 - x_2$, onde x_1 e $x_2 \in [0, 0.5]$.

Tabela 5.5 - Dados referentes ao Exemplo2.

x_1	x_2	y_1	y_2
0	0	0	0
0	0.5	0.5	−0.5
0.5	0	0.5	0.5
0.5	0.5	1	0

O *SI* associado à tabela possui dois atributos de condição (x_1 e x_2), definindo o reduto do *SI* correspondente, e dois atributos de decisão (y_1 e y_2). Para valores das variáveis independentes na faixa [0, 0.5] é possível escrever a regra para a variável dependente y_1:

IF x_1 = [0, 0.5] AND x_2 = [0, 0.5] THEN y_1 = [0, 1].

Para estimar valores em faixas de valores intermediários, utiliza-se a expressão de interpolação linear (5.5), que é o caso particular de (5.6) para $N = 2$.

$$y_j = y_j{}^{(k)} + \frac{(y_j{}^{(m)} - y_j{}^{(k)})}{2}\left(\frac{(x_1 - x_1{}^{(k)})}{(x_1{}^{(m)} - x_1{}^{(k)})} + \frac{(x_2 - x_2{}^{(k)})}{(x_2{}^{(m)} - x_2{}^{(k)})}\right)$$

Exemplificando, para $x_1 = 0.4$ e $x_2 = 0.3$, tem-se y_1 = 0 + (1 – 0)/2*((0.4 – 0)/(0.5 – 0) + (0.3 – 0)/(0.5 – 0)) = 0.7, que é o mesmo valor dado pela primeira função do exemplo.

Para valores das variáveis independentes na faixa [0, 0.5] é possível escrever a regra para a variável dependente y_2:

IF x_1 = [0, 0.5] AND x_2 = [0.5, 0] THEN y_2 = [–0.5, 0.5].

Para $x_1 = 0.4$ e $x_2 = 0.3$, tem-se y2 = –0.5 + (0.5 – (–0.5))/2*((0.4 – 0)/(0.5 – 0) + (0.3 – 0.5)/(0 – 0.5)) = 0.1, que é o mesmo valor dado pela segunda função do exemplo.

CAPÍTULO 6

6 EXPERIMENTOS

O objetivo deste capítulo é apresentar estudos de casos para a proposta feita no capítulo anterior, exemplificando modelos baseados em regras de sistemas dinâmicos MIMO, incluindo o exemplo de um processo real. Os resultados obtidos e os processamentos correspondentes serão confrontados com dados resultantes de modelos fuzzy para efeitos de comparações.

6.1 Introdução

Neste capítulo são mostrados estudos de casos para a proposta apresentada no capítulo anterior relacionada com a geração de modelos aproximados para sistemas MIMO não lineares, incluindo dados de um sistema real. Os resultados obtidos e os processamentos relacionados serão confrontados com as informações resultantes de modelos *fuzzy*.

Na modelagem aproximada de sistemas MIMO apresentada nesse capítulo, para a geração de regras de decisão foi empregada à ferramenta Rosetta, cuja utilização esta exemplificada no Anexo A. Para tornar o processo de geração das regras mais automatizado possível e também deixá-los no formato apresentado na equação (5.2), foi desenvolvido o programa apresentado no Anexo B, e para a computação das regras resultantes e da expressão de interpolação (5.6) foi desenvolvido um programa em MatLab que está no Anexo C.

Para o modelo *fuzzy Takagi-Sugeno*, usado em comparação com a modelagem proposta, os parâmetros foram ajustados pelo toolbox ANFIS do MatLab e para a computação das regras resultantes foi desenvolvido um programa em MatLab, que está no Anexo D.

6.2 Experimentos[1]

Experimento 6.2.1

As equações (6.1) e (6.2) representam o modelo discreto de um dado sistema não linear com duas entradas e duas saídas (Wang et al., 2009).

1- Os resultados dos experimentos deste item fazem parte das publicações originadas desta tese e estão citadas nas referências bibliográficas (Pinheiro e Oliveira, $2013_{a,b}$, 2014).

$$z_1(k+1) = -0.8\sin[z_1(k)] - 0.16z_1(k-1) + 1.7u_1(k-1) + u_1(k) - 0.5u_2(k-1) + \frac{u_2(k-1)}{1+[u_2(k-1)]^2} \tag{6.1}$$

$$z_2(k+1) = -0.8sin[z_2(k)] - 0.16z_2(k-1) + 0.3u_1(k-1) + u_2(k) + 2u_2(k-1) + \frac{u_1(k-1)}{1+[u_1(k-1)]^2} \tag{6.2}$$

Os gráficos (em cor azul e vermelha) ilustrados na Figura 6.1 representam, respectivamente, os dados de entrada $u_1(k)$ e $u_2(k)$, juntamente com as informações de saída $z_1(k)$ e $z_2(k)$ do modelo discreto em questão.

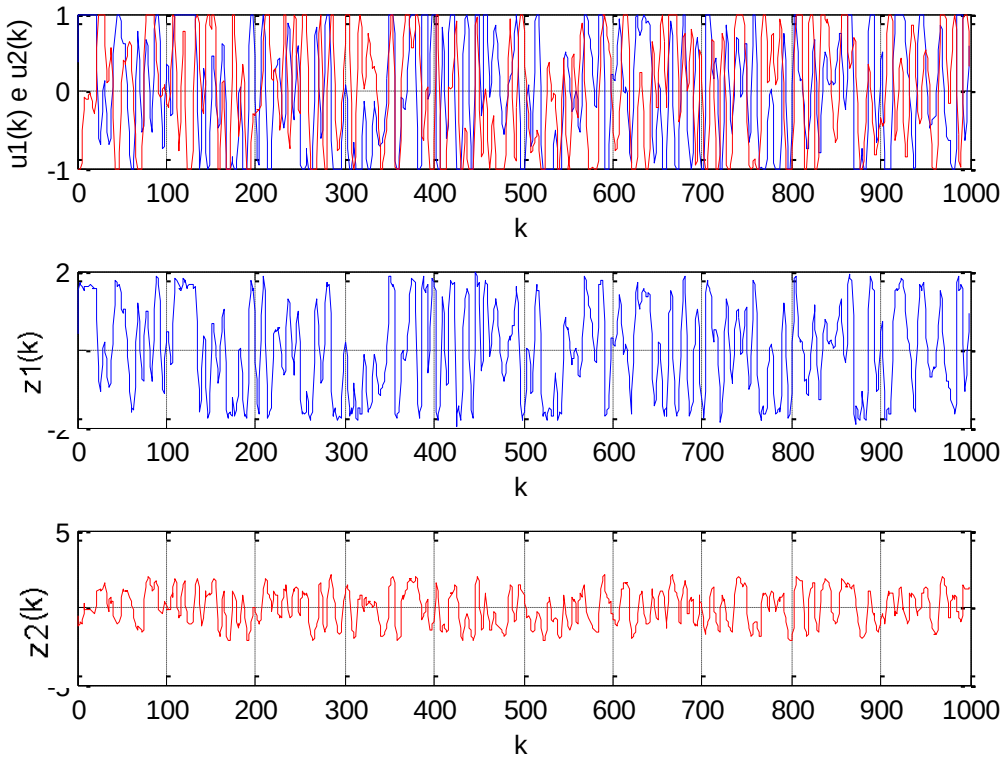

Figura 6.1 - Dados de entrada e saída do modelo MIMO 2x2 do Exemplo 6.2.1.

A representação do sistema em questão pode ser constituída com os dados da figura acima, onde são definidos dois sistemas de informações correspondentes. Para o primeiro *SI* foram definidos os seguintes atributos de condição $x_1 = y_1(k)$, $x_2 = y_1(k-1)$, $x_3 = u_1(k-1)$, $x_4 = u_1(k)$, $x_5 = u_2(k-1)$, e como atributo de decisão tem-se $y_1 = z_1(k+1)$. O reduto do *SI* correspondente é constituído pelas variáveis que definem os atributos de condição, ou seja, $\{x_1, x_2, x_3, x_4, x_5\}$. Para o segundo *SI* os atributos de decisão

são $x_1 = y_2(k),\ x_2 = y_2(k-1),\ x_3 = u_1(k-1),\ x_4 = u_2(k),\ x_5 = u_2(k-1),$ com o mesmo reduto citado e como atributo de decisão a informação $y_2 = z_2(k+1)$. Os dados dos dois *SI* correspondentes foram processados pela ferramenta Rosetta para a geração das regras de decisão. Os seguintes procedimentos foram realizados no aplicativo: *Import structure* → *Plain format*; *Discretization* → *Equal frequency binning* → *Intervals* = 2; *Reduction* → *Manual Reducer* → *Full*; *Rule generator (RSES)*. A seguir estão exemplificadas algumas regras de decisão geradas pelo aplicativo. Para o primeiro e segundo *SI*, têm-se algumas dessas regras de decisão representadas respectivamente em (6.3) e (6.4), onde o símbolo "$*$" representa os valores limites dos atributos de condição, que estão apresentados nas Tabela 6.1 e 6.2.

x_1([*, 0.0078)) AND x_2([*, 0.0069)) AND x_3([-0.0002, *)) AND x_4([-0.0002, *)) AND x_5([*, 0.0098)) => y_1(0.5783) OR y_1 (0.1715) OR y_1(0.3690) OR y_1 (1.0196) OR y_1 (1.1125) OR y_1 (1.5524)

x_1([0.0078, *)) AND x_2 ([*, 0.0069)) AND x_3 ([-0.0002, *)) AND x_4 ([-0.0002, *)) AND x_5 ([*, 0.0098)) => y_1 (0.3608) OR y_1 (1.0755) OR y_1 (0.7028) OR y_1 (0.4221) OR y_1 (1.4286) OR y_1 (1.2102) OR y_1 (1.3921) OR y_1 (1.2862) OR y_1 (1.3248) OR y_1 (0.7078) OR y_1 (0.7198) OR y_1 (0.6062) OR y_1 (1.8441)

⋮

x_1([0.0078, *)) AND x_2 ([0.0069, *)) AND x_3 ([*, -0.0002)) AND x_4 ([*, -0.0002)) AND x_5 ([0.0098, *)) => y_1 (-1.0981) OR y_1 (-0.2737) OR y_1 (-0.3411) OR y_1 (-0.3573) OR y_1 (-0.1726)

(6.3)

x_1([*, 0.0175)) AND x_2 ([*, 0.0133)) AND x_3 ([-0.0002, *)) AND x_4 ([*, 0.0153)) AND x_5 ([*, 0.0098)) => y_2 (-1.8951) OR y_2 (-1.4427) OR y_2 (-1.2759) OR y_2 (-1.3591) OR y_2 (-1.1575) OR y_2 (-1.2468) OR y_2 (-1.4742) OR y_2 (-1.4865) OR y_2 (-0.8659) OR y_2 (0.0243) OR y_2 (-1.1652) OR y_2 (-0.9048) OR y_2 (-0.1903) OR y_2 (0.3401)

x_1 ([0.0175, *)) AND x_2 ([*, 0.0133)) AND x_3 ([-0.0002, *)) AND x_4 ([*, 0.0153)) AND x_5 ([*, 0.0098)) => y_2 (0.2086) OR y_2 (0.3603)

⋮

x_1([*, 0.0175)) AND x_2 ([0.0133, *)) AND x_3 ([*, -0.0002)) AND x_4 ([*, 0.0153)) AND x_5 ([*, 0.0098)) => y_2 (-2.0444) OR y_2 (-0.6413) OR y_2 (-1.7972) OR y_2 (-0.9847) OR y_2 (-1.6644) OR y_2 (-1.9167) OR y_2 (-1.8453) OR y_2 (-1.6594) OR y_2 (-0.9517) OR y_2 (-0.5833)

(6.4)

Tabela 6.1 - Limites dos atributos de condição do primeiro *SI*.

	x_1	x_2	x_3	x_4	x_5
min	-1,9958	-1,9958	-1,0000	-1,0000	-1,0000
max	1,9739	1,9739	1,0000	1,0000	1,0000

Tabela 6.2 - Limites dos atributos de condição do segundo *SI*.

	x_1	x_2	x_3	x_4	x_5
min	-2,1490	-2,1490	-1,0000	-1,0000	-1,0000
max	2,1750	2,1750	1,0000	1,0000	1,0000

Utilizando uma extensão da representação da equação (5.3) do Capítulo 5, têm-se como proposto neste trabalho a forma representada em (6.5). Onde o índice subscrito "$_1$" diz respeito aos parâmetros das regras que modela a informação y_1. Idem para o índice "$_2$" relacionado a informação y_2.

$$r_i: IF\ x_1 = \alpha_1^{(g)}\ AND\ \ x_2 = \beta_1^{(g)}\ AND\ \cdots AND\ \ x_N = \varepsilon_1^{(g)}\ \ THEN\ y_1 = \sigma_1^{(g)}$$

$$r_i: IF\ x_1 = \alpha_2^{(g)}\ AND\ \ x_2 = \beta_2^{(g)}\ AND\ \cdots AND\ \ x_N = \varepsilon_2^{(g)}\ THEN\ y_2 = \sigma_2^{(g)} \quad (6.5)$$

Em (6.6) e (6.7) têm-se os parâmetros dos antecedentes das regras dos modelos correspondentes. Já nas Tabelas 6.3 e 6.4 estão as combinações das partições dos antecedentes, mais os parâmetros dos consequentes do modelo baseado em regras, cujos dados estão relacionados às informações de y_1 e y_2.

$\alpha_1^{(1)} = [-1.9958, 0.0078]$	$\alpha_1^{(2)} = [0.0078, 1.9739]$
$\beta_1^{(1)} = [-1.9958, 0.0069]$	$\beta_1^{(2)} = [0.0069, 1.9739]$
$\gamma_1^{(1)} = [-1.0000, -0.0002]$	$\gamma_1^{(2)} = [-0.0002, 1.0000]$
$\delta_1^{(1)} = [-1.0000, -0.0002]$	$\delta_1^{(2)} = [-0.0002, 1.0000]$
$\varepsilon_1^{(1)} = [-1.0000, 0.0098]$	$\varepsilon_1^{(2)} = [0.0098, 1.0000]$
	(6.6)

$\alpha_2^{(1)} = [-2.1490, 0.0175]$	$\alpha_2^{(2)} = [0.0175, 2.1750]$
$\beta_2^{(1)} = [-2.1490, 0.0133]$	$\beta_2^{(2)} = [0.0133, 2.1750]$
$\gamma_2^{(1)} = [-1.0000, -0.0002]$	$\gamma_2^{(2)} = [-0.0002\ \ 1.0000]$
$\delta_2^{(1)} = [-1.0000, 0.0153]$	$\delta_2^{(2)} = [0.0153, 1.0000]$
$\varepsilon_2^{(1)} = [-1.0000, 0.0098]$	$\varepsilon_2^{(2)} = [0.0098, 1.0000]$
	(6.7)

Tabela 6.3 - Representação tabular das regras relativas a y_1.

	$\delta_1^{(1)}\varepsilon_1^{(1)}$	$\delta_1^{(1)}\varepsilon_1^{(2)}$	$\delta_1^{(2)}\varepsilon_1^{(1)}$	$\delta_1^{(2)}\varepsilon_1^{(2)}$
$\alpha_1^{(1)}\beta_1^{(1)}\gamma_1^{(1)}$	$[-1.8796, -0.0524]$	$[-1.8250, 0.0383]$	$[-0.0850, 1.0377]$	$[-0.5360, 0.9436]$
$\alpha_1^{(1)}\beta_1^{(1)}\gamma_1^{(2)}$	$[-0.1581]$	–	$[0.1715, 1.5524]$	$[0.2026, 0.3436]$
$\alpha_1^{(1)}\beta_1^{(2)}\gamma_1^{(1)}$	$[-1.7648, -0.0483]$	$[-1.9958, 0.0075]$	–	–
$\alpha_1^{(1)}\beta_1^{(2)}\gamma_1^{(2)}$	$[-0.5788]$	$[-1.6984, 0.1732]$	–	$[0.1106]$
$\alpha_1^{(2)}\beta_1^{(1)}\gamma_1^{(1)}$	–	$[-0.0372]$	$[0.0287]$	$[0.1198, 0.3750]$
$\alpha_1^{(2)}\beta_1^{(1)}\gamma_1^{(2)}$	–	$[-0.0065]$	$[0.3608, 1.8441]$	$[0.1533,1.9739]$
$\alpha_1^{(2)}\beta_1^{(2)}\gamma_1^{(1)}$	$[-1.6360, -0.2897]$	$[-1.0981, -0.1726]$	–	–
$\alpha_1^{(2)}\beta_1^{(2)}\gamma_1^{(2)}$	$[-1.1266, 0.0307]$	–	$[-0.0149, 1.8602]$	$[0.0506, 1.9067]$

Tabela 6.4 - Representação tabular das regras relativas a y_2.

	$\delta_2^{(1)}\varepsilon_2^{(1)}$	$\delta_2^{(1)}\varepsilon_2^{(2)}$	$\delta_2^{(2)}\varepsilon_2^{(1)}$	$\delta_2^{(2)}\varepsilon_2^{(2)}$
$\alpha_2^{(1)}\beta_2^{(1)}\gamma_2^{(1)}$	$[-2.1490, -0{,}1817]$	$[-0.0455]$	$[-0.3665{,}0.9336]$	$[-0.2588{,}1.1903]$
$\alpha_2^{(1)}\beta_2^{(1)}\gamma_2^{(2)}$	$[-1.8951, 0.3401]$	–	$[0.0690{,}1.0648]$	$[1.4544]$
$\alpha_2^{(1)}\beta_2^{(2)}\gamma_2^{(1)}$	$[-2.0444, -0.5833]$	$[-0.9929, -0.1899]$	$[-0.3829, -0.3500]$	$[-0.3019, -0.0529]$
$\alpha_2^{(1)}\beta_2^{(2)}\gamma_2^{(2)}$	$[-1.4168, -0.0173]$	–	–	–
$\alpha_2^{(2)}\beta_2^{(1)}\gamma_2^{(1)}$	–	–	–	$[0.0507{,}1.6800]$
$\alpha_2^{(2)}\beta_2^{(1)}\gamma_2^{(2)}$	$[0.2086{,}0.3603]$	–	$[0.4193{,}1.3315]$	$[0.5387{,}2.0107]$
$\alpha_2^{(2)}\beta_2^{(2)}\gamma_2^{(1)}$	–	$[-1.2780, -0.3466]$	–	$[-0.3596{,}1.6718]$
$\alpha_2^{(2)}\beta_2^{(2)}\gamma_2^{(2)}$	$[-1.0473, 0.3929]$	$[-0.9358, 0.5999]$	$[0.5651]$	$[0.3658, 2.1750]$

Para estimar valores intermediários deste modelo usa-se a fórmula de interpolação (5.6) com *N* = 5. Exemplo: para $x_1 = 0.0077$, $x_2 = 0.0068$, $x_3 = -0.9999$, $x_4 = -0.9999$ e $x_5 = 0.0093$, vem $y_1 = -0.7834$ e $y_2 = -0.9789$.

A Figura 6.2 ilustra os resultados do modelo baseado em regras para as informações das saídas y_1e y_2 (gráficos em cor vermelha) em relação às respostas do modelo não linear original (em cor azul), utilizando os dados originais da Figura 6.1.

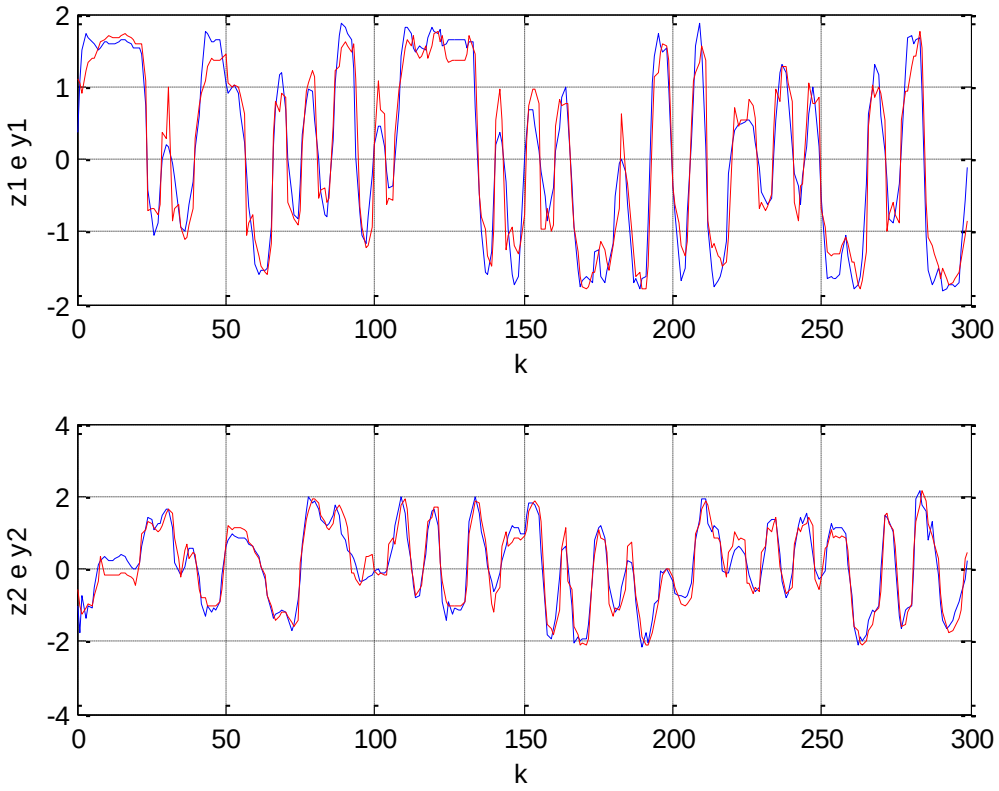

Figura 6.2 - Gráficos de resultado do modelo aproximado em relação aos dados originais.

