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

Ambiente para Desenvolvimento de Aplicações *Fuzzy* Industriais

Bruno Sielly Jales Costa

Natal, RN, Maio de 2010.

Livros Grátis

http://www.livrosgratis.com.br

Milhares de livros grátis para download.

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

Ambiente para Desenvolvimento de Aplicações *Fuzzy* Industriais

Bruno Sielly Jales Costa

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

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

Natal, RN, Maio de 2010.

Divisão de Serviços Técnicos

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

Costa, Bruno Sielly Jales.
Ambiente para desenvolvimento de aplicações *fuzzy* industriais / Bruno Sielly Jales Costa - Natal, RN, 2010
71 f.

Orientador: Luiz Affonso Henderson Guedes de Oliveira.

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

1. Lógica *fuzzy* - Dissertação. 2. Automação industrial - Dissertação. 3. Controle de processos - Dissertação. 2. OPC - Dissertação. I. Oliveira, Luiz Affonso Henderson Guedes de. II. Universidade Federal do Rio Grande do Norte. III. Título.

RN/UF/BCZM CDU 004.312(043.3)

Ambiente para Desenvolvimento de Aplicações *Fuzzy* Industriais

Bruno Sielly Jales Costa

Dissertação de Mestrado aprovada em 22 de dezembro de 2009 pela banca examinadora composta pelos seguintes membros:

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

Prof. Dr. Anderson Luiz de Oliveira Cavalcanti . . . . . DCA/UFFN

Prof. Dr. José Álvaro de Paiva . . . . . IFRN

Aos meus pais e namorada, pelo esforço e reconhecimento durante toda essa jornada.

Agradecimentos

Ao meu orientador, professor Luiz Affonso, pela paciência e disponibilidade na realização deste trabalho.

Aos professores Fábio Meneghetti e Anderson Luiz, pelas diversas sugestões no decorrer do trabalho.

Aos colegas Clauber, Daniel e Mário pelo auxílio constante prestado.

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

À minha família, pelo apoio durante esta jornada.

À minha namorada, pela paciência e apoio nos momentos mais difíceis.

Por último, mas não menos importante, a Deus pela força para enfrentar esse desafio.

Resumo

Este trabalho descreve o projeto, implementação e aplicação de um sistema para controle de processos industriais, baseado na lógica *fuzzy* e desenvolvido na linguagem Java, com suporte à comunicação industrial através do protocolo OPC (*Ole for Process Control*). Além do *framework* Java, o *software* é completamente independente de outras plataformas. Ele provê ferramentas amigáveis e funcionais para modelagem, construção e edição de sistemas de inferência *fuzzy* complexos, e utiliza tais sistemas lógicos no controle de uma grande variedade de processos industriais. Os principais requisitos do sistema desenvolvido devem ser flexibilidade, robustez, confiabilidade e facilidade de expansão.

Palavras-chave: Lógica *Fuzzy*, Automação Industrial, Controle de Processos, OPC.

Abstract

This work describes the design, implementation and application of a fuzzy logic toolbox for industrial process control, based on Java language, supporting communication through the OPC (Ole for Process Control) industrial protocol. Besides Java framework, the toolbox is completely independent of any other platforms. It provides easy and functional tools for modeling, building and editing complex fuzzy inference systems and using such logic systems to control a large variety of industrial processes. The main system requirements must be flexibility, reliability and ease of expansion.

Keywords: Fuzzy Logic, Industrial Automation, Process Control, OPC.

Sumário

Sumário i

Lista de Figuras iii

Lista de Tabelas v

1 Introdução 1
- 1.1 Objetivo . . . 1
- 1.2 Motivação . . . 2
- 1.3 Estrutura do Trabalho . . . 2

Lista de Símbolos e Abreviaturas 1

2 Automação Industrial 3
- 2.1 Nível de Controle . . . 5
  - 2.1.1 Controladores Lógicos Programáveis . . . 5
  - 2.1.2 Controle PID . . . 7
  - 2.1.3 Controle *Fuzzy* . . . 8
- 2.2 Nível de Supervisão . . . 8
  - 2.2.1 Sistemas Supervisórios . . . 8
  - 2.2.2 Interface Homem-Máquina . . . 9
  - 2.2.3 Controle Inteligente . . . 10
- 2.3 Protocolo de Comunicação OPC . . . 11
  - 2.3.1 OPC-DA . . . 12

3 Lógica *Fuzzy* 13
- 3.1 Introdução . . . 13
- 3.2 Conjuntos *Fuzzy* . . . 14
  - 3.2.1 Operações Sobre Conjuntos *Fuzzy* . . . 15
- 3.3 Variáveis Linguísticas . . . 19
- 3.4 Lógica *Fuzzy* . . . 20
  - 3.4.1 Regras Se-Então . . . 21
  - 3.4.2 Operadores de Implicação . . . 21
- 3.5 Funções de Pertinência . . . 22
- 3.6 Sistemas de Inferência *Fuzzy* . . . 24
  - 3.6.1 Modelo *Fuzzy* de Mamdani . . . 26