Os gráficos na Figura 6.4 são resultantes de outra computação do modelo aproximado, utilizando agora outro conjunto de dados do sistema original representado pela Figura 6.3, cuja finalidade é validar a modelagem obtida. Nos gráficos das informações das saídas em cor azul têm-se as informações do sistema original e em cor vermelha os dados simulados do modelo aproximado. A comparação dos resultados indica uma modelagem adequada do sistema com a utilização da representação proposta neste trabalho.

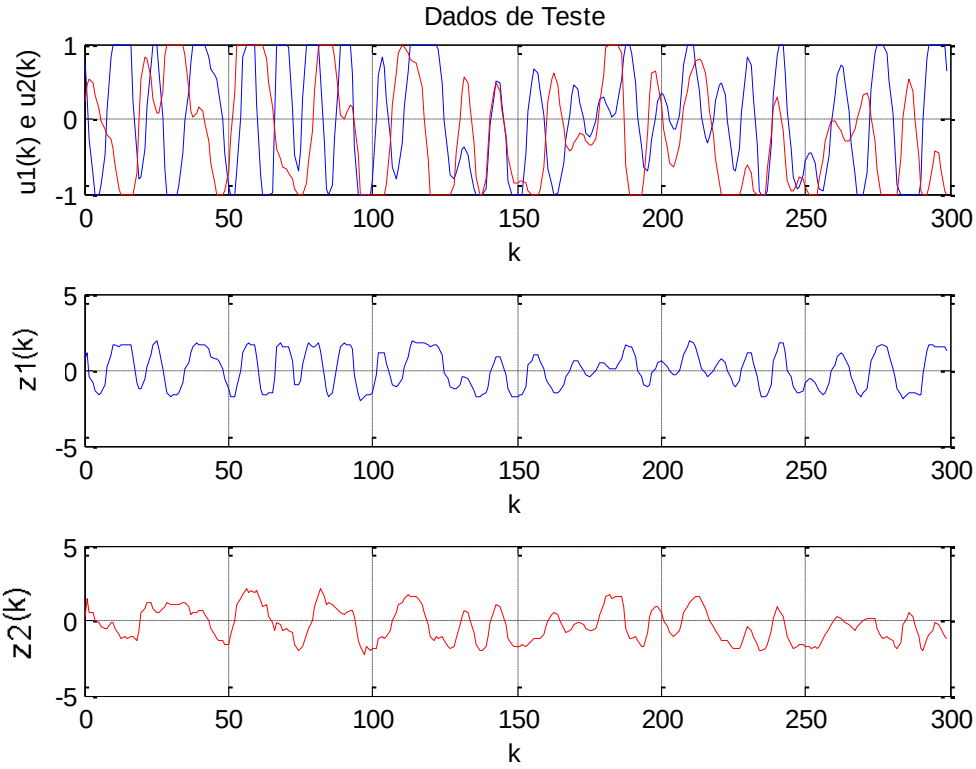

Figura 6.3 - Gráficos de dados para validação do modelo aproximado.

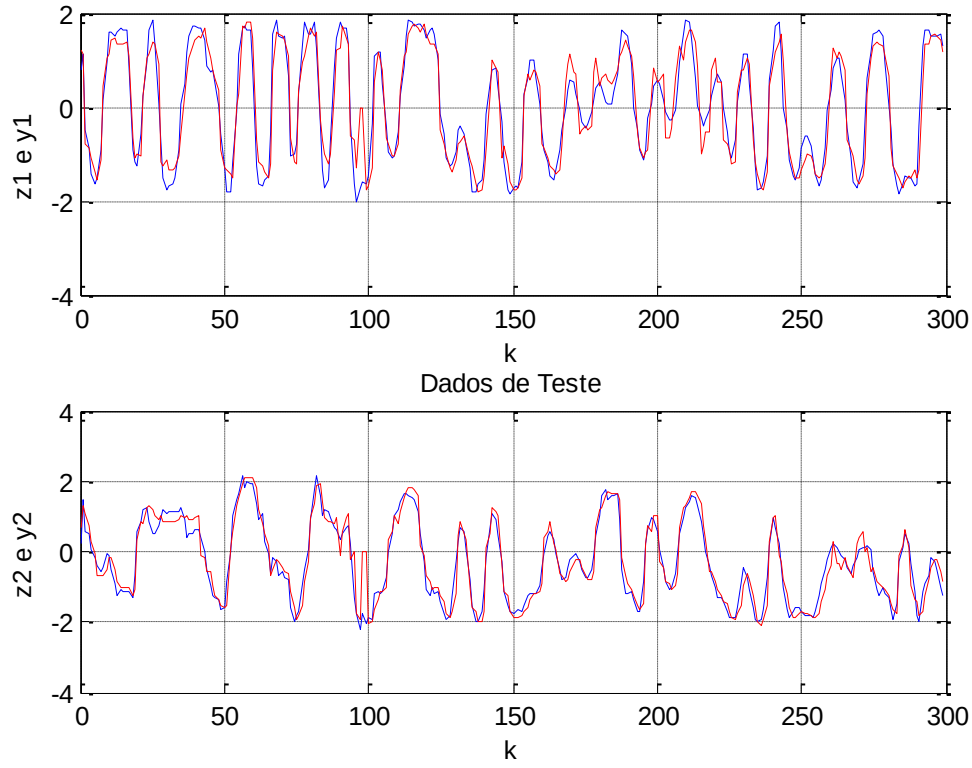

Figura 6.4 - Resultado de validação do modelo aproximado.

Também foi realizada uma comparação com um modelo *fuzzy Takagi-Sugeno* (com parâmetros ajustados pelo toolbox *ANFIS* do MatLab), onde tem-se em (6.8) e (6.9) a representação do mesmo, com duas funções de pertinência trapezoidais para cada informação de entrada, cuja representação está ilustrada na Figura 6.5. Onde $a_n, m_m, b_n \, e \, n_n$ são os parâmetros da função de pertinência.

$$r_{1i}: IF\ x_1 = A_1^{(g)} AND\ x_2 = B_1^{(g)}\ AND \ldots AND\ x_n = Z_1^{(g)} THEN\ y_{1g} = c_{10g} + c_{11g}.x_1 + \cdots + c_{1ng}.x_n \quad (6.8)$$

$$r_{2i}: IF\ x_1 = A_2^{(g)} AND\ x_2 = B_2^{(g)}\ AND \ldots AND\ x_n = Z_2^{(g)} THEN\ y_{2g} = c_{20g} + c_{21g}.x_1 + \cdots + c_{2ng}.x_n \quad (6.9)$$

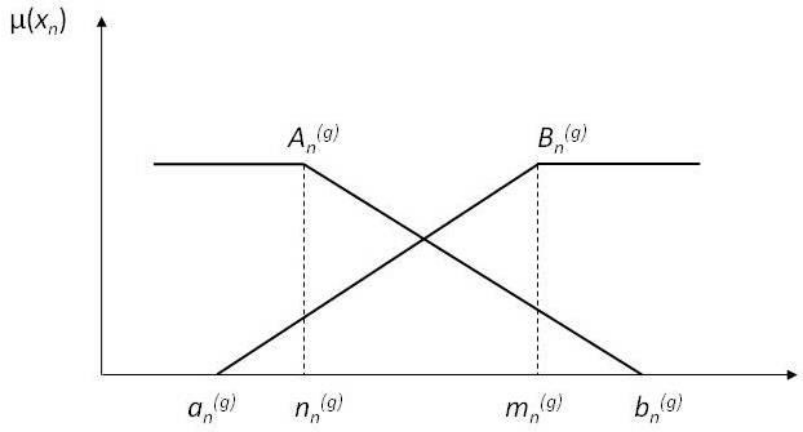

Figura 6.5 - Funções de pertinência dos conjuntos nebulosos da modelagem *fuzzy* empregada.

A Figura 6.6 ilustra as respostas das saídas do modelo fuzzy com os mesmos dados originais da Figura 6.3.

A vantagem do modelo *fuzzy* é a sua melhor capacidade de interpolação. A vantagem do modelo aproximado é o seu menor tempo de computação, pois não possui procedimentos de fuzificação que ocorre em modelos fuzzy, processando mais rapidamente as regras de modelagem. Para efeito de comparações foi utilizado um notebook com as características: processador Intel Celeron M CPU 520; 1.6GHz; 2GB de RAM. O tempo de processamento do modelo aproximado foi de 0,51segundos e do modelo *fuzzy* foi de 33 segundos. Além disso, o tempo para obter as regras também diferiu. Neste exemplo, gastou-se setenta e oito

segundos na obtenção das regras do modelo *fuzzy*, e pouco mais de um segundo para o modelo aproximado.

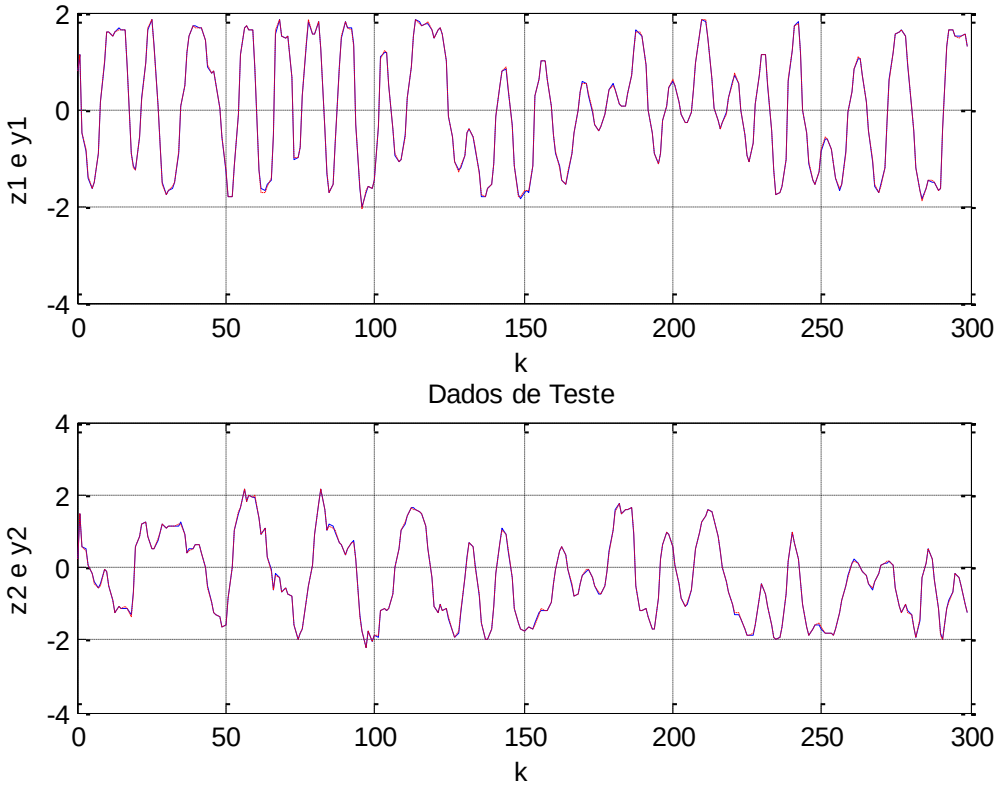

Figura 6.6 - Gráfico com resultados de validação do modelo *fuzzy*.

Experimento 6.2.2

Este exemplo aborda a modelagem prática de um sistema de nível real constituído por dois reservatórios acoplados conforme o desenho representativo ilustrado na Figura 6.7. As equações (6.10) e (6.11) representam o modelo não linear contínuo deste tipo de sistema. As variáveis $q_{i_1}(t)$ e $q_{i_2}(t)$ representam os fluxos de entrada de um determinado fluido nos reservatórios. A variável $h_1(t)$ simboliza o nível do fluido no reservatório de área transversal A_1, e a variável $h_2(t)$ representa o nível do fluido armazenado no reservatório de área transversal A_2. No sistema em questão, as áreas transversais dos dois reservatórios são iguais, portanto, $A_1 = A_2 = A$. O parâmetro "a" diz respeito à seção transversal das tubulações mais registros (R_1 e R_2) por onde escoa o fluxo q_1 entre os reservatórios e o fluxo de saída q_2. O fator "g" é valor da aceleração da gravidade. O processo pode ser classificado como um sistema MIMO (2 x 2) não linear.

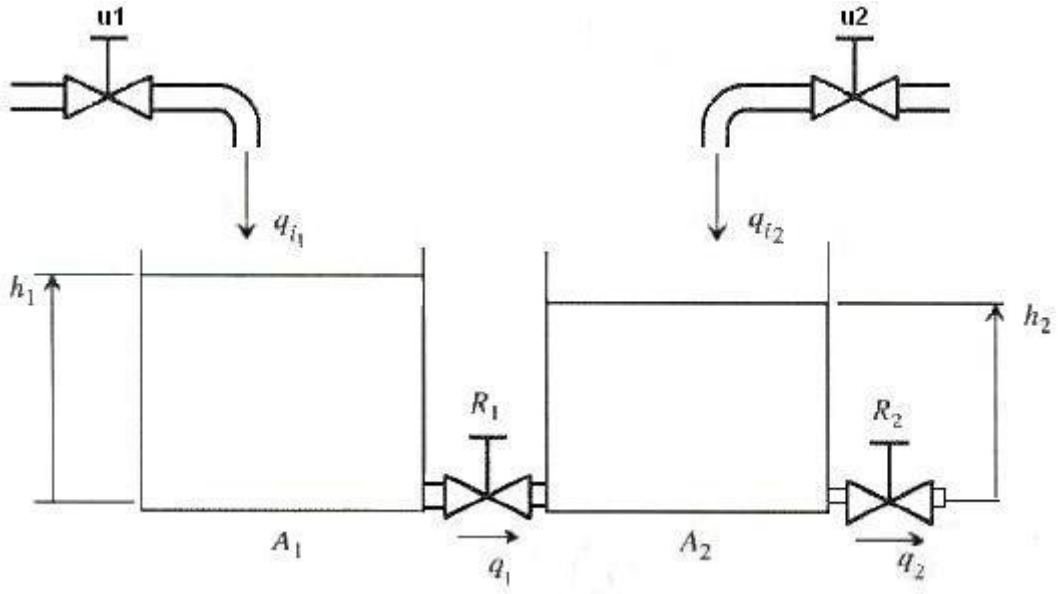

Figura 6.7 - Representação de um sistema de nível acoplado.

$$A_1 \frac{h_1(t)}{dt} = q_{i_1}(t) - a.\sqrt{2.g}.\sqrt{[h_1(t) - h_2(t)]} \tag{6.10}$$

$$A_2 \frac{h_2(t)}{dt} = q_{i_2}(t) + a.\sqrt{2g}.\sqrt{[h_1(t) - h_2(t)]} - a.\sqrt{2.g}.\sqrt{h_2(t)} \tag{6.11}$$

A Figura 6.8 contém dados reais de um sistema de nível, coletados da planta (cuja foto encontra-se na Figura 6.9), do laboratório de Controle de Processos da Universidade Federal de Itajubá, onde as variáveis u_1 e u_2 são informações de comando (valores de tensões de 0 a 5 V) que controlam as vazões de entrada (qi_1 e qi_2) por meio de servo-válvulas proporcionais comandadas eletricamente. As informações dos níveis (h_1 e h_2) são fornecidas por transdutores que fornecem correntes elétricas proporcionais (na faixa de 0 a 20 mA).

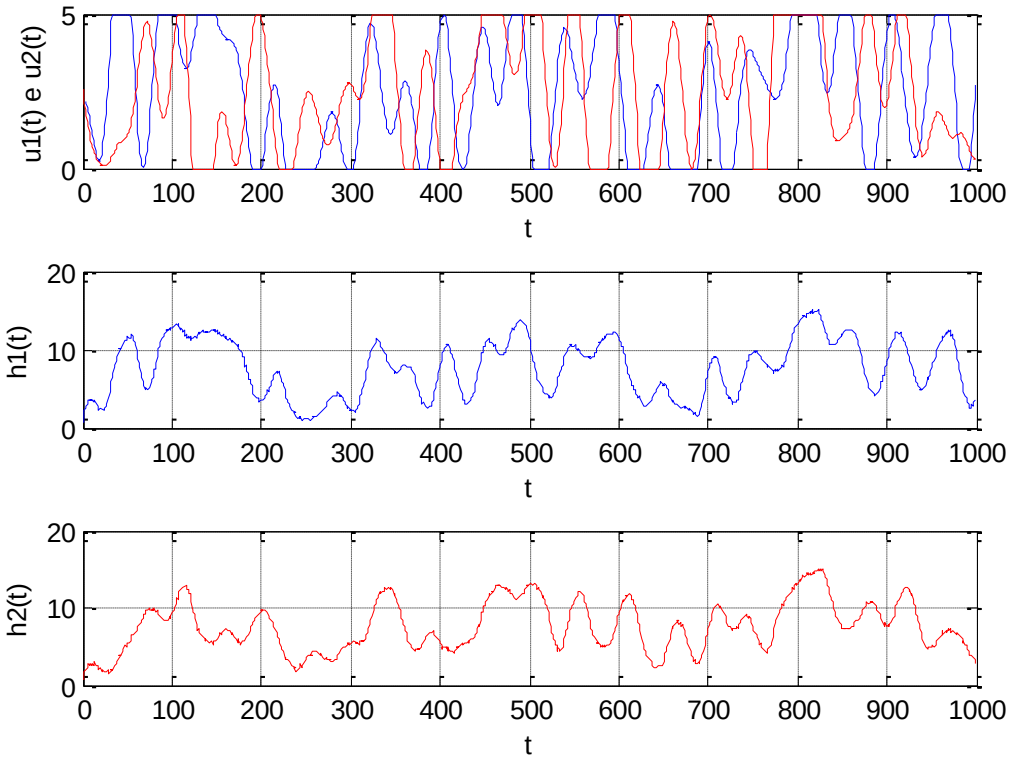

Figura 6.8 - Gráfico com dados reais de um sistema de nível.

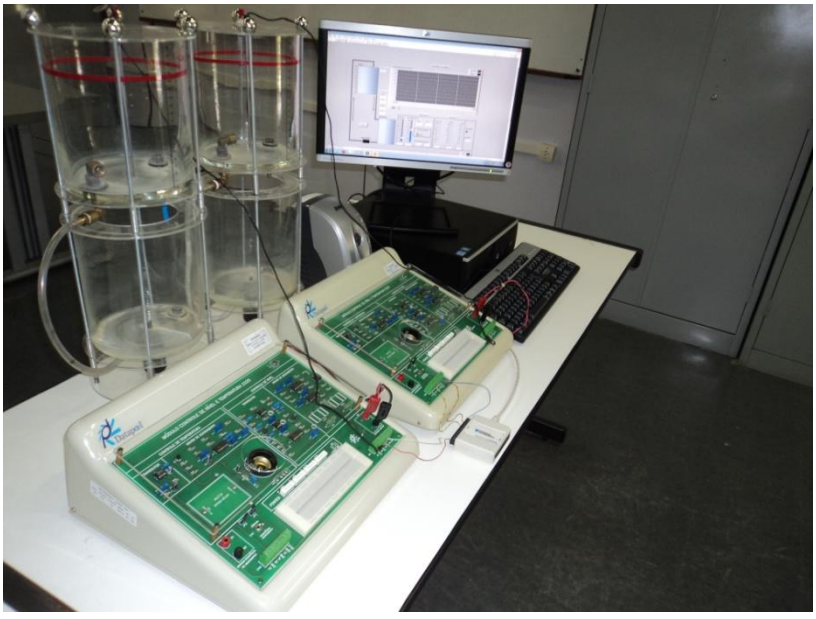

Figura 6.9 – Planta real do sistema de tanques acoplados utilizado (UNIFEI-Itajubá).

Neste experimento será obtido um modelo aproximado do processo em questão. O procedimento utilizado será similar ao do experimento anterior, onde com os dados das

medições são definidos dois *SI*. Será adotado um modelo discreto para modelar o sistema a partir das amostras das variáveis mensuradas do processo. As variáveis escolhidas como atributos de condição foram: $x_1 = h_1(k-1)$, $x_2 = h_2(k-1)$, $x_3 = u_1(k-1)$ e $x_4 = u_2(k-1)$. Logo como reduto tem-se $\{x_1, x_2, x_3, x_4\}$. Como variáveis de decisão têm-se $y_1 = h_1(k)$ e $y_2 = h_2(k)$. Utilizando-se a ferramenta Rosetta para a geração das regras de decisão, com os mesmos procedimentos citados anteriormente, têm-se em (6.12) os parâmetros do modelo baseado em regras correspondente, lembrando que o índice subscrito "$_1$" diz respeito aos parâmetros das regras que modela a informação h_1. Idem para o índice "$_2$" para a informação h_2. As Tabelas 6.5 e 6.6 trazem as combinações dos parâmetros que definem as regras resultantes conforme a metodologia adotada.

$\alpha_1^{(1)} = \alpha_2^{(1)} = [0.1161, 7.8422]$	$\alpha_1^{(2)} = \alpha_2^{(2)} = [7.8422, 15.1129]$
$\beta_1^{(1)} = \beta_2^{(1)} = [0.1161, 7.2825]$	$\beta_1^{(2)} = \beta_2^{(2)} = [7.2825, 15.1500]$
$\gamma_1^{(1)} = \gamma_2^{(1)} = [0.0000, 2.5182]$	$\gamma_1^{(2)} = \gamma_2^{(2)} = [2.5182, 5.0000]$
$\delta_1^{(1)} = \delta_2^{(1)} = [0.0000, 2.2909]$	$\delta_1^{(2)} = \delta_2^{(2)} = [2.2909, 5.0000]$
	(6.12)

Tabela 6.5 - Representação tabular das regras relativas a y_1.

	$\delta_1^{(1)}$	$\delta_1^{(2)}$
$\alpha_1^{(1)}\beta_1^{(1)}\gamma_1^{(1)}$	$[0.9132, 7.4551]$	$[0.7585, 7.4551]$
$\alpha_1^{(1)}\beta_1^{(1)}\gamma_1^{(2)}$	$[3.6721, 8.2395]$	$[3.8327, 8.0611]$
$\alpha_1^{(1)}\beta_1^{(2)}\gamma_1^{(1)}$	$[2.7679, 7.5106]$	$[2.6030, 7.5150]$
$\alpha_1^{(1)}\beta_1^{(2)}\gamma_1^{(2)}$	$[5.8319, 7.9873]$	$[5.6790, 8.3391]$
$\alpha_1^{(2)}\beta_1^{(1)}\gamma_1^{(1)}$	$[7.5026, 9.8663]$	$[7.9175, 9.9281]$
$\alpha_1^{(2)}\beta_1^{(1)}\gamma_1^{(2)}$	$[7.8454, 12.4787]$	$[8.6196, 12.3480]]$
$\alpha_1^{(2)}\beta_1^{(2)}\gamma_1^{(1)}$	$[8.9894, 11.1270]$	$[7.4801, 12.1005]$
$\alpha_1^{(2)}\beta_1^{(2)}\gamma_1^{(2)}$	$[7.9318, 12.6000]$	$[8.1094, 15.0398]$

Tabela 6.6 - Representação tabular das regras relativas a y_2.

	$\delta_2^{(1)}$	$\delta_2^{(2)}$
$\alpha_2^{(1)}\beta_2^{(1)}\gamma_2^{(1)}$	$[1.6769, 6.8028]$	$[0.6134, 7.4802]$
$\alpha_2^{(1)}\beta_2^{(1)}\gamma_2^{(2)}$	$[1.5575, 6.8032]$	$[4.4102, 7.7916]$
$\alpha_2^{(1)}\beta_2^{(2)}\gamma_2^{(1)}$	$[6.7531, 10.3696]$	$[7.1886, 12.8432]$
$\alpha_2^{(1)}\beta_2^{(2)}\gamma_2^{(2)}$	$[7.0228, 8.7780]$	$[7.7179, 10.3662]$
$\alpha_2^{(2)}\beta_2^{(1)}\gamma_2^{(1)}$	$[4.2640, 6.6692]$	$[6.3018, 7.0897]$
$\alpha_2^{(2)}\beta_2^{(1)}\gamma_2^{(2)}$	$[2.7907, 7.3063]$	$[5.5237, 7.4495]$
$\alpha_2^{(2)}\beta_2^{(2)}\gamma_2^{(1)}$	$[10.4508, 12.5830]$	$[7.5423, 13.7882]$
$\alpha_2^{(2)}\beta_2^{(2)}\gamma_2^{(2)}$	$[6.9939, 11.7686]$	$[7.6553, 15.1458]$

Para estimar valores intermediários deste modelo pode-se usar a formula de interpolação (5.6) para *N = 4*. O primeiro gráfico da Figura 6.10 mostra os resultados computados do modelo aproximado para a saída h_1 (gráfico em cor vermelha) em relação à resposta do processo real (em cor azul). O segundo gráfico ilustra o resultado do modelo aproximado para a saída h_2 (em cor vermelha) em relação à resposta da planta prática (em cor azul). As entradas de comando do sistema são as mesmas da Figura 6.8.

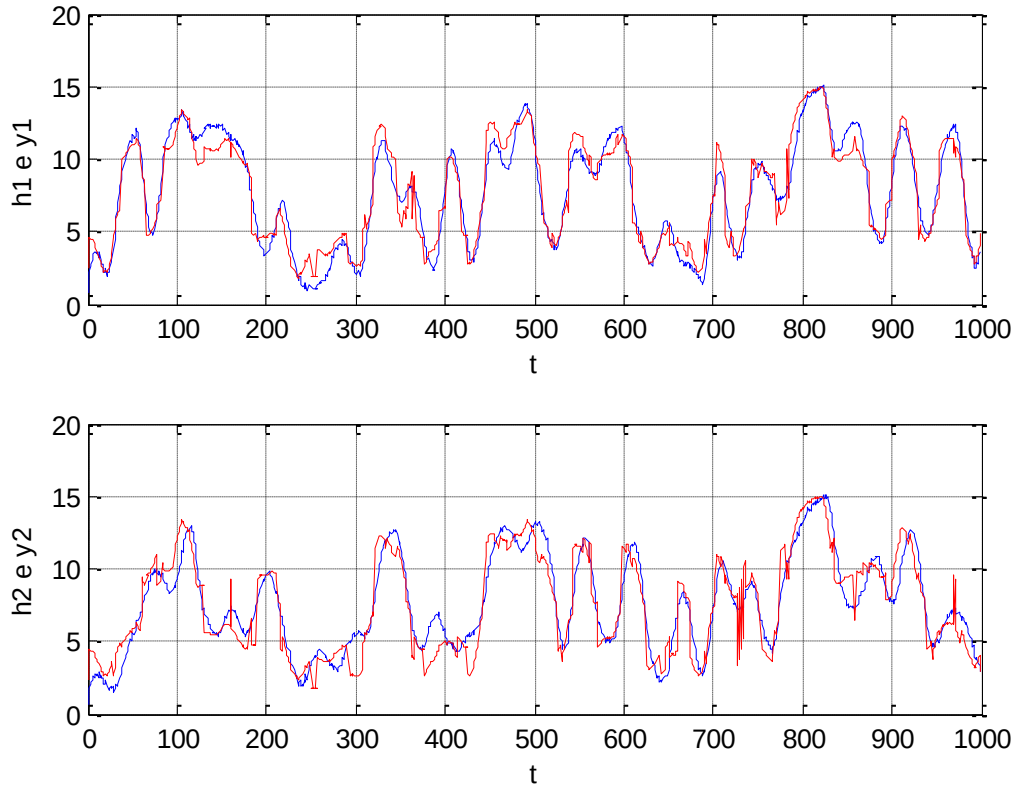

Figura 6.10 - Gráfico dos resultados do modelo aproximado com discretização em 2 níveis.

Os gráficos da Figura 6.11 constituem outro conjunto de dados de medidas reais do sistema que visam à validação do modelo aproximado.

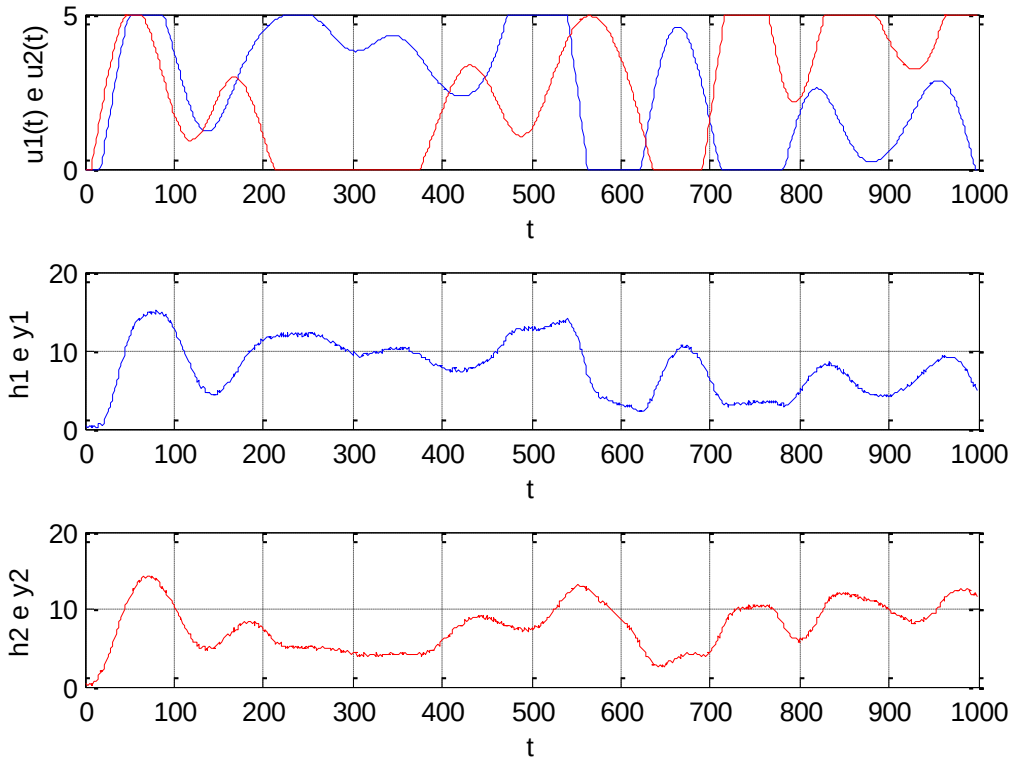

Figura 6.11 - Outro conjunto de dados para validação do modelo.

Na Figura 6.12 seguem os resultados com os mesmos dados da Figura 6.11, computados do modelo aproximado para a saída h_1 (em cor vermelha) em relação com a resposta do processo real (em cor azul) para validação do modelo. Idem para a saída h_2. Alguns valores destoantes do modelo aproximado podem ser atribuídos a algumas regras que não mapearam adequadamente as relações não lineares deste tipo de processo. A utilização de outras medições que contenham mais amostras e/ou excitações mais persistentes nas entradas de comando da planta, podem fornecer informações mais pertinentes que possibilitam aproximações melhores nos modelos resultantes. Um maior número de regras de modelagem (aumentando o intervalo de discretização das variáveis do modelo), também pode levar a uma maior precisão dos modelos resultantes.

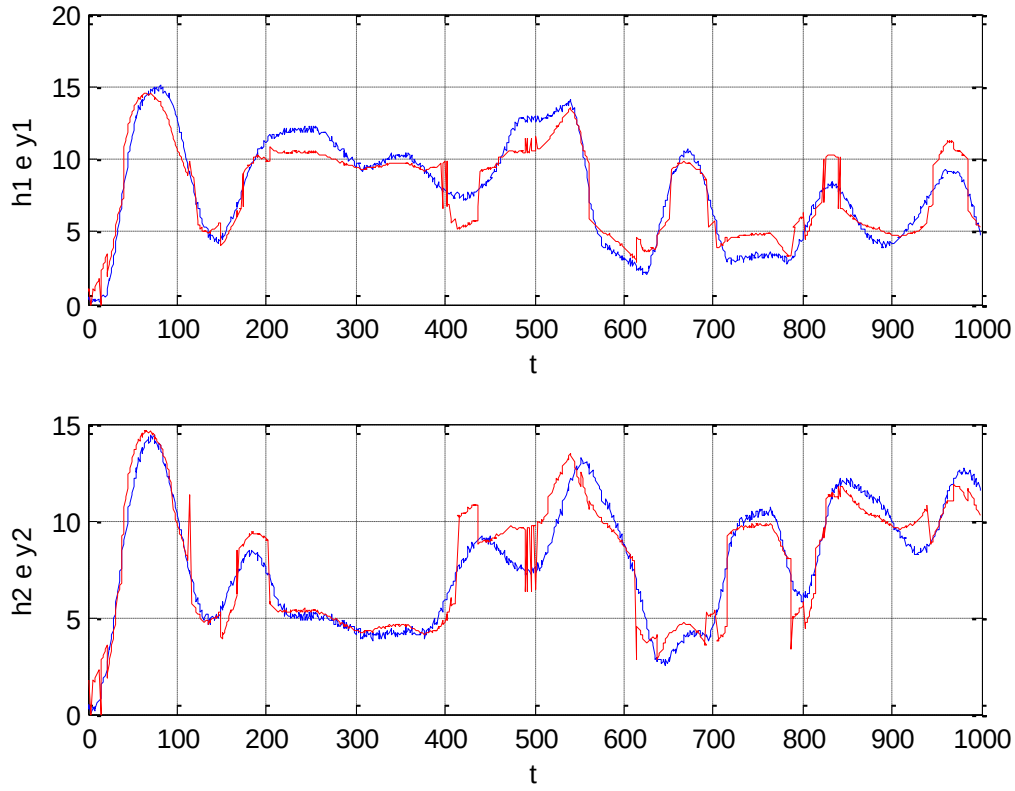

Figura 6.12 - Gráfico com resultados para validação do modelo com discretização em 2 níveis.

Com o objetivo de analisar o resultado do experimento aumentando o intervalo de discretização das variáveis do *SI* correspondente e consequentemente o número de regras, mas mantendo o mesmo método de discretização, é apresentado na Figura 6.13 o resultado das computações do modelo com os mesmos dados reais da Figura 6.8, e na Figura 6.14 o resultado para o conjunto de dados de validação dado pela Figura 6.11.

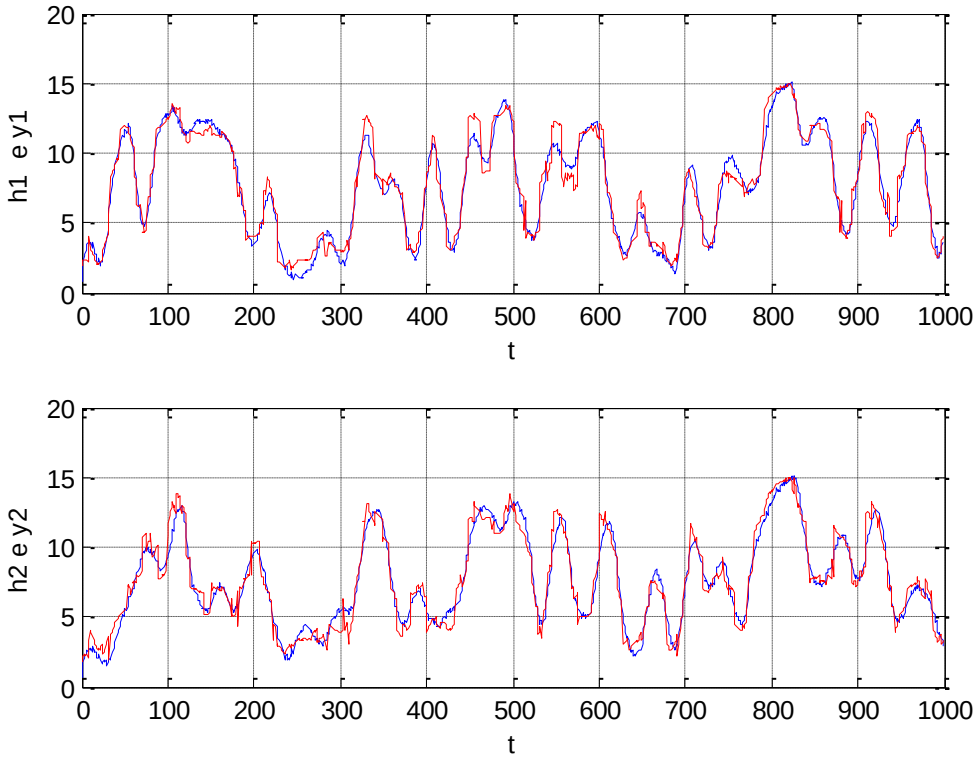

Figura 6.13 - Gráfico dos resultados do modelo aproximado com discretização em três níveis.

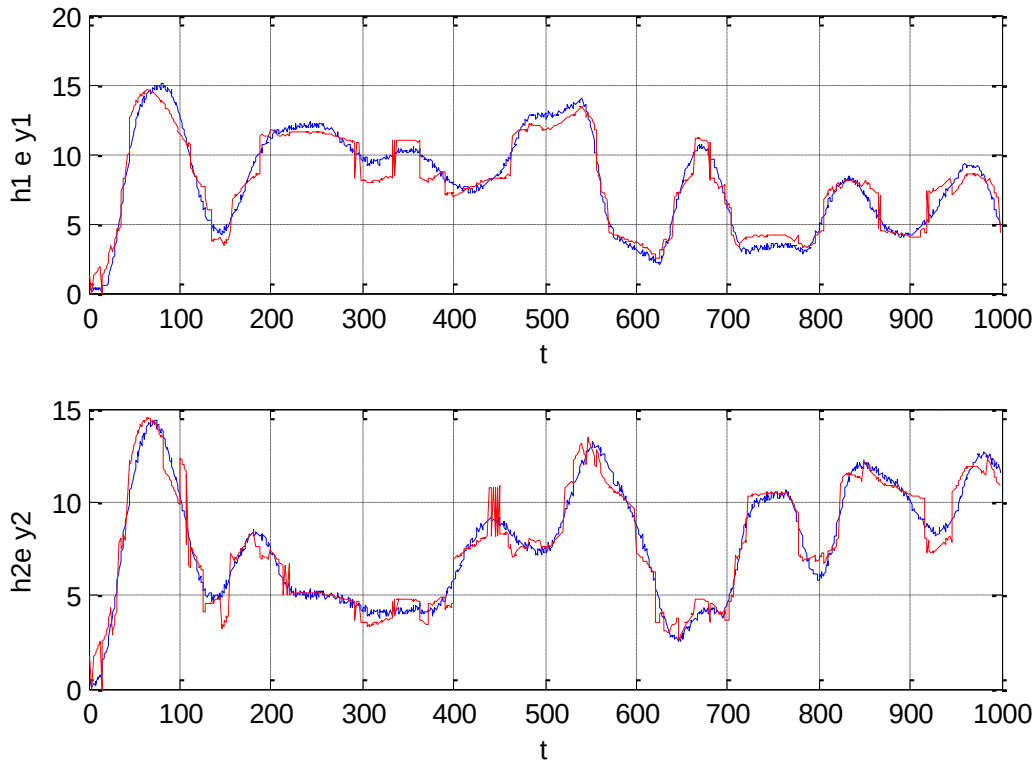

Figura 6.14 – Gráfico com resultados do outro conjunto de dados para validação do modelo com discretização em três níveis.

Com o objetivo de analisar o resultado do experimento utilizando outro método de discretização diferente do procedimento *Equal Frequency Binning*, será empregado o método de discretização *Boolean Reasoning*. Na Figura 6.15 é apresentado o resultado com os dados reais da Figura 6.8, e na Figura 6.16 o resultado para outro conjunto de dados de validação (ilustrados na Figura 6.11).

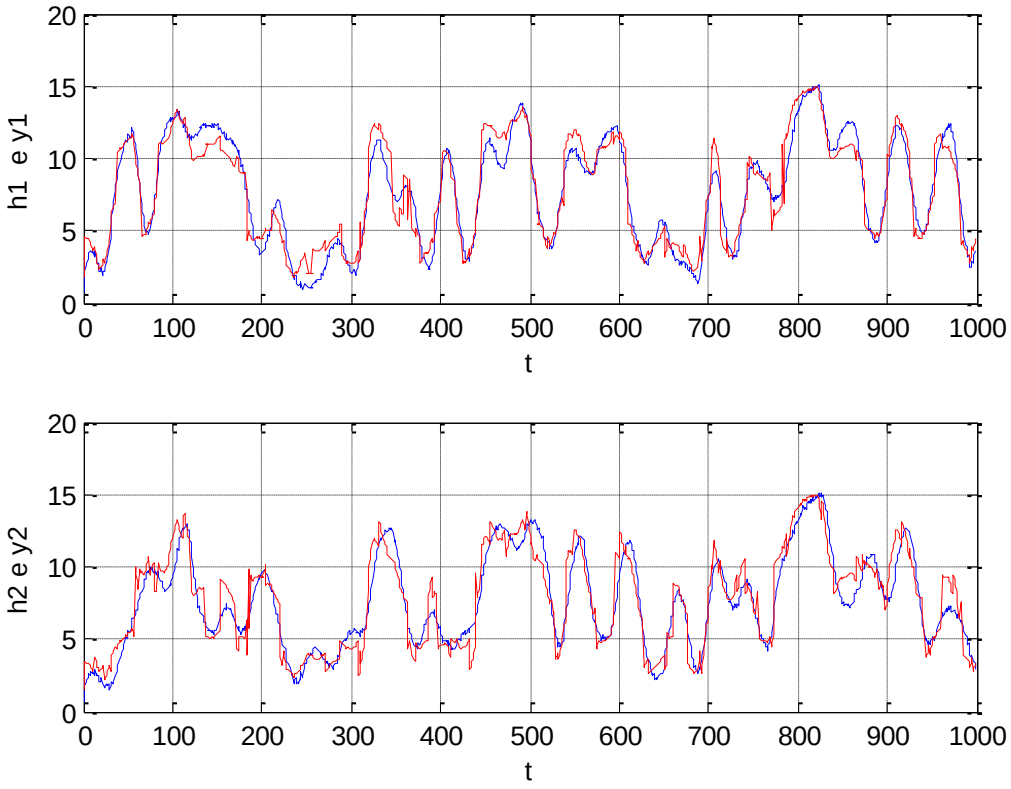

Figura 6.15 - Gráfico dos resultados do modelo aproximado com o método de discretização Boolean Reasoning.

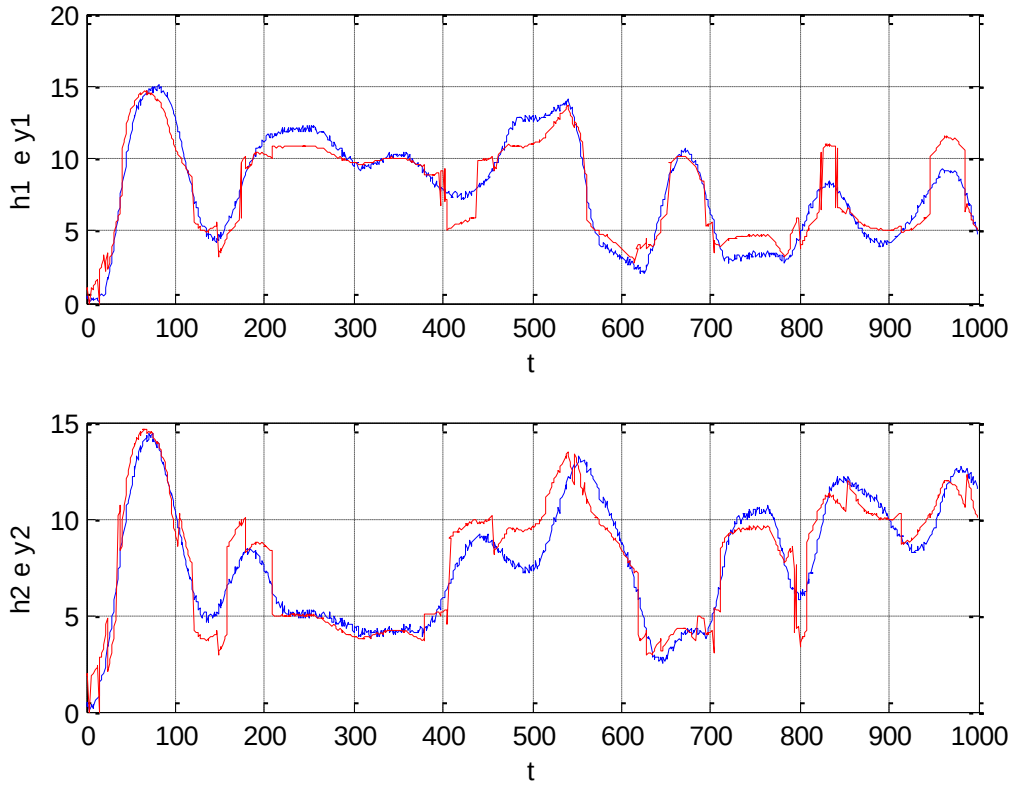

Figura 6.16 – Gráfico do resultado de outro conjunto de dados para validação do modelo com o método de discretização Boolean.