3.6.2 Modelo *Fuzzy* de Takagi-Sugeno 27
3.6.3 Comparação entre os modelos de Mamdani e de Takagi-Sugeno . 29

4 Arquitetura do JFuzZ **31**
4.1 Arquitetura do Sistema 31
4.1.1 Módulo de Edição *Fuzzy* / Configurador 31
4.1.2 Módulo de *Engine Fuzzy* 31
4.1.3 Módulo de Comunicação 33
4.1.4 Módulo Configurador de Tags 33
4.1.5 Módulo de Persistência 33
4.2 Modelagem do Sistema 35
4.3 Funcionalidades do Sistema 36
4.3.1 Editor básico de sistemas de inferência *fuzzy* 36
4.3.2 Editor de funções de pertinência 38
4.3.3 Editor de regras de inferência 39
4.3.4 Visualizador de regras 39
4.3.5 Módulo de Controle 41

5 Validação e Resultados Obtidos **43**
5.1 Validação Estática 43
5.1.1 Descrição do Problema 43
5.1.2 Modelagem do Sistema 43
5.1.3 Procedimentos Realizados e Resultados Obtidos 46
5.2 Tarefa de Controle Real 47
5.2.1 Descrição do Problema 49
5.2.2 Arquitetura da Aplicação 50
5.2.3 Configurações de Controle no CLP 51
5.2.4 Configuração do Servidor OPC 53
5.2.5 Implementação das Estratégias Controle 54
5.2.6 Resultados Obtidos 65

6 Considerações Finais **67**

Referências Bibliográficas **69**

Lista de Figuras

2.1 Pirâmide da automação industrial. 5
2.2 Arquitetura de um CLP. 6
2.3 Diagrama de blocos de um sistema de controle PID. 7
2.4 Diagrama de blocos de um sistema de controle *fuzzy* direto. 8
2.5 Rede de supervisão. 9
2.6 Exemplo de estrutura de controle adaptativo com sintonização do controlador PID através de lógica *fuzzy*. 10
2.7 Aplicações trabalhando com vários servidores OPC. 11

3.1 Função característica de A. 15
3.2 Função de pertinência de A. 15
3.3 União de dois conjuntos *fuzzy* $F = A \cup B$. 17
3.4 Interseção de dois conjuntos *fuzzy* $F = A \cap B$. 17
3.5 Complemento $\overline{F}$ de um conjunto F. 18
3.6 Ilustração de algumas das principais t-normas (azul). 19
3.7 Ilustração de algumas das principais s-normas (vermelho). 19
3.8 Valores *fuzzy* para a variável velocidade. 20
3.9 Ilustração das implicações $\mathbf{I}(x, \mu B(y))$ (em azul, as pertinências das consequências). 22
3.10 Função Triangular, modelada pelos parâmetros t_1, t_2 e t_3. 23
3.11 Função Trapezoidal, modelada pelos parâmetros r_1, r_2, r_3 e r_4. 23
3.12 Função Gaussiana, modelada pelos parâmetros g_1, g_2, e g_3. 24
3.13 Sistema de Inferência *Fuzzy*. 25
3.14 Principais métodos de defuzzificação. 26
3.15 Modelo clássico de Mamdani. Método de mínimo para união e método de máximo para agregação. 27
3.16 Modelo de interpolação de Takagi-Sugeno. Método de mínimo para união. 28

4.1 Arquitetura do sistema JFuzZ. 32
4.2 Métodos implementados no sistema. 32
4.3 Estrutura de dados do JFuzZ. 34
4.4 Diagrama de classes do JFuzZ. 35
4.5 Ambiente de edição de sistemas de inferência. 37
4.6 Ambiente de edição funções de pertinência. 38
4.7 Ambiente de edição regras de inferência. 39
4.8 Ambiente de visualização regras de inferência. 40
4.9 Ambiente de visualização do controlador *fuzzy*. 42

5.1 Funções de pertinência da variável *Distância*. 45
5.2 Funções de pertinência da variável *Velocidade*. 45
5.3 Funções de pertinência da variável *Força do freio (%)*. 45
5.4 Saídas para o sistema de inferência do tipo Mamdani. 48
5.5 Saídas para o sistema de inferência do tipo Takagi-Sugeno. 48
5.6 Erro para o sistema de inferência do tipo Mamdani. 49
5.7 Erro para o sistema de inferência do tipo Takagi-Sugeno. 49
5.8 Arquitetura da aplicação de controle. 50
5.9 Janela de configuração do SPDSW, utilizada para habilitação das entradas e saídas analógicas. 51
5.10 Programa em Ladder para controle PID do módulo de tanques. 52
5.11 Janela de configuração do controlador PID no CLP ZAP 900. 53
5.12 Janela de configuração do servidor OPC. 54
5.13 Janela de configuração dos blocos de dados no servidor OPC. 55
5.14 Nível do tanque para GP = 10. 56
5.15 Nível do tanque para GP = 2. 56
5.16 Sinal de controle para GP = 10. 57
5.17 Sinal de controle para GP = 2. 58
5.18 Módulo do erro para GP = 10. 58
5.19 Módulo do erro para GP = 2. 59
5.20 Funções de pertinência da variável *Erro*. 59
5.21 Funções de pertinência da variável *Variação do Erro*. 60
5.22 Funções de pertinência da variável *Tensão*. 60
5.23 Nível do tanque para a estratégia de controle *fuzzy* direto. 61
5.24 Tensão aplicada ao módulo de potência no controle *fuzzy* direto. 62
5.25 Módulo do erro no controle *fuzzy* direto. 62
5.26 Funções de pertinência para a variável *Erro* do controlador adaptativo. . . 63
5.27 Funções de pertinência para a variável *GP* do controlador adaptativo. . . . 63
5.28 Nível do tanque para a estratégia de controle PID adaptativo. 64
5.29 Tensão aplicada ao módulo de potência no controle PID adaptativo. . . . 64
5.30 Módulo do erro no controle PID adaptativo. 65