A Tabela 6.7, apresenta um resumo dos resultados obtidos com os dois métodos de discretização utilizados, e com nível de discretização empregada.

Nesse exemplo, também foi realizada uma comparação com um modelo *fuzzy Takagi-Sugeno*, representado pelas equações (6.8) e (6.9), com os parâmetros ajustados pelo *toolbox ANFIS* do Matlab, com duas funções de pertinência trapezoidal (representadas na Figura 6.5) para cada informação de entrada. A Figura 6.17, ilustra o resultado das saídas do modelo *fuzzy* com os mesmos dados originais da Figura 6.11.

Tabela 6.7 – Comparações entre os métodos e níveis de discretização empregados.

Modelo Baseado em Regras					
Método de discretização.	Número de intervalos de discretização.	Nº de regras.	Média da somatória do erro quadrático com os mesmos dados dos procedimentos de geração das regras. (Treinamento)	Tempo de simulação associado (segundos).	Média da somatória do erro quadrático com os dados de validação dos modelos. (Validação)
Equal Frequency Binning	2	16	Figura 6.10 $y_1 =>$ 2,32% $y_2 =>$ 2,96%	0,26	Figura 6.12 $y_1 =>$ 2,65% $y_2 =>$ 2,45%
Equal Frequency Binning	3	59	Figura 6.13 $y_1 =>$ 1,76% $y_2 =>$ 1,78%	0,70	Figura 6.14 $y_1 =>$ 1,84% $y_2 =>$ 1,69%
Boolean Reasoning	-	22	Figura 6.15 $y_1 =>$ 2,39% $y_2 =>$ 2,73%	0,36	Figura 6.16 $y_1 =>$ 2,66% $y_2 =>$ 2,51%

A Tabela 6.8, apresenta um resumo dos resultados do modelo *Fuzzy Takagi-Sugeno*.

Tabela 6.8 – Resultados do Modelo Fuzzy Takagi-Sugeno.

Modelo *Fuzzy Takagi-Sugeno*					
Modelo Fuzzy Takagi-Sugeno		Nº de Épocas	Média da somatória do erro quadrático com os mesmos dados dos procedimentos de geração das regras. (Treinamento)	Tempo de simulação associado (segundos).	Média da somatória do erro quadrático com os dados de validação dos modelos. (Validação)
Fuzzyy	-	50	$y_1 =>$ 0,028% $y_2 =>$ 0,020%	10,36	$y_1 =>$ 0,023% $y_2 =>$ 0,016%

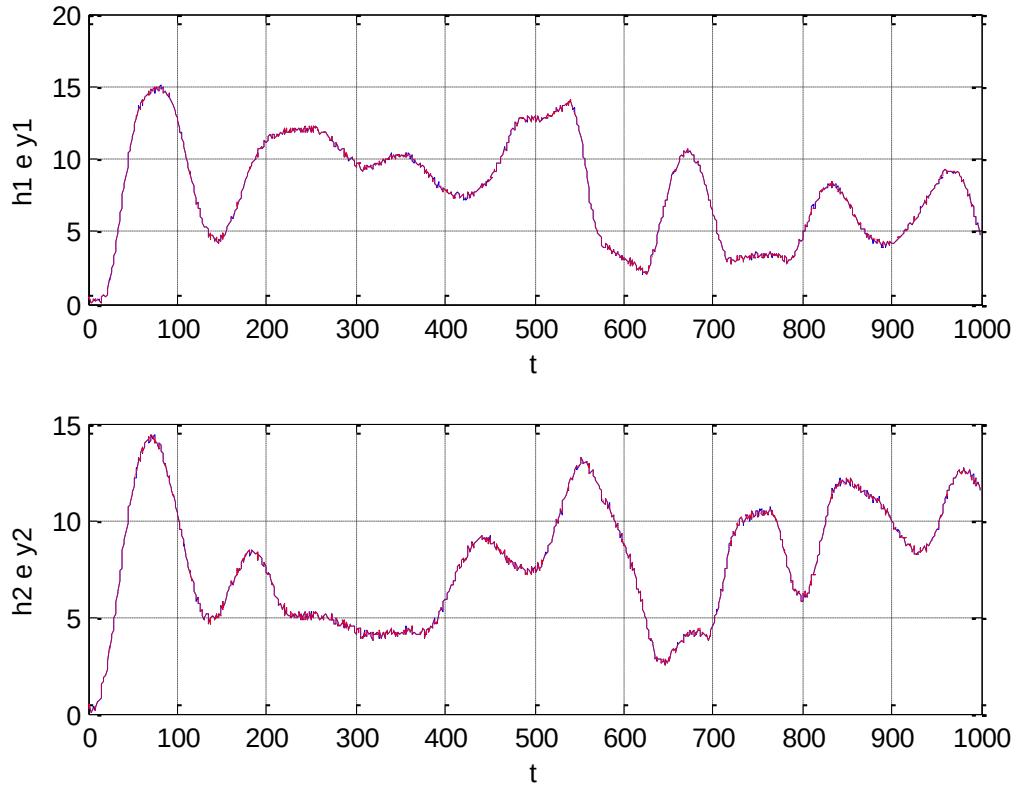

Figura 6.17 - Gráfico com o resultado para validação do modelo *fuzzy*.

Os parâmetros do modelo *fuzzy* correspondentes ao Experimento 6.2.2, são representados nas tabelas a seguir. Na Tabela 6.9 têm-se os valores dos parâmetros dos antecedentes das regras de y_{1i} e y_{2i}, e as Tabelas 6.10 e 6.11 mostram as representações tabulares das regras relativas à y_{1i} e y_{2i}. Nas Tabelas 6.12 e 6.13 estão representados os coeficientes dos consequentes das regras para y_{1i} e y_{2i}.

Tabela 6.9 - Valores dos parâmetros dos antecedentes de y_{1i} e y_{2i} do modelo *fuzzy*.

$a_n^{(g)}$	$m_n^{(g)}$	$n_n^{(g)}$	$b_n^{(g)}$
5,175	10,64	4,639	10,44
4,639	10,61	4,606	11,00
1,581	3,528	1,523	3,782
1,409	3,429	1,44	3,745

Tabela.6.10 - Representação tabular das regras relativas à y_{1i} do modelo *fuzzy*.

	$D_1^{(1)}$	$D_1^{(2)}$
$A_1^{(1)}B_1^{(1)}C_1^{(1)}$	y_{11}	y_{12}
$A_1^{(1)}B_1^{(1)}C_1^{(2)}$	y_{13}	y_{14}
$A_1^{(1)}B_1^{(2)}C_1^{(1)}$	y_{15}	y_{16}
$A_1^{(1)}B_1^{(2)}C_1^{(2)}$	y_{17}	y_{18}
$A_1^{(2)}B_1^{(1)}C_1^{(1)}$	y_{19}	y_{110}
$A_1^{(2)}B_1^{(1)}C_1^{(2)}$	y_{111}	y_{112}
$A_1^{(2)}B_1^{(2)}C_1^{(1)}$	y_{113}	y_{114}
$A_1^{(2)}B_1^{(2)}C\gamma_1^{(2)}$	y_{115}	y_{116}

Tabela 6.11 - Representação tabular das regras relativas à y_{2i} do modelo *fuzzy*.

	$\mathrm{D}_2^{(1)}$	$\mathrm{D}_2^{(2)}$
$A_2^{(1)}\mathrm{B}_2^{(1)}\mathrm{C}_2^{(1)}$	y_{21}	y_{22}
$A_2^{(1)}\mathrm{B}_2^{(1)}\mathrm{C}_2^{(2)}$	y_{23}	y_{24}
$A_2^{(1)}\mathrm{B}_2^{(2)}\mathrm{C}_2^{(1)}$	y_{25}	y_{26}
$A_2^{(1)}\mathrm{B}_2^{(2)}\mathrm{C}_2^{(2)}$	y_{27}	y_{28}
$A_2^{(2)}\mathrm{B}_2^{(1)}\mathrm{C}_2^{(1)}$	y_{29}	y_{210}
$A_2^{(2)}\mathrm{B}_2^{(1)}\mathrm{C}_2^{(2)}$	y_{211}	y_{212}
$A_2^{(2)}\mathrm{B}_2^{(2)}\mathrm{C}_2^{(1)}$	y_{213}	y_{214}
$A_2^{(2)}\mathrm{B}_2^{(2)}\mathrm{C}\gamma_2^{(2)}$	y_{215}	y_{216}

Tabela 6.12 - Coeficientes dos consequentes das regras de y_{1i} do modelo *fuzzy*.

c_{10g}	c_{11g}	c_{12g}	c_{13g}	c_{14g}
0.8623	0.04553	0.2813	0.002772	0.01446
0.8821	0.0317	0.2706	-0.01361	0.03745
0.8718	0.03967	0.2688	0.01082	0.02917
0.8810	-0.001093	0.2421	0.02942	0.1748
0.8737	0.05269	0.2634	-0.0120	-0.1161
0.8634	0.03498	0.2876	0.008339	0.07143
0.859	0.05868	0.3016	-0.03484	-0.1139
0.9055	-0.009257	0.2582	-0.06006	0.5975
0.8395	0.06495	0.2941	0.02827	0.07195
0.8841	0.06886	0.2971	-0.007948	-0.3915
-0.8696	0.04539	0.2765	-0.000499	-0.0306
0.875	0.03614	0.3018	0.001432	-0.1265
0.8732	0.04116	0.2695	-0.01179	-0.03733
0.8637	0.04299	0.2728	0.002477	0.02114
0.8699	0.0486	0.2737	0.007132	-0.08874
0.8646	0.04215	0.2766	0.009828	-0.008441

Tabela 6.13 - Coeficientes dos consequentes das regras de y_{2i} do modelo *fuzzy*.

c_{20g}	c_{21g}	c_{22g}	c_{23g}	c_{24g}
0.0378	0.8987	0.007655	0.1883	0.009964
0.05143	0.8936	0.0003021	0.1770	0.0258
0.04436	0.8947	-0.0009547	0.1939	0.0201
0.05067	0.8666	-0.01933	0.2067	0.1204
0.04568	0.9036	-0.004682	0.1781	-0.0800
0.03857	0.8914	0.0120	0.1922	0.04921
0.03549	0.9078	0.02167	0.1624	-0.07842
0.06759	0.8610	-0.008252	0.1450	0.4117
0.02208	0.9121	0.01649	0.2059	0.04963
0.0528	0.9148	0.01857	0.1809	-0.2697
0.04285	0.8986	0.004348	0.1861	-0.0211
0.04654	0.8922	0.02183	0.1874	-0.08718
0.04528	0.8957	-0.0004258	0.1783	-0.0265
0.03874	0.897	0.00181	0.1881	0.01457
0.043	0.9008	0.00244	0.1913	-0.06115
0.03938	0.8964	0.004448	0.1932	0.005815

Novamente, comparando os tempos de processamento entre o modelo aproximado e o modelo *fuzzy*, verificou-se que o tempo de processamento do modelo aproximado foi de 0,26 segundos e do fuzzy de 10,36 segundos. Além disso, os tempos para a obtenção das regras também diferem, sendo em torno de 1 segundo para o modelo aproximado e 21 segundos para o *fuzzy*. Para o modelo aproximado usou-se dois níveis de discretização e para o modelo *fuzzy* duas funções de pertinência e 50 épocas na etapa de treinamento.

Caso aumente o número de discretizações para o modelo aproximado para três, isso acarretará em um aumento do número de regras (que eram 16 e passaram para 59 regras). Porém, o resultado da aproximação entre os valores reais e do modelo aproximado serão melhores e um tempo de processamento em torno de 0,7 segundos. Já no caso do modelo

fuzzy, quando se aumenta o número de funções de pertinência em três partições, o tempo de processamento ficou em torno de sessenta e cinco segundos. Os tempos de geração das regras também diferiram, no caso do modelo aproximado ficou em torno de um segundo, porém para o modelo *fuzzy* o tempo ficou em torno de quatorze minutos. Estes resultados mostram que tanto o tempo de processamento do modelo quanto o tempo para geração de regras do modelo aproximado é muito menor do que do modelo *fuzzy*, e esta diferença se torna cada vez maior à medida que se aumenta o número de funções de pertinência do modelo *fuzzy*.

Os resultados obtidos tanto no Experimento 6.2.1 (usando dados de um exemplo numérico) quanto no Experimento 6.2.2 (usando dados reais coletados de um sistema físico), indicam que os modelos aproximados apresentam um bom potencial para modelagem e previsões em sistemas não lineares em geral.

CAPÍTULO 7

7 CONCLUSÃO

7.1 Considerações Finais

A modelagem de sistemas é muito importante em vários campos das ciências e engenharias. Os métodos clássicos de modelagem são baseados em modelos matemáticos aprimorados. No entanto, para sistemas complexos com características não lineares e/ou parâmetros variáveis no tempo, a obtenção dos modelos correspondentes não é trivial. As dificuldades geralmente aumentam em relação a sistemas com múltiplas entradas e múltiplas saídas (MIMO), comparados aos procedimentos para sistemas com entradas e saídas singelas (SISO).

A utilização de técnicas de inteligência artificial tem se mostrado promissora na modelagem desses sistemas com características não lineares.

Na prática, as informações de dados de sistemas reais são frequentemente incertas, imprecisas ou incompletas. Diversas abordagens foram desenvolvidas para tratar tais condições, dentre elas a Teoria dos Conjuntos Aproximados.

No Capítulo 5, foi apresentado o desenvolvimento de uma metodologia para modelagem baseada em regras de sistemas MIMO não lineares utilizando conceitos de conjuntos aproximados. Para representação proposta, denominada de modelagem aproximada, foi demonstrado matematicamente que os modelos aproximados resultantes constituem aproximadores universais que podem aproximar funções ou modelos dinâmicos em geral, com determinado grau de precisão (Anexo E).

Pode ser verificado que a metodologia proposta possibilita a geração de regras de modelagem de forma sistemática. Que as estimativas de valores dos modelos resultantes foram realizadas por intermédio de fórmulas de interpolações convencionais. Que os modelos baseados em regras têm a vantagem de disponibilizar o conhecimento necessário para representar um sistema de forma clara por meio de regras do tipo IF-THEN que os compõem. Essas regras podem ser implantadas em linguagens de programação de propósito geral.

Afim de, demonstrar a metodologia, foram apresentados exemplos numéricos de funções estáticas lineares e não lineares e também experimentos de sistemas dinâmicos MIMO, incluindo o exemplo de um processo real. Os resultados obtidos com a modelagem foram comparados com resultados de modelos *fuzzy*.

O primeiro experimento foi relativo a um modelo discreto de um sistema não linear com duas variáveis de entrada e duas variáveis de saída. Foi aplicada a metodologia e o resultado da modelagem apresentou-se adequado.

O segundo experimento abordou uma modelagem prática de um sistema de nível real formado por dois reservatórios acoplados, constituindo um sistema MIMO não linear com duas variáveis de entrada e duas variáveis de saída.

No desenvolvimento da modelagem, verificou-se que o resultado poderia depender do nível e do método de discretização utilizado no *SI*. Então para o segundo experimento foi analisado dois métodos de discretização do sinal (*Equal frequency Binning* e *Boolean Reasoning*) e também dois níveis (intervalos) de discretização das variáveis do *SI* correspondente, e consequentemente o número de regras do modelo resultante. Com os resultados, verificou-se que mantendo o mesmo método de discretização (*Equal Frequency Binning*) e aumentando apenas o número de intervalos de discretização, aumentava o número de regras do modelo, em contrapartida a aproximação do modelo melhorava tanto para os dados de treinamento quanto para os dados de validação. Usando outro médoto de discretização (*Boolean Reasoning*), tanto para os dados originais de treinamento, quanto para os dados de validação, este se mostrou menos eficiente em relação ao primeiro método (*Equal frequency Binning).*

O custo computacional (em tempo de execução) para a geração das regras de modelagem, e para o processamento numérico das mesmas usando a metodologia proposta baseada em conjuntos aproximados, mostraram-se menores em relação aos modelos *fuzzy* utilizados, pois não há a necessidade de etapas de treinamento (como nas estruturas *ANFIS* utilizadas nas comparações de resultados), e de procedimentos de *fuzificação* e *defuzificação* (típicos de modelos *fuzzy*). Esta característica é importante em aplicações de tempo real, como em sistemas de controle e em sistemas embarcados, por exemplo. Lembrando que para a geração das regras para os modelos aproximados considerados foram usados os programas Rosetta e o NetBeans, sendo que para os modelos *fuzzy* empregou-se o *toolbox* ANFIS do MatLab, mas em ambos os experimentos as simulações das regras resultantes foram realizadas no ambiente MatLab.

Os resultados obtidos em todos os experimentos estudados indicaram que o objetivo proposto neste trabalho foi atingido e apresentaram precisões adequadas, seja com dados provenientes de funções matemáticas conhecidas, ou com informações oriundas de simulações de modelos computacionais, assim como de dados provenientes de ensaios de um sistema real.

7.2 Trabalhos Futuros

Como trabalhos futuros são sugeridos os seguintes tópicos para investigação:

- Aplicações da metodologia desenvolvida para a modelagem de outros sistemas dinâmicos com entradas e saídas adicionais, particularmente em sistemas MIMO reais;
- Investigar a influência de diferentes procedimentos de discretização na precisão dos modelos aproximados correspondentes.
- Desenvolvimento de aplicações para sistemas de controle MIMO baseados em regras utilizando a abordagem desenvolvida. Utilização de abordagens semelhantes aos sistemas de controle *fuzzy*, onde a partir de modelos baseados em regras de um determinado processo são geradas as regras do controlador correspondente.

REFERÊNCIAS BIBLIOGRÁFICAS

AGUIRRE, L. A., MENDES, E. M. A. M.. Modeling chaotic dynamics with discrete nonlinear rational models. International Journal of Bifurcation and Chaos in Applied Sciences and Engineering, Singapura, v. 10, n. 5, p. 1019-1032, 2000.

BAY, S. D. Multivariate discretization of continuous variables for set mining. In: ACM Sigkdd International Conference on Knowledge Discovery and Dada Mining, 6, Boston. Proceedings Boston: ACM, p.315-319, 2000.

BONALDI, E. L. Diagnóstico preditivo de Avarias em Motores de Indução Trifásicos com MCSA e Teoria dos Conjuntos Aproximados, Tese de doutorado, UNIFEI, Itajubá MG, 2005.

BROWN, F. M. Boolean Reasoning: The logic of Boolean Equations. Dover Books on Mathematics. 1998.

CAI, Z.; GONG, R. Rough set theory and its application to neuro-fuzzy modeling. International Conference on Control and Automation ICCA. p. 105-106, 2002.

CAMATTA, U. P. Projeto e Implementação de um regulador de velocidade e corrente baseado na técnica de conjuntos aproximados (Rough Sets), Dissertação de mestrado, UNIFEI, Itajubá MG, 2009.

CARVALHO, JAQUELINE C. S. Aplicação de conjuntos aproximados utilizando banco de dados relacionais, Dissertação de mestrado, UNIFEI, Itajubá MG, 2000.

CARVALHO, M.A.. Discretização de atributos contínuos em sistemas de informação utilizando algoritmos genéticos para a aplicação da teoria dos conjuntos. Dissertação de mestrado em Ciência e Tecnologia da Computação. Universidade Federal de Itajubá, 2010.

CASTILHO, O.; MELIN, P. Hibrid intelligent systems for time series prediction using neural networks, fuzzy logic, and fractal theory. IEEE, n. 6, p. 1395–1408, 2002.

CERCHIARI, S. C. Determinação de curvas típicas de demanda de consumidores de baixa tensão utilizando Mapas Auto-Organizáveis (SOM) para Agrupamentos e Conjuntos Aproximados para Classificação de Consumidores. Dissertação de mestrado da UFMS, 2006.

CHEN, D.W.; ZHANG, J.P.. Time series prediction based on ensemble ANFIS. Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, p. 18–21, 2005.

CHEN, M.-S.; YING, L.-C.; PAN, M.-C. Forecasting tourist arrivals by using the adaptive network-based fuzzy inference system. Expert Systems with Applications, v. 37, p. 1185–1191, 2010.

COUTINHO, M. P. Detecção de ataques em infraestruturas críticas de sistemas elétricos de potência usando técnicas inteligentes. Tese de doutorado, UNIFEI, Itajubá MG, 2007.

DAVIS, P. J.Interpolation & Approximation. Dover publications Inc., New York, p. 123-126, 1975.

DEMPSTER, A. P.. Upper and lower probabilities induced by a multivalued mapping (1967). The annals of mathematical statistics, v. 38, n. 2, p. 325-339, April 1967.

DENAÏ, M. A.; PALIS, F.; ZEGHBIB, A.. Anfis based automatic voltage regulator with hybrid learning algorithm. IEEE Transaction on Systems, Man and Cybernetics, p. 397–401, 2007.

DENAÏ, M.A.; PALIS, F; ZEGHBIB, A.. ANFIS based modelling and control of non-linear systems: a tutorial. SMC (4), p.3433-3438, 2004.