Lista de Tabelas

3.1 Definição de t-normas e s-normas. 19
3.2 Operadores duais de implicação. 22

5.1 Características das estratégias utilizadas no controle do módulo de tanques 65

Capítulo 1

Introdução

Desde a década de 1970, com a primeira aplicação em meio industrial implementada por Ebrahim Mamdani [MAMDANI & ASSILIAN 1975], a lógica *fuzzy* vem sendo usada no controle de processos dos mais diversos tipos.

Uma das principais potencialidades da lógica *fuzzy*, quando comparada a outras técnicas de manipulação de dados imprecisos, como as redes neurais, é que a sua base de conhecimento, formada por regras de inferência, é fácil de examinar e de compreender. O formato de regras também facilita a manutenção e atualização da base de conhecimentos.

Os controladores *fuzzy* vêm sendo cada vez mais utilizados em ambientes industriais, devido à sua capacidade de combinar a estrutura fácil da lógica com a utilidade dos controles realimentados no mundo real. Devido à similaridade da lógica *fuzzy* com o comportamento humano na solução de problemas complexos, o projetista ou operador pode utilizar o conhecimento empírico no projeto do controlador.

1.1 Objetivo

O objetivo deste trabalho é desenvolver e apresentar um ambiente para manipulação da teoria *fuzzy*, aplicando as suas bases e conceitos nas tarefas de controle industrial, através de um conjunto de ferramentas de rico conteúdo e alto nível de integração com os processos reais da indústria.

O ambiente de criação de aplicações *fuzzy* deve apresentar-se como uma ferramenta de *software* flexível, robusta, confiável e facilmente expansível. Tais requisitos são essenciais para qualquer *software* de ambiente industrial, dados os altos níveis de criticidade da maioria dos processos. Além dos requisitos gerais, a ferramenta desenvolvida deve ser capaz de comunicar-se em meio industrial, inicialmente através do protocolo OPC (*OLE for Process Control*).

Alguns trabalhos encontrados na literatura apresentam ambientes de *software* para criação de sistemas de inferência *fuzzy*. O *Matlab Fuzzy Logic Toolbox* é descrito como uma ferramenta de interface gráfica amigável para construção e edição de sistemas de inferência *fuzzy* [HALL & HATHAWAY 1996]. O *Scilab Fuzzy logic Toolbox* é apresentado como uma ferramenta direcionada à lógica *fuzzy* baseada na linguagem *Scilab*, mostrando-se como uma alternativa livre ao *software* da *MathWorks* [CHUAN et al. 2004].

1.2 Motivação

O *toolbox* do *MathWoks Matlab* para sistemas de inferência *fuzzy* apresenta uma boa abordagem para a construção de sistemas baseados na teoria dos conjuntos e regras *fuzzy*, através de uma poderosa interface gráfica, facilitando todo o processo de construção e edição. Apesar disso, esta alternativa ainda não é a mais adequada para ambientes industriais.

O *Fuzzy Logic Toolbox* do Matlab não possui intrinsecamente uma ferramenta de integração a processos e protocolos industriais, não sendo possível, desta forma, ler e escrever dados de forma direta na planta. Apesar da existência de ferramentas com este propósito no próprio ambiente do Matlab, este fator torna o processo mais difícil de implementar e gerenciar. Outro fator crucial é que o Matlab é uma ferramenta muito cara, especialmente para ambientes não-acadêmicos.

A escassez de um *software* simultaneamente eficiente na manipulação da teoria *fuzzy*, facilmente utilizável pelos projetistas e integrável ao controle de processos da indústria é a principal motivação deste trabalho, aliada à necessidade da utilização de técnicas avançadas aos modelos de controladores atualmente existentes, otimizando o controle da planta.

1.3 Estrutura do Trabalho

O presente trabalho é dividido em 5 capítulos. O capítulo 2 apresenta uma revisão sobre Automação Industrial, mostrando os principais conceitos, definições e tecnologias a respeito dos sistemas de controle. O capítulo 3 explana a teoria dos conjuntos *fuzzy*, bem como as técnicas de controle baseadas na lógica *fuzzy*. O capítulo 4 apresenta a arquitetura do sistema desenvolvido, detalhando os módulos, classes e telas do sistema. O capítulo 5 apresenta a validação e os resultados obtidos com o trabalho. Por fim, no capítulo 6 são apresentadas as considerações finais sobre o trabalho realizado e os trabalhos futuros.

Capítulo 2

Automação Industrial

Automação industrial trata do uso de sistemas de controle combinados com aplicações diversas de tecnologia da informação para controlar máquinas e processos industriais, reduzindo-se a necessidade de intervenção humana.

Em outras palavras, a automação estuda e aplica metodologias, ferramentas e equipamentos, objetivando definir quando e como converter o controle de um processo manual para automático. Ela também possibilita a coleta metodológica e precisa de dados que podem ser empregados para se obter a otimização do processo [ANDRADE 2001]. Assim, a automação pode reduzir de maneira significativa a necessidade de sensoriamento humano e requisitos mentais [SOMESSARI et al. 2009], e sua importância é crescente na economia global e na experiência diária.

Engenheiros esforçam-se para combinar dispositivos automatizados com ferramentas matemáticas e organizacionais, voltadas para a criação de sistemas complexos, contribuindo para uma rápida expansão do número de aplicações e atividades humanas realizadas por máquinas.

Porém, muitos dos papéis atribuídos a operadores humanos em meio industrial ainda encontram-se além do escopo da automação industrial. Tarefas que requerem avaliações subjetivas ou sínteses de dados complexos de sensoriamento, tais como classificação de sons e odores, bem como tarefas de alto nível como planejamento estratégico, ainda requerem decisões humanas. Além disso, em muitos casos, o uso de operadores humanos apresenta uma melhor relação custo benefício sobre abordagens mecânicas, mesmo onde a automação industrial pode ser aplicada.

Atualmente, existe grande interesse por sistemas de automação, especialmente onde a presença humana é insalubre ou perigosa, e ainda por sistemas de segurança em que os tempos de resposta têm de ser curtos. Mais especificamente, existe grande demanda de automação no setor industrial, devido às seguintes motivações [SANTOS et al. 2006]:

- Substituir operadores humanos em tarefas repetitivas.
- Substituir operadores humanos em tarefas realizadas em ambientes perigosos (fogo, espaço, instalações nucleares, sob a água etc.).
- Realizar tarefas que estão além das capacidades humanas, tais como o manuseio de cargas muito pesadas, de objetos de dimensões muito grandes, de substâncias muito quentes ou muito frias, ou tarefas que necessariamente precisam ser realizadas de forma muito rápida ou muito lenta.

- Avanço da economia: algumas vezes sistemas de automação implicam diretamente em avanço da economia das empresas, da sociedade ou até da humanidade. Por exemplo, quando uma empresa que investe em automação recupera o seu investimento, quando estados ou países aumentam as suas receitas através da automação, ou quando a humanidade pode usar a internet, que por sua vez utilizam satélites e outros mecanismos automatizados.

Em contrapartida, podemos citar algumas desvantagens da automação industrial:

- Tecnologia limitada: nos dias de hoje a tecnologia não é capaz de automatizar todas as tarefas desejadas;
- Custos de desenvolvimento elevados: os custos de pesquisa e desenvolvimento de processos de automação são difíceis de prever de forma precisa. Uma vez que esses custos podem ter um grande impacto na rentabilidade, é possível que se descubra que não há vantagem econômica em desenvolver um processo automatizado somente quando este já foi desenvolvido.
- Custos iniciais são relativamente altos: a automação de um novo produto requer um grande investimento inicial em comparação ao custo unitário do produto, embora este investimento seja o mesmo para grandes lotes do produto. A automação de uma planta também requer um grande investimento inicial, embora este custo seja distribuído nos produtos a serem produzidos.

Atualmente, o avanço da automação industrial está fortemente associado como as tecnologias de *software*. A utilização dessas tecnologias possibilitou aumentar a intensidade da automação industrial a níveis não imagináveis até pouco tempo atrás, de tal modo que o processo de automação não atinge apenas a produção em si, substituindo o trabalho manual por robôs e máquinas computadorizadas, mas permite enormes ganhos de produtividade ao integrar tarefas distintas como a elaboração de projetos, o gerenciamento administrativo e a produção [COSTA 2008]. Como o sistema de automação industrial se tornou bastante complexo, é comum analisá-lo de forma modular hierárquica, onde cada nível de abstração tem seus atributos, funcionalidades e requisitos específicos. Na figura 2.1 representa-se a chamada pirâmide de automação, com os diferentes níveis de automação encontrados em uma planta industrial. A pirâmide pode ser descrita em quatro níveis.

- **Sensores e atuadores:** Camada mais baixa da planta, onde encontram-se os sensores, atuadores, transmissores, módulos de conversão e outros dispositivos de campo. Essa camada foge do escopo deste trabalho.
- **Controle Direto:** Nível onde estão presentes os Controladores Lógicos Programáveis e alguns tipos de supervisão associadas ao processo. É detalhado nas subseções a seguir.
- **Supervisão:** Camada que permite o controle e monitoramento do processo produtivo da planta. É detalhada nas subseções a seguir.
- **Gerência:** Nível onde encontram-se os Sistemas de Execução de Manufatura (do inglês, *Manufacturing Execution System*) e o Planejamento de Recursos da Empresa