DENAÏ, M.A.; PALIS, F; ZEGHBIB, A.. Modeling and control of non-linear systems using soft computing techniques. Elsevier Science Publishers B. V. Applied Soft Computing. v.7**,** June 2007.

DUBOIS, D.; PRADE, H.; YANGER, R. R. Fuzzy Information Engineering: A Guided Tour of Applications. 1st. ed. [S.l.: s.n.], 1996.

DUN, L.; HUAXIONG, L.; XIANZHONG, Z.. Two decades' research on decision-theoretic Rough Sets . 9th IEEE International Conference on Cognitive Informatics (ICCI), p. 968 – 973, 2010.

FAHIMIFARD, S. et al. Application of ANFIS to agricultural economic variables forecasting. case study: Poultry retail price. Journal of Artificial Intelligence, v. 2, n. 2, p. 65–72, 2009.

FAUSTINO, C. P. Previsão de séries temporais via modelos baseados em regras, Dissertação de mestrado, UNIFEI, Itajubá MG, 2011.

FAUSTINO, C. P.; PINHEIRO, C.; CARPINTEIRO, O.; LIMA, I. . Time series forecasting through rule-based models obtained via rough sets. Artificial Intelligence Review, v. 1, p. 1-12, 2011.

FAUSTINO, C.P, NOVAES, C.P, PINHEIRO, C., CARPINTEIRO, O. Improving the performance of fuzzy rules-based forecasters through application of FCM algorithm ARTIFICIAL INTELLIGENCE REVIEW, 2012.

FAYYAD, U., IRANI, K.. Multi-interval Discretization of Continuous valued Attributes for Classification Learning. In Thirteenth International Joint Conference on Artificial Intelligence (IJCAI), p.. 1022–1027. Morgan Kaufmann, 1993.

FIRAT, M.; GÜNGÖR, M. River flow estimation using adaptive neuro fuzzy inference system. Mathematics and Computers in Simulation, v. 75, p. 87–96, 2007.

GOH, C.; LAW, R., Incorporating the rough sets theory into travel demand analysis. Tourism Management, v. 24, p. 511–517, 2003.

HAYKIN, S. Neural Networks: Principles and Pratices. 1st. ed [S.1]. Bookman, 2000.

HENRIQUES, S. B. Classificador de sinais de voz usando conjuntos aproximados. Dissertação de mestrado, UNIFEI, Itajubá MG, 2001.

HERBERT, J.; YAO, J. Time series data analysis with rough sets. Applied Soft Computing, v. 9, p. 1000–1007, 2009.

HUANG, B.; GUO, L. ; ZHOU, X. Approximation Reduction Based on Similarity Relation. IEEE Fourth International Conf. on Fuzzy Systems and Knowledge Discovery, pp. 124-128, 2007.

ILCZUK, G.; WAKULICZ, A.D.. Visualization of Rough Set Decision Rules for Medical Diagnosis Systems. RSFDGrC '07 Proceedings of the 11th International Conference on Rough Sets, Fuzzy Sets, Data Mining and Granular Computing. Springer-Verlag Berlin, Heidelberg, p.371 – 378, 2007.

JANG, J. S. R.; SUN, C. T.; MIZUTANI, E.. Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence. 1st. ed. [S.l.]: Prentice Hall, 1997.

JANG, J.S. R.; SUN, C.T.. Neuro-fuzzy modeling and control. Proceedings of the IEEE, v. 83, p. 378–406, 1995.

JANG, J.S.R.. ANFIS: Adaptive-Network-based Fuzzy Inference System. IEEE Transaction on Systems, Man and Cybernetics, v. 23, n. 3, p. 665–685, 1993.

KASABOV, N.; SONG, Q.. DENFIS: Dynamic Evolving Neural-Fuzzy Inference System and its application for time-series prediction. IEEE Transactions on Fuzzy Systems, 2001.

KLIR, G. J.; YUAN, B.. Fuzzy Sets and Fuzzy Logic: Theory and Applications. 1st. ed. [S.l.]: Prentice Hall, 1995.

KURIAN, C. P. et al. ANFIS model for the time series prediction of interior daylight illuminance. AIML Journal, v. 6, 2006.

KUSIAK, A.; SHAH, A. Data-Mining-Based System for Prediction of Water Chemistry Faults. IEEE Transactions on Industrial Electronics, No. 2, p. 593- 596, 2006.

LI, R.J.; XIONG, Z.B.. Forecasting stock market with fuzzy neural networks. Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, p. 18–21, 2005.

LIAO, G.C.; TSAO, T.P.. Application of fuzzy neural networks and artificial intelligence for load forecasting. Electric Power Systems Research, v. 70, p. 237–244, 2007.

LIUYANG, Z.; YUWEN, S.; PENGCHENG, T.; HUI, Z.. A fault diagnosis modeling method combined RBF neural network with rough set theory. ISECS International Colloquium on Computing, Communication, Control, and Management, 2009. CCCM 2009, p.501 - 504, Aug. 2009.

LOTFABADI, M.S.; MOGHADAM, A.M.E. The Comparison of Different Feature Decreasing Methods Based on Rough Sets and Principal Component Analysis for Extraction of Valuable Features and Data Classifying Accuracy Increasing. First International Conference on Integrated Intelligent Computing (ICIIC), p.108-113, 2010.

MING-BAO, P.; XIN-PING, Z. Traffic flow prediction of chaos time series by using subtractive clustering for fuzzy neural network modeling. Second International Symposium on Intelligent Information Technology Application, 2008.

MITRA, P. et al.. Anfis based modelling and control of non-linear systems: A tutorial. Universities Power Engineering Conference, 2007. UPEC 2007. 42nd International, 2004.

MITRA, S.; PAL, S. K. Self-organizing neural network as a fuzzv classifier. IEEE Transactions on Systems, Man, and Cybernetics, v. 24, n. 3, p. 385–399, 1994.

NAUCK, D.; KRUSE, R.. Nefclass - a neuro-fuzzy approach for the classification of data. Proceedings of the 1995 ACM symposium on Applied computing, p. 461–465, 1995.

NAYAK, P. et al. A neuro-fuzzy computing technique for modeling hydrological time series. Journal of Hydrology, v. 291, p. 52–66, 2004.

NGUYEN, S.; SKOWRON, A. (1995). Quantization of real value attributes. Proceedings. of the Second Joint Annual Conference on Information Sciences, p. 37-40.

OHRN, A.; KOMOROWSKI, J. Rosetta - a rough sets toolkit for analyses of data. Third International Joint Conference on Information Sciences, p. 403–407, 1997. http://www.idi.ntnu.no/~aleks/rosetta/.

PAWLAK, Z. et al. Rough sets. Communications of ACM, v. 38, p. 89–95, 1995.

PAWLAK, Z. Rough set theory and its applications. Journal of Communications and Information Technology, p. 7–10, 2003.

PAWLAK, Z. Rough sets. International Journal of Information and Computer Sciences, v. 11, p. 341–356, 1982.

PAWLAK, Z. Rough sets: theoretical aspects of reasoning about data. London, Kluwer, 1991.

PAWLAK, Z., MUNAKATA, T. Rough Control: Application of Rough Set Theory to control. Proceeding of the Fourth European Congress on Intelligent Techniques and Soft Computing (EUFIT'96), pp 209-218, 1996.

PAWLAK, Z.; SKOWRON, A. Rudiments of rough sets. Information Sciences, v. 117, p.3–27, 2007.

PEDRYCZ, W., GOMIDE, F. Fuzzy Systems Engineering: Toward Human Centric Computing. Wiley Interscience/IEEE, Hoboken, 2007.

PESSOA, A.S. A.. Mineração de dados meteorológicos pela teoria dos conjuntos aproximativos na previsão de clima por redes neurais artificiais. Dissertação de Mestrado do Curso de Pós-Graduação em Computação Aplicada, http://urlib.net/sid.inpe.br/jeferson/2005/02.15.15.46, INPE, 2004.

PINHEIRO, C. Projeto de controladores fuzzy via rough sets. IX Congresso Brasileiro de Automação Inteligente – SBAI'09, em CD, 2009.

PINHEIRO, C., GOMIDE, F., CARPINTEIRO, O., LIMA, I. Modelos baseados em conjuntos aproximados, XVIII Congresso Brasileiro de Automática, Bonito, 2010_a.

PINHEIRO, C., GOMIDE, F., CARPINTEIRO, O.; LOPEZ, B. Granular Synthesis of Rule-Based Models and Function Approximation using Rough Sets. Chapter in the book Novel Developments in Granular Computing, ed. JingTaoYao, Information Science Publishing, 2010_b.

PINHEIRO, C., CAMATTA, U., REZEK, A. Rough Controller Synthesis. Chapter in the book Fuzzy Logic: Controls, Concepts, Theories and Applications, ed. Elmer P. Dadios, INTECH, 2012.

PINHEIRO, C., OLIVEIRA, R. Rule-Based Models via Rough Sets in the Approximation of Systems with Multiple Inputs and Outputs. Accepted to Journal of Experimental & Theoretical Artificial Intelligence, 2013_a.

PINHEIRO, C., OLIVEIRA, R. Modeling Based on Rough Sets to MIMO Nonlinear Coupled Tanks. ICADE - 1^{st} International Conference on Intelligent Control and Applications (ICADE), Paris, France, 2013_b.

PINHEIRO, C., OLIVEIRA, R. Rule-Based Model via Rough to MIMO Discrete-Time Nonlinear Dynamical Systems. 8^{th} International Conference of Circuits, Systems, Signal and Telecommunications (CSST), Tenerife, Spain, 2014.

RISSINO, S. D. Metodologia de avaliação da relevância de atributos em grandes bases de dados incompletas utilizando Conjuntos Aproximados e Lógica Paraconsistente. Dissertação de mestrado, UNIFEI, Itajubá MG, 2009.

RODOR, F. F. Aplicação de conceitos de conjuntos aproximados na adaptação da constante de filtro de controladores IMC. Dissertação de mestrado, UNIFEI, Itajubá MG, 2012.

ROSSI, R. Classificador hierárquico sistêmico para redes elétricas de alta tensão. Tese de doutorado, UNIFEI, Itajubá MG, 2000.

SAKAI, H. , NAKATA, M. On Rough sets based rule generation from tables. International Journal of Innovative Computing, Information and Control, 2, 3-31, 2006.

SANKAR, K. P., MITRA, P. Multispectral image segmentation using the rough-set-initialized EM algorithm. IEEE Transactions on Geoscience and Remote Sensing, 40, 2495-2501, 2002.

SANTOS, P. Metodologia de otimização do aproveitamento da energia de painéis fotovoltaicos sombreados usando a teoria dos conjuntos aproximados. Dissertação de mestrado, UNIFEI, Itajubá MG, 2009.

SASSI, R. J. Uma arquitetura híbrida para descoberta de conhecimento em base de dados: Teoria dos Rough sets e Redes Neurais artificiais Mapas auto-organizáveis. Tese de doutorado da USP SP, 2006.

SHAFER, G.. A Mathematical Theory of Evidence. Princeton University Press, 1976.

SHEN, L.; LOH, H. T. Applying rough sets to market timing decisions. Decision Support Systems, 2003.

SKOWRON, A., NGUYEN, H. S. Boolean reasoning scheme with some applications in data mining. Springer-Verlag, Berlin, pp 107-115, 1999.

SYED-AHMAD, M. N. et al. Short-Term Load Forecasting Using Adaptive Neuro-Fuzzy Inference System (ANFIS). Application to aleppo load demand. 2007.

TAJIRI, L. L. Proposição de um controlador digital para Conversores Buck e Boost usando a teoria dos conjuntos aproximados. Dissertação de mestrado, UNIFEI, Itajubá MG, 2009.

TAY, F. E.; SHEN, L. Economic and financial prediction using rough sets model. European Journal of Operational Research, v. 141, p. 641–659, 2002.

VIEIRA, J. M. Aplicação de técnicas inteligentes no auxílio à operação dos centros de controle. Dissertação de mestrado da UFSCAR, 2005.

WANG, W., HOU, Z., JIN, S. Model-Free Indirect Adaptive Decoupling Control for Nonlinear Discrete-Time MIMO Systems. Joint 48th IEEE Conference on Decision and Control and 28th Chinese Control Conference Shanghai, P.R. China, 2009.

YING, L.C.; PAN, M.C.. Using adaptive network based fuzzy inference system to forecast regional electricity loads. Energy Conversion and Management, v. 49, p. 205–211, 2008.

YUN,G.;YUANBIN,H..Application of rough set theory on system modeling. WCICA 2004

Fifth World Congress on Intelligent Control and Automation, v.3, p.2352-2354, 2004.

ZADEH, L. A. Fuzzy sets. Information and Control, v. 8, p. 338–353, 1965.

ZADEH, L. A.. Fuzzy logic = computing with words. IEEE Transactions on Systems, Man, and Cybernetics, v. 4, n. 2, p. 103–111, 1996.

ZADEH, L.A..Outline of a new approach to the analysis of complex systems and decision processes". IEEE Trans. Systems, Man and Cybernetics, v. 3, p .28–44, 1973.

ZIARKO, W.; Katzberg, J. D.. Rough sets approach to system modeling and control algorithm acquisition. IEEE Conference: Communications, Computers and Power in the Modern Environment, pp. 154- 164, 1993.

ZIMMERMANN, H. J. Fuzzy Sets and Fuzzy Logic: Theory and Applications. 4st. ed. [S.1]: Printice Hall, 2001.

ANEXO A

O SOFTWARE ROSETTA

O software Rosetta é uma ferramenta que processa sistemas de informação genéricos utilizando conceitos associados com a Teoria dos Conjuntos Aproximados. O utilitário aceita dados no formato de tabelas de bancos de dados (Access, planilhas do Excel, e outros.), assim como arquivos em formato de texto (extensão txt).

Como ilustração será utilizado um exemplo de Pinheiro et al. (2010_a) referente à geração de regras de decisão do modelo de um sistema estático não linear $h(x)$ descrito pela curva apresentada na Figura A-1, cuja tabela de dados está ilustrada na Figura A-2

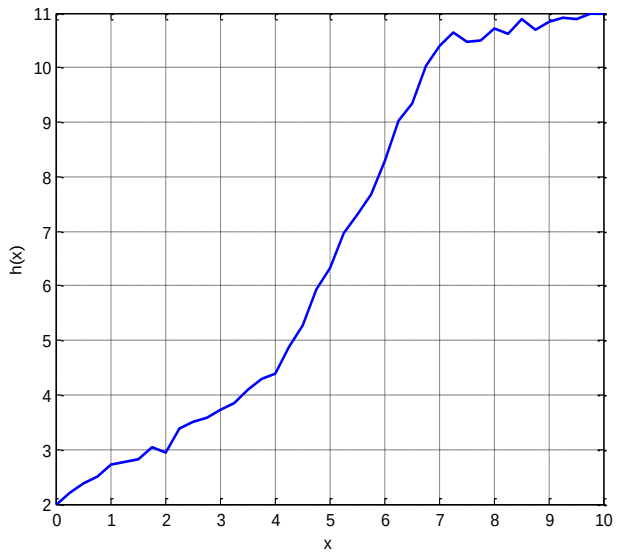

Figura A-1. Sistema não linear.

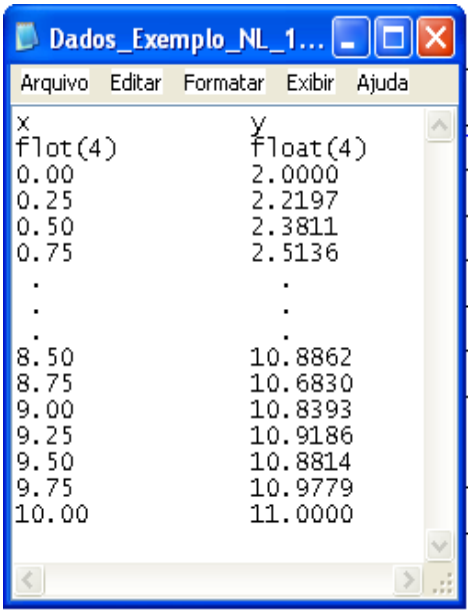

Figura A-2. Dados do sistema não linear.

Para carregar os dados da tabela no Rosetta realizar os seguintes passos:

1. Acionar o software ROSETTA;
2. Clicar em File → Open (Figura A-3);
3. Selecionar o tipo de planilha de dados (Figura A-4);
4. Abrir o arquivo de dados (Figura A-5);
5. A janela Import Structure será aberta e a opção Plain Format deve ser escolhida.

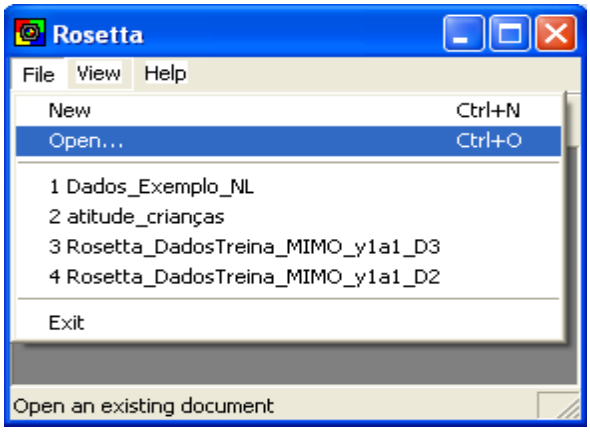

Figura A-3. Abrir arquivo de dados.

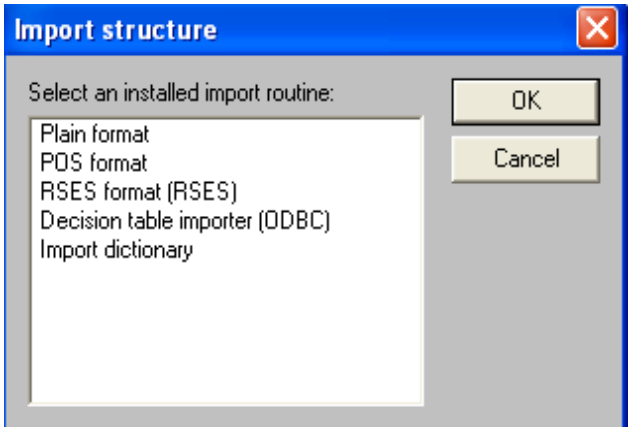

Figura A-4. Selecionar o tipo de planilha dos dados.

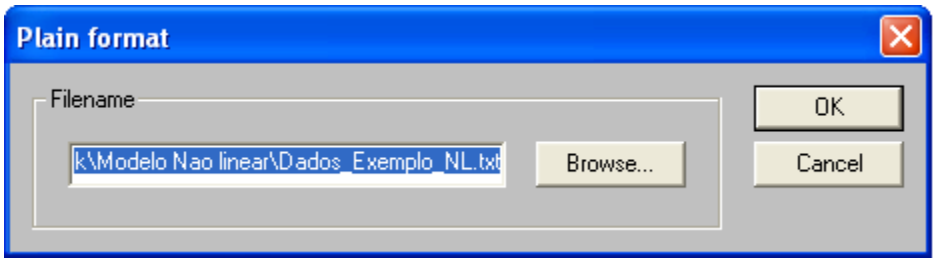

Figura A-5. Abrir o arquivo de dados.

O arquivo de dados pode ser renomeado a qualquer momento, bastando selecioná-lo e acionar uma vez com o botão esquerdo do mouse (Figura A- 6).

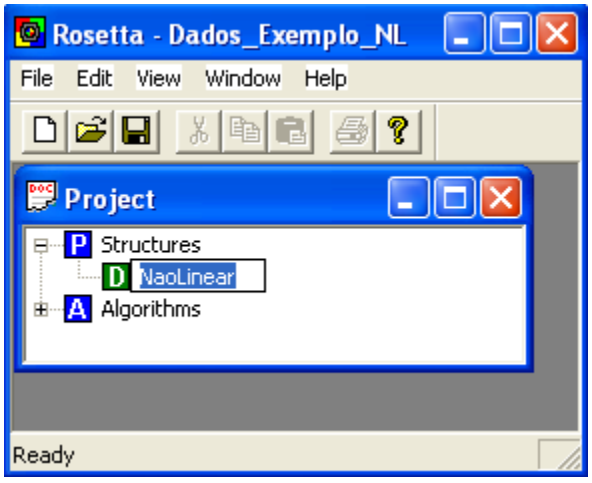

Figura A- 6 – Renomear o arquivo de dados.

A visualização dos dados pode ser feita usando a opção View conforme indicado nas Figura A-7 e Figura A-8.

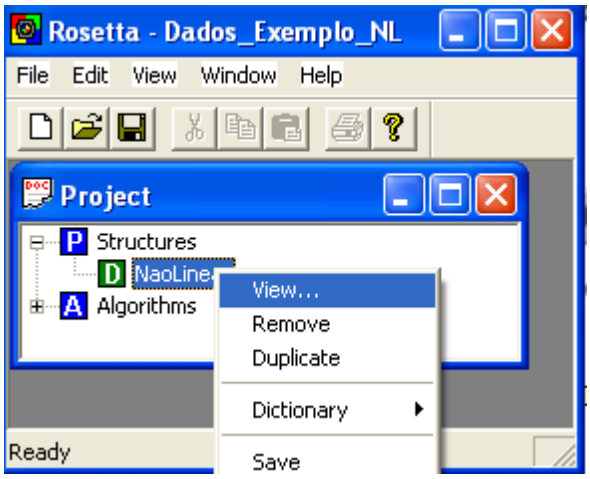

Figura A-7. Visualização do *SI*.

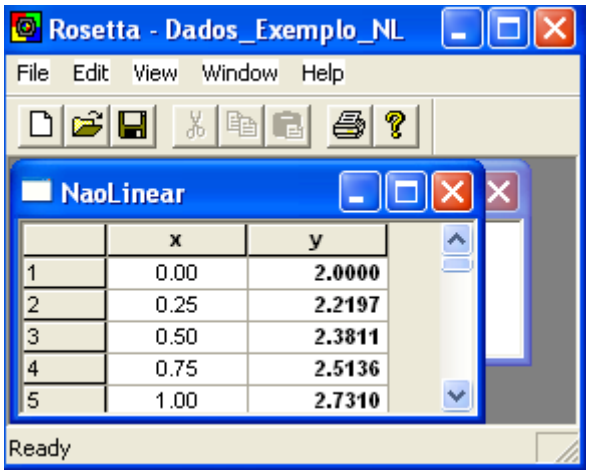

Figura A-8. Visualização dos dados.