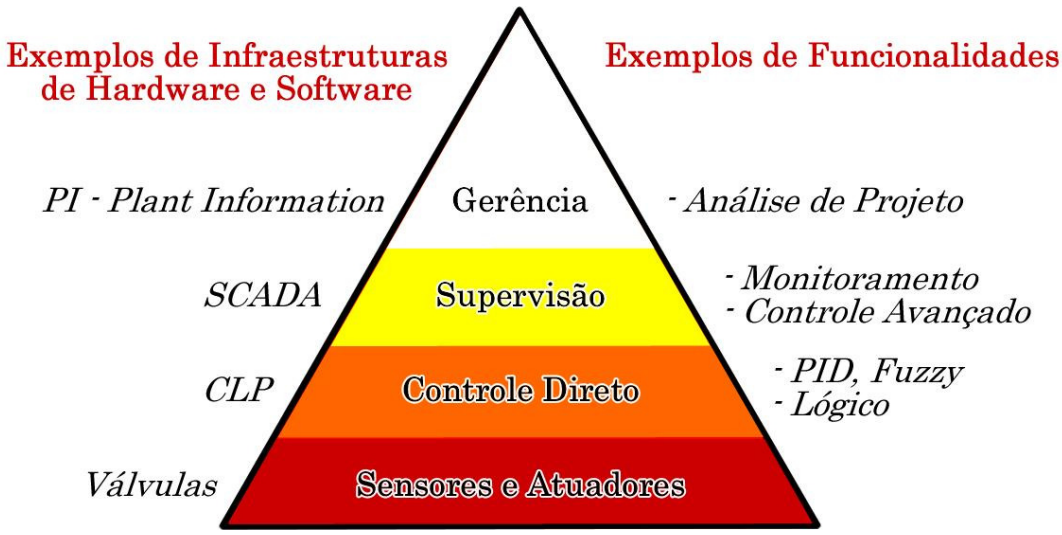

Figura 2.1: Pirâmide da automação industrial.

(do inglês, *Enterprise Resource Planning*). Camada responsável pela programação e planejamento da produção, bem como pela administração e gestão dos recursos da empresa. Esse nível foge do escopo deste trabalho.

2.1 Nível de Controle

Um sistema de controle é um dispositivo ou um conjunto de dispositivos que tem o objetivo de gerenciar, comandar, direcionar ou regular o comportamento de outros dispositivos ou sistemas.

Na indústria, encontramos basicamente duas classes de sistemas de controle, cada uma delas possuindo uma vasta gama de variações: controle lógico e controle regulatório.

O controle lógico é utilizado para sequenciar operações em diversas aplicações, e utilizam basicamente a lógica de relés como componente principal. Esta modalidade de controle foge do escopo deste trabalho.

Por sua vez, os sistemas de controle regulatório destinam-se fundamentalmente a manter uma variável de interesse (variável controlada) ajustada a um dado valor de referência (*setpoint*). No meio industrial, esse tipo de controle encontra vasto campo de aplicação na regulação das malhas de controle básicas do processo produtivo, constituindo um importante alicerce da produtividade industrial. De modo geral, a falta ou ineficiência de controle regulatório das malhas básicas de um processo industrial implica em alguma perda de produtividade. Em geral, este tipo de controle é implementado em dispositivos chamados de Controladores Lógicos Programáveis (CLPs).

2.1.1 Controladores Lógicos Programáveis

CLP é um dispositivo de controle baseado em um microprocessador programável através de linguagens específicas. Um controlador lógico programável moderno é nor-

malmente programado em qualquer linguagem de um conjunto de diversas linguagens de programação, tais como *Ladder*, *Basic* ou *C* [JONES 1996]. Tipicamente, o programa é escrito em um ambiente de desenvolvimento integrado a um computador de propósitos gerais, e depois é carregado para o CLP, diretamente através de um cabo de comunicação. O programa carregado é armazenado em uma memória não-volátil do CLP.

Controladores lógicos programáveis tipicamente contêm um número variável de portas de entrada e saída, e em sua grande maioria fazem parte dos computadores com conjunto de instruções reduzidas (do inglês, *Reduced Instruction Set Computer*, RISC). Eles são projetados para uso em tempo real, e devem resistir ao uso em ambientes severos a nível de chão de fábrica. Os circuitos dos controladores lógicos programáveis monitora o status dos múltiplos sensores em portas de entrada e, consequentemente, controlam os atuadores através das portas de saída, atuadores estes podendo ser *starters*, solenoides, lâmpadas, *displays* ou válvulas. A figura 2.2 ilustra a arquitetura de um CLP comum.

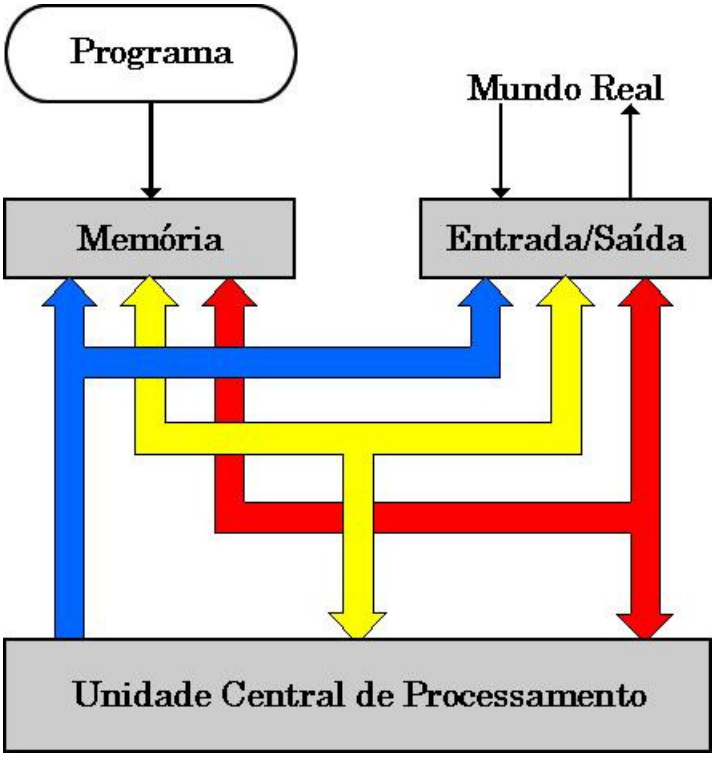

Figura 2.2: Arquitetura de um CLP.

Estes controladores têm dado uma contribuição significante para a automação industrial. Antigamente, os sistemas de automação eram compostos por milhares de relés e temporizadores individuais. Na maioria dos casos, a substituição de todos estes relés e temporizadores por um único controlador.

Hoje em dia, os controladores lógicos programáveis dispõem de um amplo conjunto de funcionalidades, incluindo o controle básico de relés, controle de movimentos, módulos para controle regulatório de processos e uso em redes complexas em sistemas de controle distribuídos. Dentre os módulos para controle regulatório, destacam-se os blocos

que implementam estratégias de controle do tipo PID, sendo que alguns CLPs mais atuais já disponibilizam blocos de controle baseados em lógica *fuzzy*.

2.1.2 Controle PID

Um controlador PID é um algoritmo de controle por realimentação mais utilizado em sistemas de controle industriais. Ele é responsável por corrigir o erro entre um valor medido e o valor que deseja-se obter, prontamente executando uma ação corretora no processo [YAP & DEPOI 2009].

O algoritmo de cálculo do controle PID é realizado através de três parâmetros distintos: o proporcional, o integral e o derivativo. O valor proporcional determina a reação do erro atual. O integral gera uma correção proporcional à integral do erro, assegurando que, aplicando-se um esforço de controle suficiente, o erro posterior tende a reduzir-se a zero. O parâmetro derivativo determina a reação do tempo na qual o erro se produz. A soma ponderada das três ações é usada para ajustar o processo através de um elemento de controle, como a posição de uma válvula ou a tensão aplicada a um motor, por exemplo. O diagrama de blocos de um sistema de controle PID é ilustrado na figura 2.3.

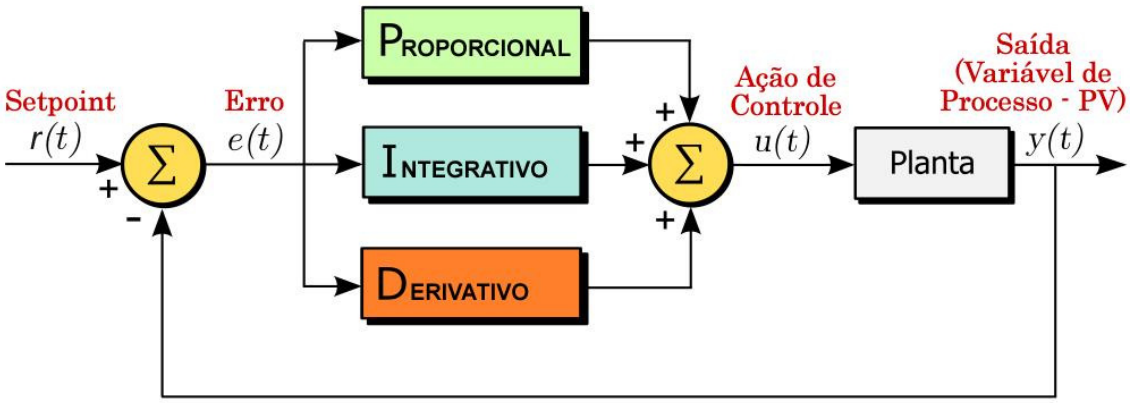

Figura 2.3: Diagrama de blocos de um sistema de controle PID.

A resposta do controlador pode ser descrita em termos da resposta ao erro, da velocidade de aproximação da variável medida ao *setpoint* e do grau de oscilação do sistema. Note que o uso de controladores PID não garante o controle ótimo ou a estabilidade do sistema.

Algumas aplicações podem exigir apenas um ou dois modos do controlador PID. Ele pode ser chamado também de controlador PI, PD, P ou I, na ausência das respectivas ações de controle. Os controladores PI são particularmente comuns, devido ao fato de a ação derivativa ser muito sensível a ruídos. Um controlador PI é utilizado no capítulo de resultados deste trabalho, para validação da aplicação desenvolvida.

2.1.3 Controle *Fuzzy*

Um sistema de controle *fuzzy* é um sistema que utiliza a lógica *fuzzy* para controlar processos diversos. A lógica *fuzzy*, na forma como é utilizada nos sistemas de controle, inclui imprecisões na descrição dos processos controlados [FLORESCU et al. 2001].