Como os dados trabalhados são do tipo *float* é necessário fazer a discretização dos mesmos. Neste exemplo foram feitas discretizações em intervalos de freqüência regular e outros algoritmos certamente apresentarão respostas diferenciadas. Para iniciar a discretização dos dados da tabela basta seguir os passos:

1. Algorithms → Discretization → Equal Frequency Scaler (Figura A-9 e Figura A-10);
2. No campo Discretize and save cuts to file, deve ser indicado um nome para o arquivo de discretização. Em Advanced parameters, deve ser escolhido o intervalo de discretização, neste caso três (Figura A-11);
3. Acionar OK.

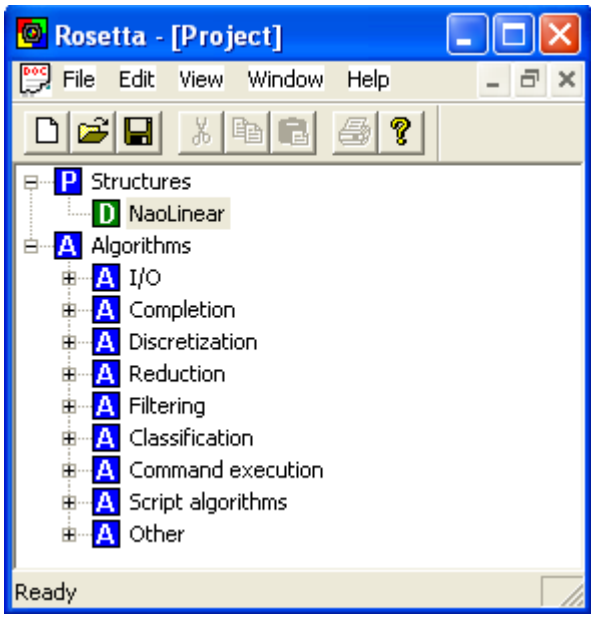

Figura A-9. Expansão do menu Algorithms.

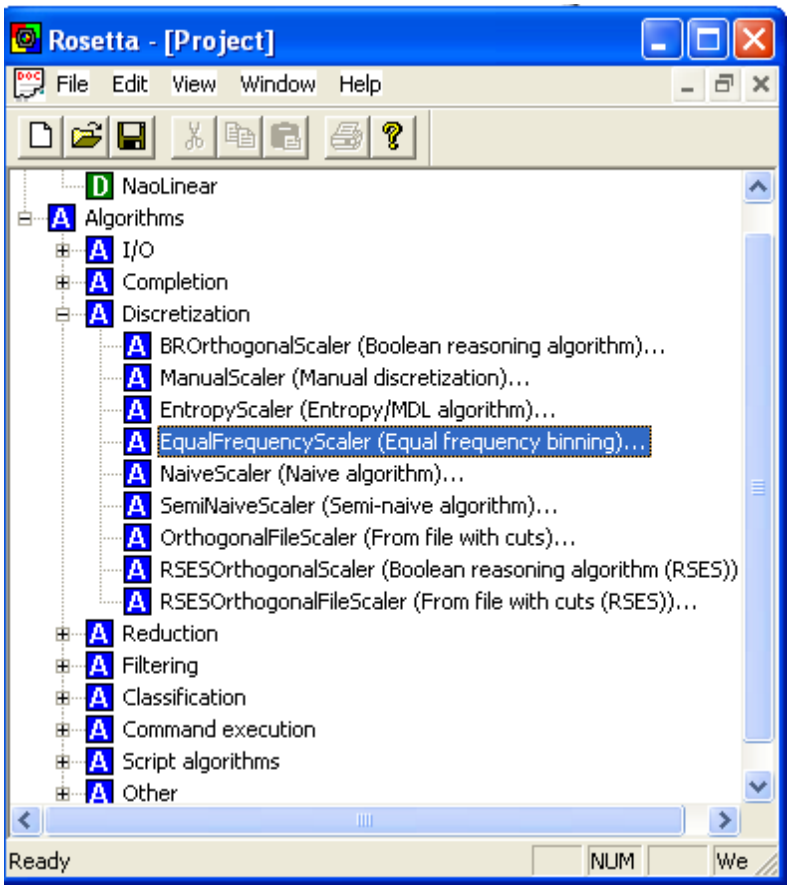

Figura A-10. Seleção do método de discretização.

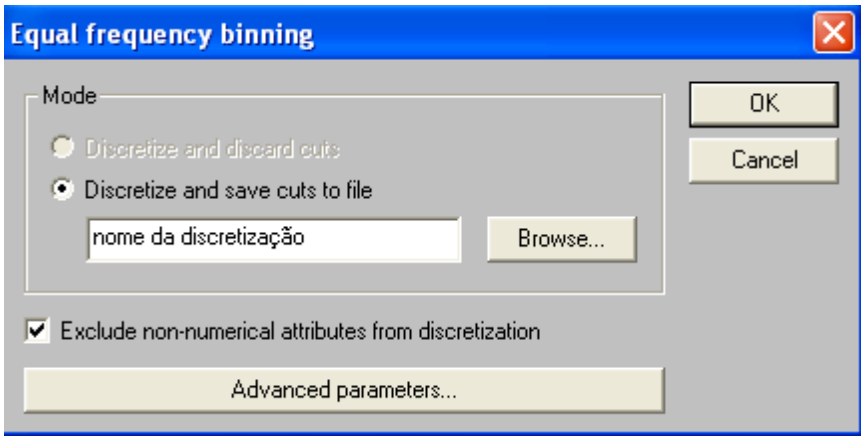

Figura A-11. Tabela de opções.

Após a conclusão dos passos citados, são gerados dois arquivos contendo as informações dos dados discretizados. Para visualização da tabela contendo os dados discretizados basta utilizar a opção View (Figura A-12).

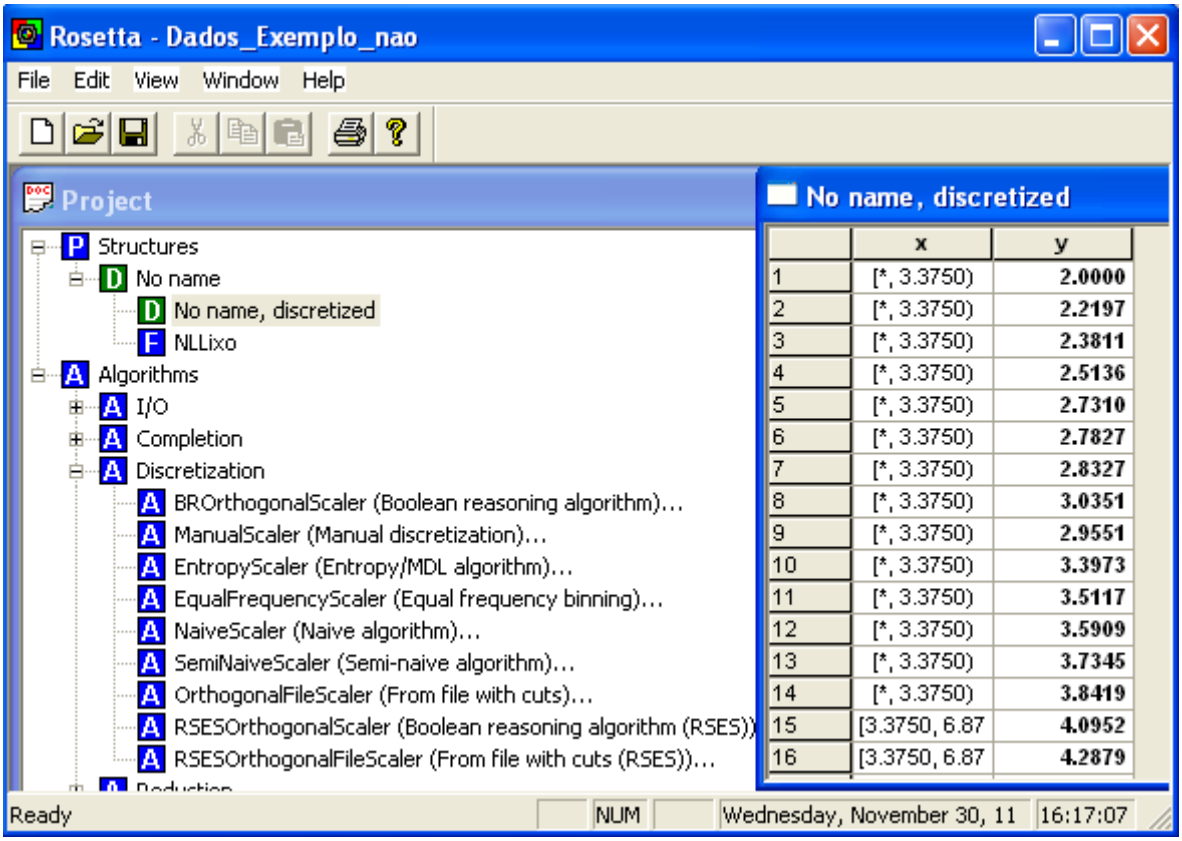

Figura A-12. Visualização dos dados discretizados.

O próximo passo é a obtenção dos redutos, que se dá a partir das etapas a seguir:

1. Algorithms → Reduction → Exhaustive calculation (RSES) (Figura A-13);
2. Na janela Exhaustive calculation, selecionar a opção full para o campo discernibility (Figura A-14);
3. Acionar OK.

Para a geração das regras de decisão: Algorithms → Other → Rule generator (RSES) (Figura A-15). As regras geradas podem ser visualizadas através da opção View (Figura A-16) e exportadas em diferentes formatos. Para isso clica-se com o botão esquerdo do mouse em cima do arquivo e escolhe-se a opção desejada como mostra a Figura A-17.

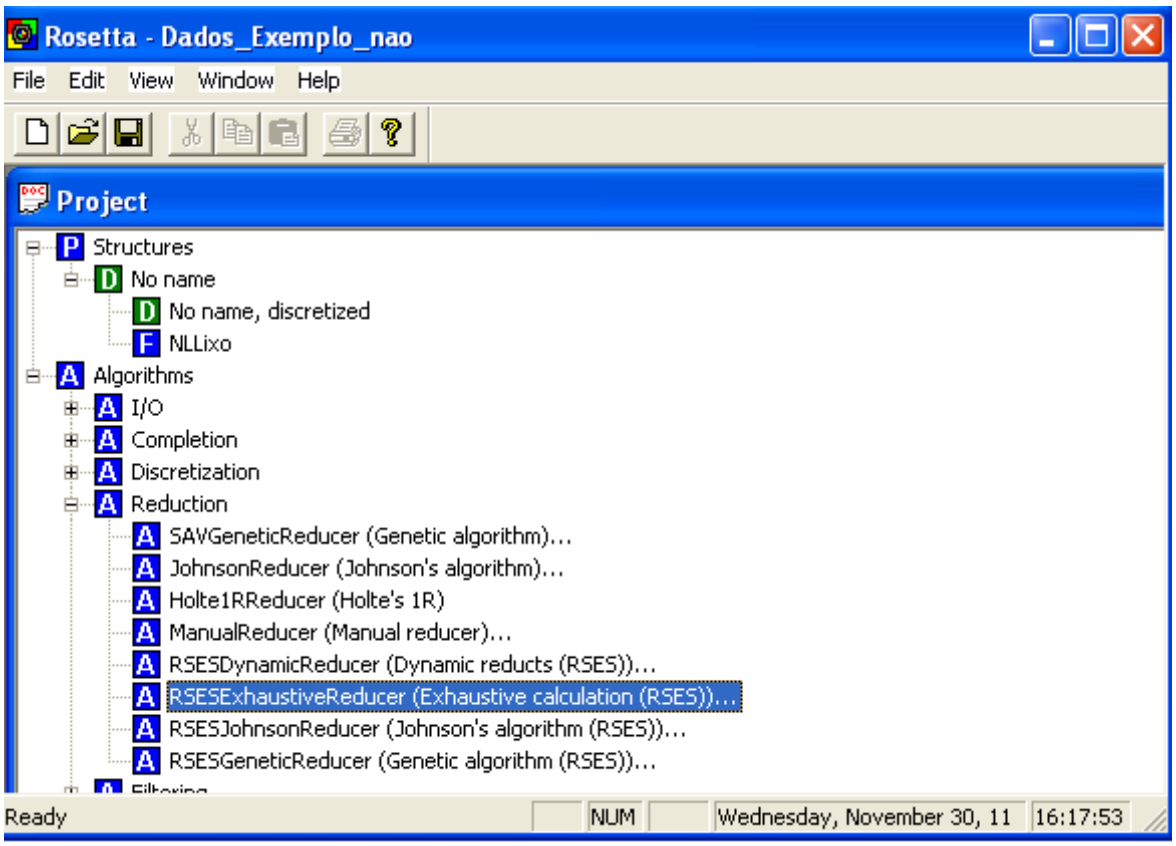

Figura A-13. Janela de obtenção dos redutos.

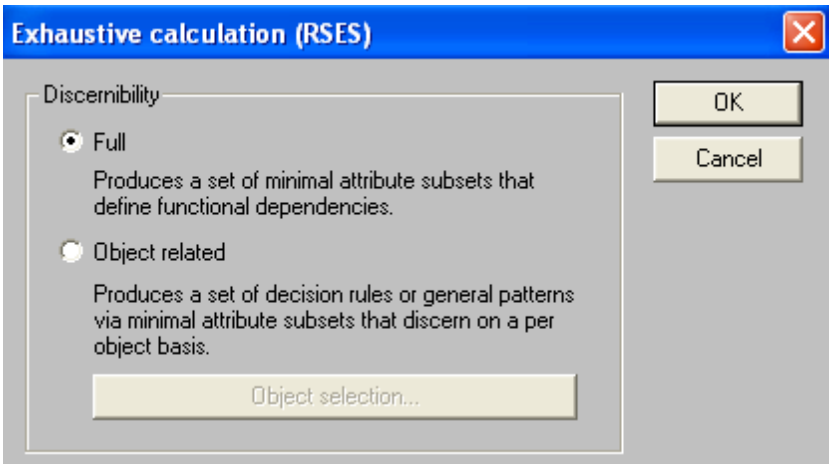

Figura A-14. Tabela de opções para Exhaustive calculation.

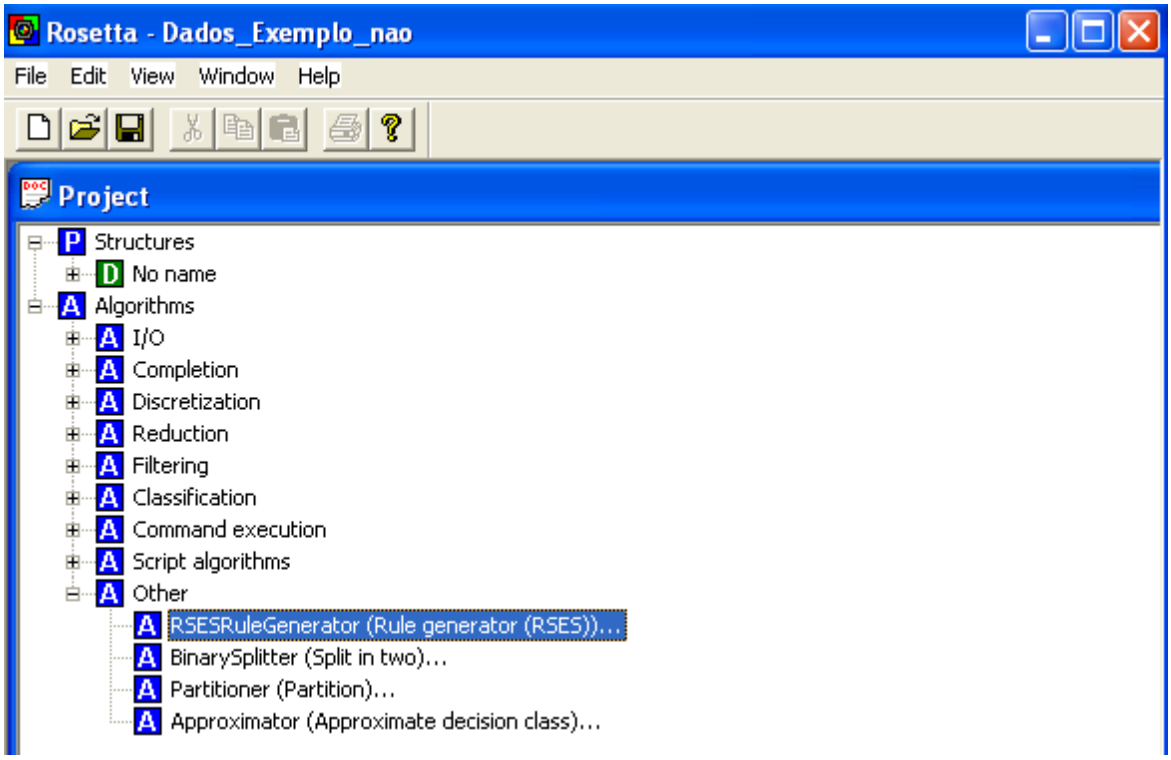

Figura A-15. Geração das regras de decisão.

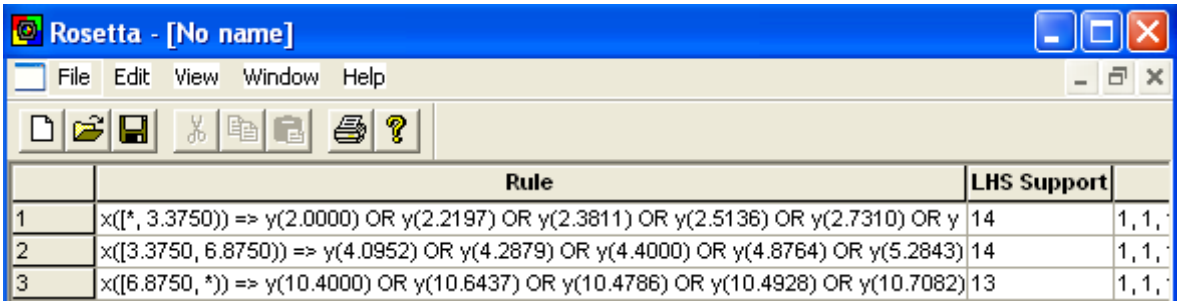

Figura A-16. Visualização das regras geradas.

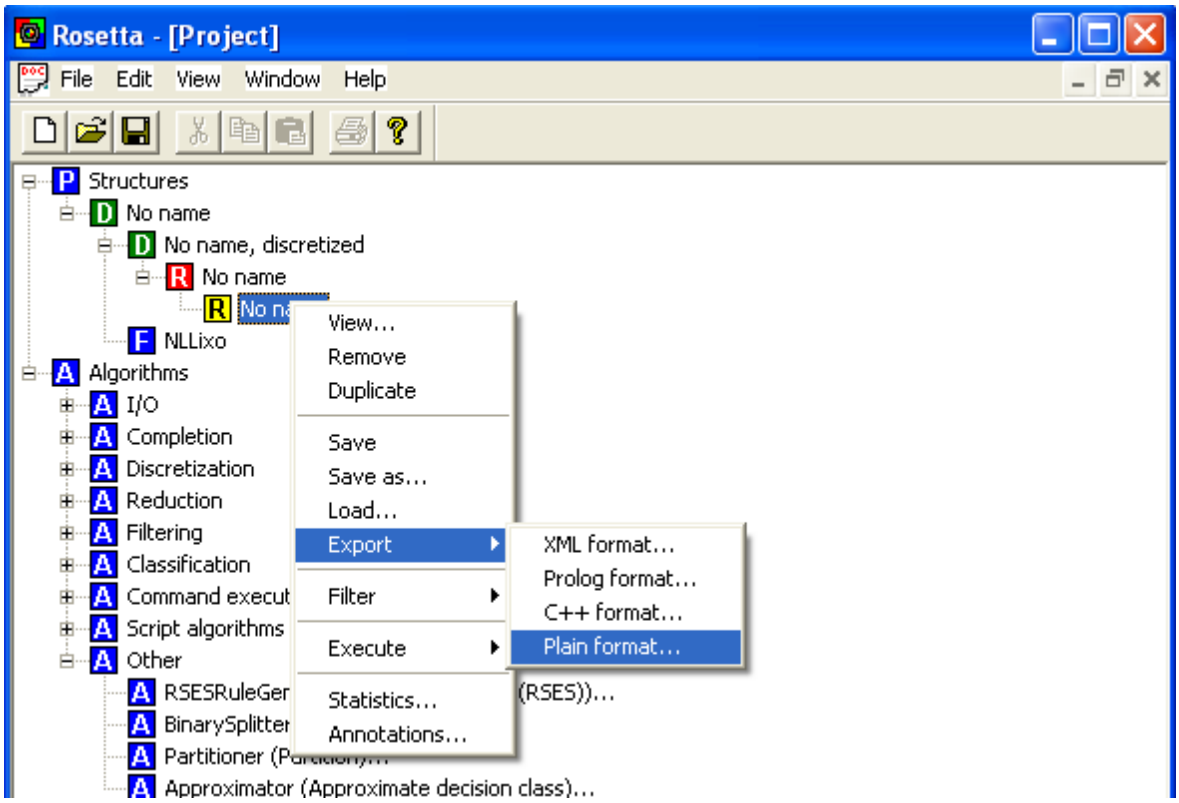

Figura A-17. Exportar arquivos de regras.