A lógica *fuzzy* é amplamente utilizada no controle dispositivos, partindo do pressuposto de que a lógica envolvida é capaz de trabalhar com conceitos nebulosos, e que determinadas variáveis do sistema não podem ser expressadas apenas como "verdadeiro"ou "falso", por exemplo, e necessitam da existência de estados intermediários.

O problema típico de controle direto consiste em obter, a partir do modelo da planta, uma lei de controle que atenda às especificações fornecidas previamente. O diagrama de blocos de um sistema de controle *fuzzy* direto é mostrado na figura 2.4.

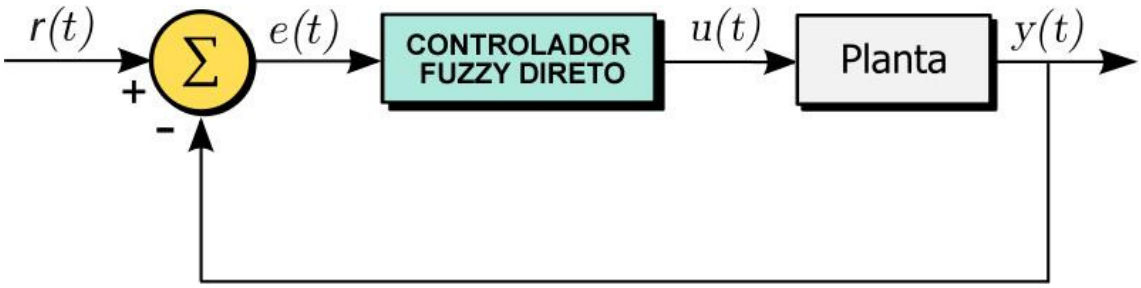

Figura 2.4: Diagrama de blocos de um sistema de controle *fuzzy* direto.

O controle direto é o esquema clássico onde o controlador calcula o valor da variável manipulada ($u(t)$) a partir do sinal de erro $e = r(t) - y(t)$, onde $r(t)$ é o sinal de referência (*setpoint*) e $y(t)$ é a variável controlada.

No capitulo 3 serão apresentados em detalhes os sistemas baseados em lógica *fuzzy*.

2.2 Nível de Supervisão

As características principais desse nível são as soluções para a supervisão associada ao processo, bem como a busca para solucionar as necessidades de controle avançado e inteligente. É o nível onde as informações necessárias para operar a camada de dispositivos de campo estão concentradas. Esse nível tipicamente envolve a aplicação de tecnologias como Sistemas Supervisórios (SCADA), Interfaces Homem-Máquina (IHM), controle avançado e otimização de processo.

2.2.1 Sistemas Supervisórios

Também são conhecidos por sistemas SCADA, sigla para *Supervisory Control And Data Acquisition*, os sistemas supervisórios são sistemas capazes de monitorar e controlar um processo através de um computador. Um sistema SCADA geralmente é composto por três elementos principais:

- Computador Central (MTU): é o coração do sistema SCADA, localizado na central de controle. Sua principal finalidade é adquirir informações e enviar comandos para o processo.
- Interface Homem-Máquina: aparato que apresenta dados do processo ao operador humano, e através disto, o operador humano monitora e controla o processo.
- Infra-estrutura de comunicação: conjunto de equipamentos e protocolos de comunicação que permitem a troca de informações entre os dispositivos do sistema.

A figura 2.5 ilustra a arquitetura genérica de um sistema SCADA.

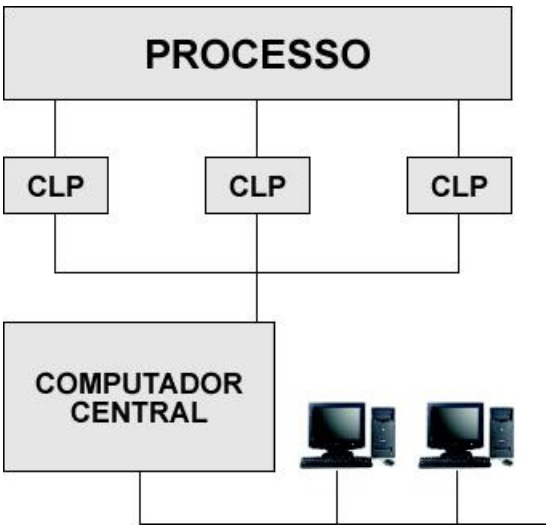

Figura 2.5: Rede de supervisão.

2.2.2 Interface Homem-Máquina

Também conhecida como interface de usuário, a Interface Homem Máquina é o elemento que permite que os usuários manipulem um determinado sistema, bem como permite que o sistema indique aos usuários os efeitos de tais manipulações.

Os sistemas automatizados necessitam de uma interface para facilitar a sua operação. É difícil avaliar ou analisar o que acontece no processo acompanhando diretamente no CLP ou outro dispositivo de controle. As IHMs permitem a monitoração, visualização e a supervisão do processo. Quando necessário também é possível a reconfiguração e controle da planta através da IHM.

Nos sistemas de automação industrial, as interfaces homem-máquina, em geral, são elementos de hardware e *software* responsáveis pela interação do ser humano com o processo. Exemplos comuns de IHMs são *software*s de supervisão e controle, *displays* e gráficos, botões, chaves, teclados etc.