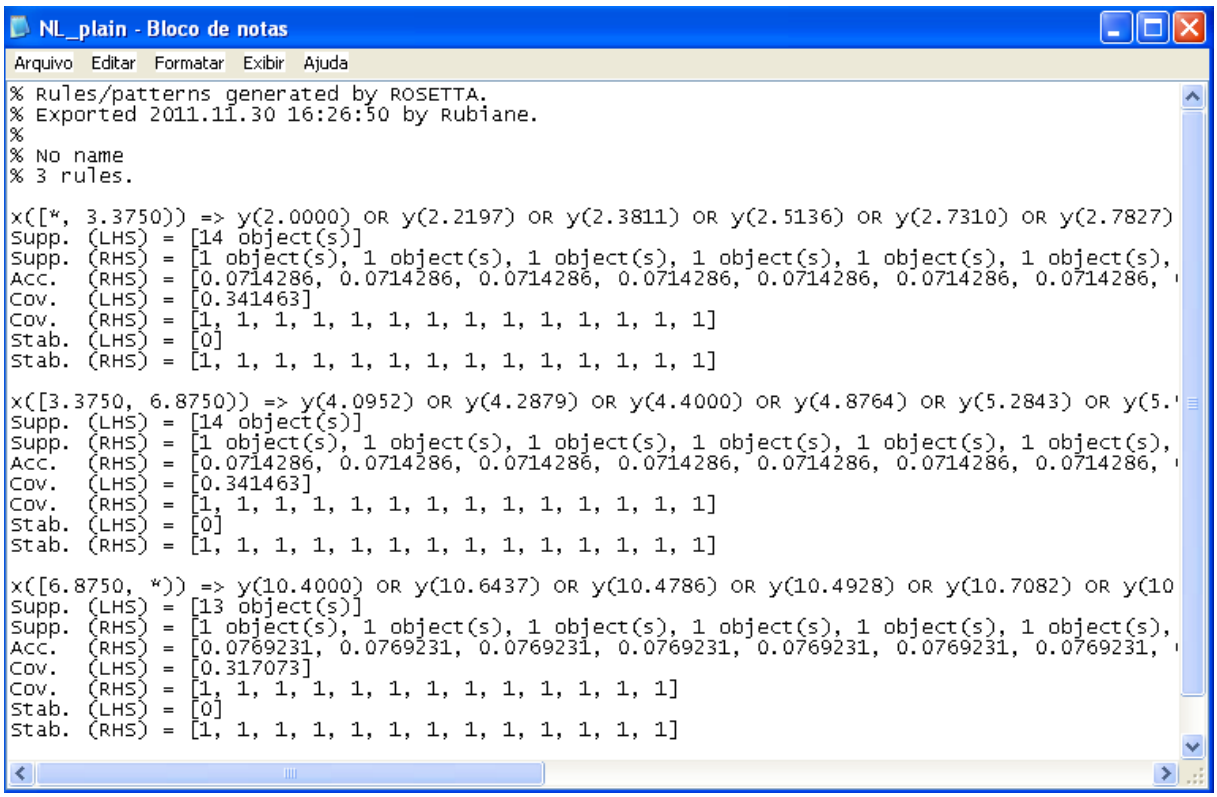

Figura A-18. Regras exportadas para um arquivo de texto.

Para a tabela descrita na Figura A-2 foram geradas três regras de decisão ilustradas abaixo. Estas regras são representadas na forma proposta neste trabalho para serem interpoladas e gerar valores numéricos nos valores de interesse em questão. Assim é possível escrever um modelo baseado em regras do sistema não linear representado na Figura A-1.

x([*, 3.3750)) => y(2.0000) OR y(2.2197) OR y(2.3811) OR y(2.5136) OR y(2.7310) OR y(2.7827) OR y(2.8327) OR y(3.0351) OR y(2.9551) OR y(3.3973) OR y(3.5117) OR y(3.5909) OR y(3.7345) OR y(3.8419)

x([3.3750, 6.8750)) => y(4.0952) OR y(4.2879) OR y(4.4000) OR y(4.8764) OR y(5.2843) OR y(5.9241) OR y(6.3302) OR y(6.9608) OR y(7.3044) OR y(7.6791) OR y(8.2819) OR y(9.0139) OR y(9.3387) OR y(10.0420)

x([6.8750, *)) => y(10.4000) OR y(10.6437) OR y(10.4786) OR y(10.4928) OR y(10.7082) OR y(10.6233) OR y(10.8862) OR y(10.6830) OR y(10.8393) OR y(10.9186) OR y(10.8814) OR y(10.9779) OR y(11.0000)

ANEXO B

PROGRAMA PARA TRATAMENTO DAS REGRAS GERADAS PELO ROSETTA

A fim de facilitar o uso do método baseado em conjuntos aproximados apresentado neste trabalho, tornando-o o mais automatizado possível, foi desenvolvido um programa cuja finalidade é processar as regras geradas pelo Rosetta deixando-as no formato representado na equação (5.2).

A saída do programa consiste em um arquivo de texto cujos dados define uma matriz onde cada linha representa uma regra do modelo aproximado. Considerando *n* o número total de dados em cada linha da matriz, os $n - 2$ primeiros termos representam os antecedentes das regras. Cada par desses *n* primeiros especifica o intervalo em que uma variável está inserida. Os dois últimos dados de cada linha definem o intervalo o qual o atributo de decisão está inserido.

Este arquivo de saída possui um formato compatível com os arquivos reconhecidos pelo MatLab, sendo assim, no processo de simulação do modelo aproximado carrega-se o arquivo e aplica-se o processo discutido na seção 5.2.

O código desse programa encontra nas listagens apresentadas a seguir, assim como alguns detalhamentos mais expecífico de conversão do arquivo do Rosetta para ser usado diretamente no Matlab.

Partindo do arquivo final do Anexo A (*Export→Plain format*), teremos as regras do software Rosetta exportadas para um arquivo de texto.

Listagem A.1: Classe principal do programa para tratamento das regras

```
****Rules converter

package rulesprocessor;

import java.io.File;

import java.io.FileNotFoundException;

import java.util.Formatter;

import java.util.Scanner;
```

```
import java.util.logging.Level;
import java.util.logging.Logger;
public class RulesConverter {
  //define-se o número de antecedentes das regras
    private int numberOfTerms = 2;
 //vetores de valores mínimos e máximo de cada variável
    private double[ ] min;
    private double[ ] max;
 //quarto vetores auxiliaries para o processamento
    private double regraTermo1[ ] = new double[32];
    private double regraTermo2[ ] = new double[32];
    private double regrasConsequente1[ ];
    private double regrasConsequente2[ ];
 //número de regras
    private int numberOfRules = 16;
    private String arquivo;
 //constructor da classe, seta o valor das variáveis definidas
    public RulesConverter(int nor, int nof, double min[ ], double max[ ], String arquivo)
{
        numberOfRules = nor;
        numberOfTerms = nof;
        this.min = min;
        this.max = max;
        regraTermo1 = new double[numberOfTerms * numberOfRules];
        regraTermo2 = new double[numberOfTerms * numberOfRules];
        regrasConsequente1 = new double[numberOfRules];
```

```
        regrasConsequente2 = new double[numberOfRules];
        this.arquivo = arquivo;
    }
  //método para gerar as regras
    public void generateRules() {
        try {
            int cont = 0;
            int cont1 = 0;
                    //lê o arquivo do rosetta
            Scanner s = new Scanner(new File(arquivo));
            int contgeral= 1;
                    //inicia leitura
            while (s.hasNext()) {
                            //lê a linha atual
                String linha = s.nextLine( );
                            //substitue caracteres não compatíveis para o processamento
                linha = linha.replace("[", "u");
                linha = linha.replace(">", "j");
                            //obtem cada valor da linha atual
                String[ ] linhasep = linha.split("u");
                            //obtem os consequentes das regras
                String last = linhasep[linhasep.length - 1];
                String[ ] lasts = last.split("= j");
                linhasep[linhasep.length - 1] = lasts[0];
                            //percorre todos os valores da linha
                System.out.println(linha + " +++++++");
```

```
                for (int i = 0; i < linhasep.length; i++) {
                        //verifica o format correto
                   //System.out.println(linhasep[i]);
                   if (!linhasep[i].contains("x1a")) {
                        //verifica se é início de intervalo
                      if (linhasep[i].split(",")[0].equals("*")) {
                         regraTermo1[cont] = min[i - 1];
                      } else {
                         regraTermo1[cont] = Double.parseDouble(linhasep[i].split(",")[0]);
                      }
                     //preenche vetor que representa os maiores valores para os intervalos
                      if (linhasep[i].split(",")[1].replace(")",
"%").split("%")[0].trim().equals("*")) {
                         regraTermo2[cont] = max[i - 1];
                      } else {
                         regraTermo2[cont] =
Double.parseDouble(linhasep[i].split(",")[1].replace(")", "%").split("%")[0].trim());
                      }
                     //imprime os termos na tela oara verificar se estão corretos
                      System.out.println("TERMOS: " + regraTermo1[cont] + " - " +
regraTermo2[cont]);
                      cont++;
                   }
                }
                String consequente[ ] = lasts[1].split("OR");
                double menor = 10000;
```

```
                    double maior = -10000;
                   //varre os consequentes disponíveis
                    for (int j = 0; j < consequente.length; j++) {
                   //substitue caracteres incompatíveis
                       String str = consequente[j].replace("(", "%").split("%")[1].trim();
                       double n = Double.parseDouble(str.substring(0, str.length() - 1));
                  //obtem menores e maiores valores dos consequentes
                       if (n < menor) {
                          menor = n;
                       }
                       if (n > maior) {
                          maior = n;
                       }
                    //preenche vetores de maior e menor valores para os intervalos dos consequentes
                       regrasConsequente1[cont1] = menor;
                       regrasConsequente2[cont1] = maior;
                    }
                  //imprime na tela o resultado do processamento de regra de interação atual
                    System.out.println(cont1 +" - " + regrasConsequente1[cont1] + " - " + regrasConsequente2[cont1]);
                    cont1++;
                  }
                  s.close( );
               } catch (FileNotFoundException ex) {
                  Logger.getLogger(Main.class.getName()).log(Level.SEVERE, null, ex);
               }
```

```
        }
            //gera arquivo com as regras já processadas
        public void generateFile() {
           java.util.Formatter output = null;
           try {
              output = new Formatter("entradaMDTidiD2y1amodelo.txt");
              for (int i = 0; i < regrasConsequente1.length; i++) {
                 for (int k = 0; k < numberOfTerms; k++) {
                    output.format("%s %s ", regraTermo1[i * numberOfTerms + k],
regraTermo2[i * numberOfTerms + k]);
                 }
                 output.format("%s %s", regrasConsequente1[i], regrasConsequente2[i]);
                 output.format("\n");
              }
              output.close( );
           } catch (Exception ex) {
              output.close( );
              ex.printStackTrace( );
           }
        }
           //exibe o arquivo com as regras processadas na tela
        public void showRules() {
           for (int i = 0; i < regrasConsequente1.length; i++) {
              for (int k = 0; k < numberOfTerms; k++) {
                 System.out.print("x" + (k + 1) + " > " + regraTermo1[i * numberOfTerms +
k] + " AND  x" + (k + 1) + " < " + regraTermo2[i * numberOfTerms + k] + " AND ");
```

```
                //output.format("%s %s ", regraTermo1[i * numberOfTerms + k], regraTermo2[i * numberOfTerms + k]);
            }
            System.out.println(" entao y esta entre " + regrasConsequente1[i] + " e " + regrasConsequente2[i] );
        }
    }
    public static void tratamentoInicial(String f) {
        try {
            Scanner s = new Scanner(new File(f));
            String aux = "";
            String r = "";
            int cont = 0;
            while (s.hasNext()) {
                aux = s.nextLine();
                if (aux.equals("") || aux.contains("%") || aux.contains("Supp") || aux.contains("Acc") || aux.contains("Cov") || aux.contains("Stab")) {
                } else {
                    r += aux + "\n";
                    cont++;
                }
            }
            System.out.println(cont);
            s.close();
            Formatter rf = new Formatter("rulesFilesMDTidiD2y1amodelo");
            rf.format("%s", r);
            rf.close();
```

```
            } catch (FileNotFoundException ex) {
                Logger.getLogger(RulesConverter.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
        public static void main(String[] args){

RulesConverter.tratamentoInicial("C:\\MATLAB\\R2008a\\work\\MIMO_DISCRETO\\MD2x2IdInput_Novo\\MDTidi_D2y1a_R1");
        }

    }
```

Listagem A.2: Classe principal do programa para tratamento das regras

****** Main Java**

```
package rulesprocessor;
public class Main {
    public static void main(String[] args) {
        String arquivo;
        //  carrega arquivo gerado pelo rosetta
        arquivo = "D:\\Arquivos de Programa\\PROGRAMAS RUBIANE\\NetBeans 6.8\\RulesProcessor\\rulesFilesMDTidiD2y1amodelo";
       //deve-se definir 2 vetores com os maiores e menores valores de variável
        double[ ] min = new double[5];
        min[0] = -1.3507;
        min[1] = -1.3507;
        min[2] = 0.0000;
        min[3] = 0.0000;
        min[4] = 0.0000;
        double[ ] max = new double[5];
```

```
        max[0] = 12.5381;
        max[1] = 12.5381;
        max[2] = 5.0000;
        max[3] = 5.0000;
        max[4] = 5.0000;
           //Esta é a classe responsável pelo processamento
        RulesConverter conversor = new RulesConverter(23,5, min, max, arquivo);
        conversor.generateRules( );
        conversor.showRules( );
        conversor.generateFile( );
    }
}
```

Listagem A.3: Classe principal do programa para tratamento das regras

****Matlab (tratativa)

- Na linha de comando do Matlab, colocar o caminho da pasta *Rules processor:* D:\Arquivos de Programa\PROGRAMAS RUBIANE\NetBeans 6.8\RulesProcessor
- *Import Data* – importo o arquivo *entradaMDTidiD2y1amodelo.txt – abrir*
- Vai abrir a tela *Import Wizard → next→* na coluna, *name,* estará o arquivo *entradaMDTidiD2y1amodelo,* clico em cima do nome do arquivo e com o botão direito do mouse – renomeio o arquivo para arquivo → *regras_MDTidiD2y1a*
- Na Linha de comando do Matlab digito: *regras_MDTidiD2y1a*
- Irá mostrar as regras na tela;
- No comando de linha escreve-se: *regrasRes_MDTidiD2y1a= regras_MDTidiD2y1a*
- Salvo esse arquivo *regrasRes_MDTidiD2y1a* na pasta que estará sendo usada para armazenar os resultados do sistema em questão. Este arquivo será utilizado no programa para o processamento dos modelos baseados em regras descritos no próximo Anexo C.

ANEXO C

PROGRAMA PARA PROCESSAMENTO DOS MODELOS BASEADOS EM REGRAS

A simulação dos modelos aproximados é realizada através de um programa desenvolvido em MatLab. Este programa recebe como entrada o arquivo gerado pelo processamento realizado pelo programa descrito na Apêndice B.

A saída desse programa consiste nos valores de simulação dos dados utilizados para geração dos modelos e de valores da previsão *n* passos à frente. Gráficos da simulação e da previsão também são gerados.

A seguir, tem-se a listagem do programa de simulação dos modelos aproximados, para o exemplo do sistema MIMO discreto mostrado no experimento 6.1 e para o sistema de nível mostrado no experimento 6.2 do capítulo 6.

- Programa do Experimento 6.1 - modelo discreto de um dado sistema não linear com duas entradas e duas saídas (Wang et al., 2009).

```
% Sistema MIMO discreto

clear all;

% Carregar regras

load C:\MATLAB\R2008a\work\MIMO_DISCRETO\MD2x2IdInput_Novo\regrasRes_MDTidiD2y1a;
load C:\MATLAB\R2008a\work\MIMO_DISCRETO\MD2x2IdInput_Novo\Dados_MIMO_DISC_TESTE_y1a_idinput;
load C:\MATLAB\R2008a\work\MIMO_DISCRETO\MD2x2IdInput_Novo\regrasRes_MDTidiD2y2b;
load C:\MATLAB\R2008a\work\MIMO_DISCRETO\MD2x2IdInput_Novo\Dados_MIMO_DISC_TESTE_y2b_idinput;


Np=1000; % Parâmetros da série.
vn=1:Np;


x1 = Dados_MIMO_DISC_TESTE_y1a_idinput(1:1000,1);
x2 = Dados_MIMO_DISC_TESTE_y1a_idinput(1:1000,2);
x3 = Dados_MIMO_DISC_TESTE_y1a_idinput(1:1000,3);
x4 = Dados_MIMO_DISC_TESTE_y1a_idinput(1:1000,4);
x5 = Dados_MIMO_DISC_TESTE_y1a_idinput(1:1000,5);
y1 = Dados_MIMO_DISC_TESTE_y1a_idinput(1:1000,6);

x6 = Dados_MIMO_DISC_TESTE_y2b_idinput(1:1000,1);
x7 = Dados_MIMO_DISC_TESTE_y2b_idinput(1:1000,2);
```

```
x8 = Dados_MIMO_DISC_TESTE_y2b_idinput(1:1000,3);
x9 = Dados_MIMO_DISC_TESTE_y2b_idinput(1:1000,4);
x10 = Dados_MIMO_DISC_TESTE_y2b_idinput(1:1000,5);
y2 = Dados_MIMO_DISC_TESTE_y2b_idinput(1:1000,6);

msea1=0;
msea2=0;% variável da somatória do erro quadrático
ti = cputime;

for n=1:Np

  %y1kp1

nregras=23; % número de regras

%Para ye1a
  for a=1:nregras
    for i = 1:nregras
      za1(i) = -10000;
    end

    for i = 1:nregras

       if(x1(n) >= regras_MDTidiD2y1a(i,1) & x1(n) <= regras_MDTidiD2y1a(i,2) & x2(n) >=
regras_MDTidiD2y1a(i,3) & x2(n) <= regras_MDTidiD2y1a(i,4) & x3(n) >=
regras_MDTidiD2y1a(i,5) & x3(n) <= regras_MDTidiD2y1a(i,6) & x4(n) >=
regras_MDTidiD2y1a(i,7) & x4(n) <= regras_MDTidiD2y1a(i,8) & x5(n) >=
regras_MDTidiD2y1a(i,9) & x5(n) <= regras_MDTidiD2y1a(i,10))
         u1=0.2*(regras_MDTidiD2y1a(i,12)-regras_MDTidiD2y1a(i,11))*((x1(n)-
regras_MDTidiD2y1a(i,1))/(regras_MDTidiD2y1a(i,2)-regras_MDTidiD2y1a(i,1))+(x2(n)-
regras_MDTidiD2y1a(i,3))/(regras_MDTidiD2y1a(i,4)-regras_MDTidiD2y1a(i,3))+(x3(n)-
regras_MDTidiD2y1a(i,5))/(regras_MDTidiD2y1a(i,6)-regras_MDTidiD2y1a(i,5))+(x4(n)-
regras_MDTidiD2y1a(i,7))/(regras_MDTidiD2y1a(i,8)-regras_MDTidiD2y1a(i,7))+(x5(n)-
regras_MDTidiD2y1a(i,9))/(regras_MDTidiD2y1a(i,10)-
regras_MDTidiD2y1a(i,9)))+regras_MDTidiD2y1a(i,11);
         if u1 == 0
            u1 = regras_MDTidiD2y1a(i,12);
         end
          za1(i) = u1;
      end

    end

    zaTotal1=0;na1=0;

    for i = 1:nregras

       if(za1(i) > -10000)
          zaTotal1 = zaTotal1 + za1(i);
```

```
        na1 = na1 + 1;
      end
  end

   if(na1 == 0)
      na1 = 1;
   end

   zaTotal1 = zaTotal1 / na1;
   ye1a1 = zaTotal1;
 end
  %Armazenamento das variaveis
 vn1(n)=n-1;
 vye1a1(n) = ye1a1;

 %y2kp1

nregras1=22;

%Para ye2b

for b=1:nregras1
 for i = 1:nregras1
    za(i) = -10000;
 end

  for i = 1:nregras1

      if(x6(n) >= regras_MDTidiD2y2b(i,1) & x6(n) <= regras_MDTidiD2y2b(i,2) & x7(n) >=
regras_MDTidiD2y2b(i,3) & x7(n) <= regras_MDTidiD2y2b(i,4) & x8(n) >=
regras_MDTidiD2y2b(i,5) & x8(n) <= regras_MDTidiD2y2b(i,6) & x9(n) >=
regras_MDTidiD2y2b(i,7) & x9(n) <= regras_MDTidiD2y2b(i,8) & x10(n) >=
regras_MDTidiD2y2b(i,9) & x10(n) <= regras_MDTidiD2y2b(i,10))
        u=0.2*(regras_MDTidiD2y2b(i,12)-regras_MDTidiD2y2b(i,11))*((x6(n)-
regras_MDTidiD2y2b(i,1))/(regras_MDTidiD2y2b(i,2)-regras_MDTidiD2y2b(i,1))+(x7(n)-
regras_MDTidiD2y2b(i,3))/(regras_MDTidiD2y2b(i,4)-regras_MDTidiD2y2b(i,3))+(x8(n)-
regras_MDTidiD2y2b(i,5))/(regras_MDTidiD2y2b(i,6)-regras_MDTidiD2y2b(i,5))+(x9(n)-
regras_MDTidiD2y2b(i,7))/(regras_MDTidiD2y2b(i,8)-regras_MDTidiD2y2b(i,7))+(x10(n)-
regras_MDTidiD2y2b(i,9))/(regras_MDTidiD2y2b(i,10)-
regras_MDTidiD2y2b(i,9)))+regras_MDTidiD2y2b(i,11);
        if u == 0
           u = regras_MDTidiD2y2b(i,12);
        end
         za(i) = u;
     end

  end

  zaTotal=0;na=0;
```

```
    for i = 1:nregras1

        if(za(i) > -10000)
            zaTotal = zaTotal + za(i);
            na = na + 1;
        end
    end

    if(na == 0)
        na = 1;
    end

    zaTotal = zaTotal / na;
   ye2b = zaTotal;
end
 vn(n)=n-1;
 vye2b(n) = ye2b;

 vx4(n)=x4(n);
 vx9(n)=x9(n);

 y1(n)= y1(n); %armazenando os valores reais de entrada
    ea1=(y1(n)- vye1a1(n))^2;
    msea1 = msea1+ea1;
    ean1(n)=ea1;

  y2(n)= y2(n); %armazenando os valores reais de entrada
    ea2=(y2(n)-vye2b(n))^2;
    msea2 = msea2+ea2;
    ean2(n)=ea2;


end
tf = cputime;

msea1=msea1/2 %valor da somatória do erro quadrático
(sqrt(msea1)/Np) %Média da somatória do erro quadrático


msea2=msea2/2 %valor da somatória do erro quadrático
(sqrt(msea2)/Np) % Média da somatória do erro quadrático
delta4=(tf - ti)

  % para usar esses subplots, só quando quero u1, u2, y1, y2, originais e
  % estimados no mesmo gráfico
  % subplot(311);
   %plot(vn1(1:300),vx4(1:300),'b-',vn1(1:300),vx9(1:300),'r-');grid;
   %xlabel('amostras');
   %ylabel('u1(k) e u2(k)');

   subplot(311);
   plot(vn1(1:300),vx4(1:300),'b-',vn1(1:300),vx9(1:300),'r-');grid;
   xlabel('amostras');
   ylabel('u1(k) e u2(k)');
```

```
subplot(312);
plot(vn1(1:300),y1(1:300),'b-');grid;
xlabel('amostras');
ylabel('y1 Original Teste');

subplot(313);
plot(vn1(1:300),y2(1:300),'r-');grid;
xlabel('amostras');
ylabel('y2 Original Teste');
figure;

subplot(211);
plot(vn1(1:300),y1(1:300),'b-',vn1(1:300),vye1a1(1:300),'r-');grid;
xlabel('amostras');
ylabel('y1o  e y1e Teste');

subplot(212);
plot(vn1(1:300),y2(1:300),'b-',vn1(1:300),vye2b(1:300),'r-');grid;
xlabel('amostras');
ylabel('y2o e y2e Teste');

figure;
```

- **Programa do experimento 6.2:** Modelagem prática de um sistema de nível real constituído por dois reservatórios acoplados, representando um modelo não linear contínuo.

```
% Sistema de Nível acoplado
clear all;

% Carregar regras
load C:\MATLAB\R2008a\work\Sistema_Nivel\SN_RS\regrasRes_D2y1aSN;

load C:\MATLAB\R2008a\work\Sistema_Nivel\SN_RS\DadosTeste_MIMO_y1;

 load C:\MATLAB\R2008a\work\Sistema_Nivel\SN_RS\regrasRes_D2y2bSN;

load C:\MATLAB\R2008a\work\Sistema_Nivel\SN_RS\DadosTeste_MIMO_y2;

 Np=1000; % Parâmetros da série.

vn=1:Np;

x1 = DadosTeste_MIMO_y1(1:1000,1);%vh1n
x2 = DadosTeste_MIMO_y1(1:1000,2); %vh2n
x3 = DadosTeste_MIMO_y1(1:1000,3);%vU1n
x4 = DadosTeste_MIMO_y1(1:1000,4);%vU2n
y1 = DadosTeste_MIMO_y1(1:1000,5);%vX1np1

x5 = DadosTeste_MIMO_y2(1:1000,1);%vh1n
x6 = DadosTeste_MIMO_y2(1:1000,2); %vh2n
x7 = DadosTeste_MIMO_y2(1:1000,3);%vU1n
x8 = DadosTeste_MIMO_y2(1:1000,4);%vU2n
y2 = DadosTeste_MIMO_y2(1:1000,5);%vX2np

msea1=0;
msea2=0;%variável da somatória do erro quadrático
```

```
ti = cputime;

for n=1:Np

  %y1kp1

nregras=16;

%Para ye1a
  for a=1:nregras
    for i = 1:nregras
      za1(i) = -10000;
    end


    for i = 1:nregras

       if(x1(n) >= regras_D2y1aSN(i,1) & x1(n) <= regras_D2y1aSN(i,2) & x2(n) >=
regras_D2y1aSN(i,3) & x2(n) <= regras_D2y1aSN(i,4) & x3(n) >= regras_D2y1aSN(i,5) & x3(n) <=
regras_D2y1aSN(i,6) & x4(n) >= regras_D2y1aSN(i,7) & x4(n) <= regras_D2y1aSN(i,8))
          u1=0.25*(regras_D2y1aSN(i,10)-regras_D2y1aSN(i,9))*((x1(n)-
regras_D2y1aSN(i,1))/(regras_D2y1aSN(i,2)-regras_D2y1aSN(i,1))+(x2(n)-
regras_D2y1aSN(i,3))/(regras_D2y1aSN(i,4)-regras_D2y1aSN(i,3))+(x3(n)-
regras_D2y1aSN(i,5))/(regras_D2y1aSN(i,6)-regras_D2y1aSN(i,5))+(x4(n)-
regras_D2y1aSN(i,7))/(regras_D2y1aSN(i,8)-regras_D2y1aSN(i,7)))+regras_D2y1aSN(i,9);
          if u1 == 0
             u1 = regras_D2y1aSN(i,10);
          end
           za1(i) = u1;
       end

    end

    zaTotal1=0;na1=0;

    for i = 1:nregras

       if(za1(i) > -10000)
          zaTotal1 = zaTotal1 + za1(i);
          na1 = na1 + 1;
       end
    end

    if(na1 == 0)
       na1 = 1;
    end

    zaTotal1 = zaTotal1 / na1;
    ye1a1 = zaTotal1;
  end
```

```
  %Armazenamento das variaveis
 vn1(n)=n-1;
 vye1a1(n) = ye1a1;

 %y2kp1

nregras1=16;

%Para ye2b

for b=1:nregras1
 for i = 1:nregras1
   za(i) = -10000;
 end

  for i = 1:nregras1

     if(x5(n) >= regras_D2y2bSN(i,1) & x5(n) <= regras_D2y2bSN(i,2) & x6(n) >=
regras_D2y2bSN(i,3) & x6(n) <= regras_D2y2bSN(i,4) & x7(n) >= regras_D2y2bSN(i,5) & x7(n) <=
regras_D2y2bSN(i,6) & x8(n) >= regras_D2y2bSN(i,7) & x8(n) <= regras_D2y2bSN(i,8))
       u=0.25*(regras_D2y2bSN(i,10)-regras_D2y2bSN(i,9))*((x5(n)-
regras_D2y2bSN(i,1))/(regras_D2y2bSN(i,2)-regras_D2y2bSN(i,1))+(x6(n)-
regras_D2y2bSN(i,3))/(regras_D2y2bSN(i,4)-regras_D2y2bSN(i,3))+(x7(n)-
regras_D2y2bSN(i,5))/(regras_D2y2bSN(i,6)-regras_D2y2bSN(i,5))+(x8(n)-
regras_D2y2bSN(i,7))/(regras_D2y2bSN(i,8)-regras_D2y2bSN(i,7)))+regras_D2y2bSN(i,9);
       if u == 0
          u = regras_D2y2bSN(i,10);
       end
        za(i) = u;
     end

  end

  zaTotal=0;na=0;

  for i = 1:nregras1

     if(za(i) > -10000)
        zaTotal = zaTotal + za(i);
        na = na + 1;
     end
  end

  if(na == 0)
     na = 1;
  end

  zaTotal = zaTotal / na;
 ye2b = zaTotal;
```

```
end
 vn(n)=n-1;
 vye2b(n) = ye2b;

 vx3(n)=x3(n);
 vx4(n)=x4(n);

 y1(n)= y1(n); % armazenando os valores reais de entrada
 ea1=(y1(n)- vye1a1(n))^2;
 msea1 = msea1+ea1;
 ean1(n)=ea1;

 y2(n)= y2(n); % armazenando os valores reais de entrada
 ea2=(y2(n)-vye2b(n))^2;
 msea2 = msea2+ea2;
 ean2(n)=ea2;

end
 msea1=msea1/2 % valor da somatória do erro quadrático
(sqrt(msea1)/Np) % Média da somatória do erro quadrático


msea2=msea2/2 % valor da somatória do erro quadrático
(sqrt(msea2)/Np) % Média da somatória do erro quadrático
tf = cputime;
delta1=(tf - ti)

subplot(311);
  plot(vn,vx3,'b-',vn,vx4,'r-');grid;
  xlabel('amostras');
  ylabel('u1(k) e u2(k)');

  subplot(312);
  plot(vn,y1,'b-');grid;
  xlabel('amostras');
  ylabel('h1 Teste Original');

  subplot(313);
  plot(vn,y2,'r-');grid;
  xlabel('amostras');
  ylabel('h2 Teste Original');

  figure

  subplot(211);
  plot(vn,y1,'b-',vn,vye1a1,'r-');grid;
  xlabel('amostras');
  ylabel('h1o  e h1e');

  subplot(212);
  plot(vn,y2,'b-',vn,vye2b,'r-');grid;
  xlabel('amostras');
  ylabel('h2o e h2e');
```

ANEXO D

PROGRAMA PARA PROCESSAMENTO DOS MODELOS FUZZY TAKAGI-SUGENO

Para o modelo *fuzzy Takagi-Sugeno*, usado em comparação com a modelagem baseada em regras, os parâmetros foram ajustados pelo *toolbox ANFIS* do MatLab e para a computação das regras resultantes foi desenvolvido um programa em MatLab, que está presente nesse anexo.

A seguir, tem-se a listagem do programa de simulação para o exemplo do sistema MIMO discreto mostrado no experimento 6.1 e para o sistema de nível mostrado no experimento 6.2 do capítulo 6.

- Programa do Experimento 6.1 - modelo discreto de um dado sistema não linear com duas entradas e duas saídas (Wang et al., 2009).

```
% Sistema MIMO discreto
% Entrada igual ao modelo usado para gerar RS pelo Rosetta
clear all;

% Carregar os dados

load C:\MATLAB\R2008a\work\MIMO_DISCRETO\MD2x2_Fuzzy\Dados_MIMO_DISC_TESTE_y1a_idinput;
load C:\MATLAB\R2008a\work\MIMO_DISCRETO\MD2x2_Fuzzy\Dados_MIMO_DISC_TESTE_y2b_idinput;
Np=1000; % Parâmetros da serie.
vn=1:Np;
x1 = Dados_MIMO_DISC_TESTE_y1a_idinput(1:1000,1);%y1k
x2 =Dados_MIMO_DISC_TESTE_y1a_idinput(1:1000,2); %y1k_1
x3 = Dados_MIMO_DISC_TESTE_y1a_idinput(1:1000,3);%u1k_1
x4 = Dados_MIMO_DISC_TESTE_y1a_idinput(1:1000,4);%u1k
x5 = Dados_MIMO_DISC_TESTE_y1a_idinput(1:1000,5);%u2k_1
y1 = Dados_MIMO_DISC_TESTE_y1a_idinput(1:1000,6);%y1k+1
x6 = Dados_MIMO_DISC_TESTE_y2b_idinput(1:1000,1);%y2k
x7 = Dados_MIMO_DISC_TESTE_y2b_idinput(1:1000,2); %y2k_1
x8 = Dados_MIMO_DISC_TESTE_y2b_idinput(1:1000,3);%u1k_1
x9 = Dados_MIMO_DISC_TESTE_y2b_idinput(1:1000,4);%u2k
x10 = Dados_MIMO_DISC_TESTE_y2b_idinput(1:1000,5);%u2k_
y2 = Dados_MIMO_DISC_TESTE_y2b_idinput(1:1000,6);%y2k+1
```

```
fismat1 = readfis('MFTS_MIMOnly1a_teste');
fismat2 = readfis('MFTS_MIMOnly2b_teste');
t=0; T=1;
msea1=0;
msea2=0;% variável da somatória do erro quadrático
ti = cputime;
for n=1:Np
Y1kp1 = evalfis([x1 x2 x3 x4 x5],fismat1);
Y2kp1 = evalfis([x6 x7 x8 x9 x10],fismat2);
vn(n)=n-1;
Y1k = Y1kp1; Y2k = Y2kp1;
vx4(n)=x4(n);% armazenando o valor de U1k
vx9(n)=x9(n);% armazenando o valor de U2k
vY1kp1(n)=Y1kp1(n);% armazenando os valores das saídas das eq. as diferenças estimada
vY2kp1(n)=Y2kp1(n);% armazenando os valores das saídas das eq. as diferenças estimada
Y1k = Y1kp1; Y2k =Y2kp1; % atualizando as informações
 t = t + T;
y1(n)= y1(n); % armazenando os valores reais de entrada
ea1=(y1(n)- vY1kp1(n))^2;
msea1 = msea1+ea1;
ean1(n)=ea1;
y2(n)= y2(n); % armazenando os valores reais de entrada
 ea2=(y2(n)-vY2kp1(n))^2;
 msea2 = msea2+ea2;
ean2(n)=ea2;
end
tf = cputime;
msea1=msea1/2 % valor da somatória do erro quadrático
(sqrt(msea1)/Np) % valor eficaz do erro quadrático
msea2=msea2/2 % valor da somatória do erro quadrático
(sqrt(msea2)/Np) %valor eficaz do erro quadrático
delta5=(tf - ti)
%subplot(311);
% plot(vn(1:300),vx4(1:300),'b-',vn(1:300),vx9(1:300),'r-');grid;
% xlabel('amostras');
% ylabel('u1(k) e u2(k)');
subplot(211);
plot(vn(1:300),y1(1:300),'b-',vn(1:300),vY1kp1(1:300),'r-.');grid;
xlabel('amostras');
```

```
ylabel('y1o  e y1e Teste');
subplot(212);
 plot(vn(1:300),y2(1:300),'b-',vn(1:300),vY2kp1(1:300),'r-.');grid;
xlabel('amostras');
ylabel('y2o e y2e Teste');
%plot(vn,vx4,'b.-',vn,vx9,'r.-');grid;
%figure
%plot(vn,y1,'b.-',vn,Y1kp1,'r.-');grid;
%figure
%plot(vn,y2,'b.-',vn,Y2kp1,'r.-');grid;
%figure
%plot(vn,Y1kp1,'b.-',vn,Y2kp1,'r.-');grid;
```

- **Programa do experimento 6.2:** Modelagem prática de um sistema de nível real constituído por dois reservatórios acoplados, representando um modelo não linear contínuo.

```
% Sistema de Nível Acoplado
% entrada igual ao modelo usado para gerar RS pelo Rosetta
clear all;

load C:\MATLAB\R2008a\work\Sistema_Nivel\SN_FuzzyTS\DadosTeste_MIMO_y1;
load C:\MATLAB\R2008a\work\Sistema_Nivel\SN_FuzzyTS\DadosTeste_MIMO_y2;
Np=1000; % Parâmetros da serie.
vn=1:Np;
x1 = DadosTeste_MIMO_y1(1:1000,1);
x2 = DadosTeste_MIMO_y1(1:1000,2);
x3 = DadosTeste_MIMO_y1(1:1000,3);
x4 = DadosTeste_MIMO_y1(1:1000,4);
y1 = DadosTeste_MIMO_y1(1:1000,5);
x5 = DadosTeste_MIMO_y2(1:1000,1);
x6 = DadosTeste_MIMO_y2(1:1000,2);
x7 = DadosTeste_MIMO_y2(1:1000,3);
x8 = DadosTeste_MIMO_y2(1:1000,4);
y2 = DadosTeste_MIMO_y2(1:1000,5);
fismat1 = readfis('MFTS_MIMOnlSNtestey1');
fismat2 = readfis('MFTS_MIMOnlSNtestey2');
t=0; T=1;
msea1=0;
msea2=0;%variável da somatória do erro quadrático
ti = cputime;
```

```
for n=1:Np
Y1kp1 = evalfis([x1 x2 x3 x4],fismat1);
Y2kp1 = evalfis([x5 x6 x7 x8],fismat2);
vn(n)=n-1;
vx3(n)=x3(n);
 vx4(n)=x4(n);
vY1kp1(n)=Y1kp1(n);
vY2kp1(n)=Y2kp1(n);
Y1k = Y1kp1; Y2k =Y2kp1;
t = t + T;
y1(n)= y1(n); % armazenando os valores reais de entrada
 ea1=(y1(n)- vY1kp1(n))^2;
msea1 = msea1+ea1;
 ean1(n)=ea1;
y2(n)= y2(n); % armazenando os valores reais de entrada
ea2=(y2(n)-vY2kp1(n))^2;
msea2 = msea2+ea2;
ean2(n)=ea2;
end

tf = cputime;
msea1=msea1/2 % valor da somatória do erro quadrático
(sqrt(msea1)/Np) % valor eficaz do erro quadrático
msea2=msea2/2 % valor da somatória do erro quadrático
(sqrt(msea2)/Np) % valor eficaz do erro quadrático
deltat1=(tf - ti)
%subplot(311);
%plot(vn,vx3,'b.-',vn,vx4,'r.-'); grid;;
%xlabel('amostras');
%ylabel('u1(k) e u2(k)');
subplot(211);
plot(vn,y1,'b-',vn,Y1kp1,'r-.');grid;
xlabel('amostras');
```

ANEXO E

Justificativa Teórica de Aproximações de Funções via Modelos Aproximados

Será mostrado, matematicamente, que modelos aproximados podem aproximar uniformemente qualquer função com certo grau de precisão. Como a modelagem de sistemas dinâmicos é realizada com funções específicas, modelos aproximados podem definir qualquer função de modelagem.

Visando a estimativa de um valor de f relacionado com um modelo aproximado, o qual é relativo a certo valor de x compreendido entre $x^{(i)}$, $i = k, m$ e uma aproximação inferior e superior, mostrado aqui como $y^{(i)}$, $i = k, m$, isso é suficiente empregar uma expressão de interpolação. Por exemplo, a interpolação expressa por (D.1).

$$f = y_j = y_{Interp_j}(x_n, x_n^{(i)}, y_n^{(i)})_{\substack{i=k,m \\ n=1,N}} = y_j{}^{(k)} + \frac{(y_j{}^{(m)} - y_j{}^{(k)})}{N} \sum_{n=1}^{N} \frac{(x_n - x_n^{(k)})}{(x_n^{(m)} - x_n^{(k)})} \qquad \text{(D.1)}$$

O seguinte teorema mostra que uma função f relacionada a um modelo aproximado pode aproximar uniformemente qualquer função com certo grau de precisão.

Teorema de Aproximação Uniforme: Para uma dada função real g em um intervalo finito *F* de R^n e arbitrário $\varepsilon > 0$, existe uma função f tal que $|f(x) - g(x)| \leq \varepsilon$.

O teorema de Stone-Weierstrass será usado para provar o teorema da aproximação uniforme.

Definição: Uma álgebra Z de funções contínuas de valores reais definida em *F* é um conjunto de funções que possui a seguintes propriedades: $f_1, f_2 \in Z$; *c* real, implica em $f_1 + f_2 \in Z$, $cf_1 \in Z, f_1 f_2 \in Z$.

Lema1: Seja Q uma família de funções reais e contínuas em *F* tal que $f_1, f_2 \in Q$ implica $min[f_1, f_2] \in Q$ *e* $max[f_1, f_2] \in Q$.

Lema2: Seja Z uma função algébrica de valor real continuo definido em *F*. Seja L nesse contexto. Então, os elementos de L são uniformemente aproximados por elementos de Z.

As provas desses lemas são detalhadas em [Davis, 1975], bem como a apresentação formal do teorema de Stone-Weierstrass, o qual diz o seguinte:

Seja Z uma função algébrica de valor real continuo definido em *F*. A fim de que uma função contínua e real de valor arbitrário f seja uniformemente aproximada em *F* por elementos de Z, é necessário e suficiente que para quaisquer dois pontos $P^{(k)}, P^{(m)} \in F$, e qualquer $\varepsilon > 0$, possamos encontrar um $g \in Z$, tal que $\left|f\left(P^{(i)}\right) - g(P^{(i)})\right| \leq \varepsilon, i = k, m.$

Prova: Se f é uniformemente aproximada em *F* por elementos de Z, então esta condição obviamente se mantém. Seja L(Z) uma função plena de Z e seja f continuo em *F*. Para quaisquer dois pontos $P^{(k)}, P^{(m)} \in F$ e qualquer $\varepsilon > 0$, podemos encontrar um $g \in Z$ (e pela mesma razão $L(Z)$), que permite aproximar f uniformemente em *F* por elementos de $L(Z)$. Por outro lado, os elementos de L(Z) podem, pelo Lema 2, ser uniformemente aproximados por elementos de Z. Combinando estas duas aproximações, podemos então aproximar f por elementos de Z.

Corolário: Seja $f(x_1, x_2, \dots, x_N)$ uma função real e continua em *F*. Ela pode ser aproximada uniformemente em *F* por polinômios em $x_1, x_2, \dots, x_N$.

Prova: Para uma álgebra Z em um conjunto polinomial em $x_1, x_2, \dots, x_N$. Seja $P^{(i)} = \left[x_1^{(i)}, x_2^{(i)}, \dots x_N^{(i)}\right], i = k, m,$ para pontos distintos. Considere (D.2) um polinômio em $x_1, x_2, \dots, x_N$ e $f(P^{(i)} = g\left(P^{(i)}\right), i = k, m$. As condições do teorema de Stone-Weierstrass são satisfeitas com $\varepsilon > 0$.

$$f(x_1, x_2, \dots, x_N) = f\left(P^{(k)}\right) + \frac{f\left(P^{(m)}\right) - f\left(P^{(k)}\right)}{\sum_{n=1}^{N}\left(x_n^{(m)} - x_n^{(k)}\right)} \sum_{n=1}^{N}\left(x_n - x_n^{(k)}\right)\left(x_n^{(m)} - x_n^{(k)}\right) \qquad \text{(D.2)}$$

Observações: (D.1) fornece os mesmos resultados numéricos de (D.2) que é expressão de Lagrange para interpolações lineares. (D.1) é usada para interpolar valores intermediários entre as aproximações inferiores e superiores de um determinado modelo aproximado, em outras palavras: $f = f(x_1, x_2, \dots, x_N) = f_{Interp}(x_1, x_2, \dots, x_N) = y_{Interp}(.)$. Assim, um modelo aproximado pode aproximar qualquer função real genérica.