2.2.3 Controle Inteligente

Com o avanço da tecnologia, os sistemas e processos ficaram cada vez mais robustos, tornando ineficaz ou até mesmo impossível a utilização dos controladores convencionais obtidos a partir da teoria clássica [LOPES et al. n.d.]. Isso desencadeou uma busca por novos métodos e estratégias de controle tais como controle multivariável, controle adaptativo, controle preditivo e sistemas de controle inteligente [FERNANDES et al. n.d.].

Em muitos casos reais, devido à complexidade da dinâmica do processo controlado, a estratégia de controle regulatório não é suficiente para atender aos requisitos operacionais do processo, como nos casos de mudança de *setpoint* de operação e presença de fortes distúrbios de carga.

Assim, quando o sistema de controle regulatório não apresenta um desempenho satisfatório, é necessária a utilização de um esquema de controle mais complexo e elaborado. Algumas das técnicas mais utilizadas em controle inteligente são o controle preditivo, as redes neurais e a lógica *fuzzy*.

Estratégias de controle inteligente podem apresentar-se em diversas estruturas de controle. Neste trabalho, destacamos a estrutura de controle adaptativo, onde o bloco de controle inteligente (inserido na camada de supervisão) sintoniza os parâmetros de um controlador regulatório do tipo PID através de inferências *fuzzy*, a partir dos valores do *setpoint* e da variável medida. O diagrama de blocos desse esquema é ilustrado na figura 2.6.

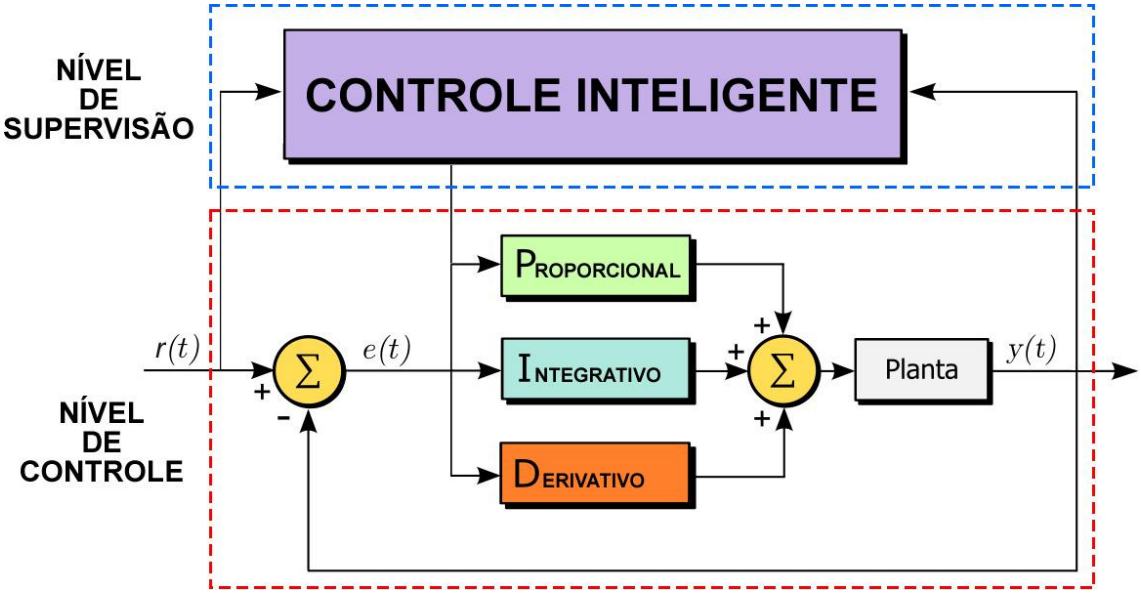

Figura 2.6: Exemplo de estrutura de controle adaptativo com sintonização do controlador PID através de lógica *fuzzy*.

Nessa estrutura de controle adaptativo, os parâmetros do controlador PID (em especial, os ganhos proporcional e integrativo) são configurados de acordo com o erro e a variação do erro atuais do sistema. O controlador PID é adaptado para apresentar menores *overshoot* e tempo de estabilização.

2.3 Protocolo de Comunicação OPC

OPC é a sigla para *Object Linking and Embedding (OLE) for Process Control*, e é o nome original para um conjunto de padrões e especificações para comunicação em tempo real entre dispositivos industriais de diferentes fabricantes [LIU et al. 2005]. Atualmente o OPC está se tornando um padrão de comunicação entre aplicações industriais.

A especificação do OPC é baseada nas tecnologias OLE, COM e DCOM da Microsoft, para a família de sistemas operacionais Microsoft Windows. Ela define um conjunto padrão de objetos, interfaces e métodos para uso no controle de processos e em aplicações de automação industrial, com o objetivo de facilitar a interoperabilidade. A especificação mais comum é a OPC-DA (OPC Data Access), que é usada para ler e escrever dados em tempo real.

O protocolo OPC foi projetado para ligar aplicações de *software* baseadas em Windows e hardware para controle de processos. O padrão define métodos consistentes para acesso a dados de campo através de dispositivos de chão de fábrica.

Na indústria, diversos fabricantes estão aptos a desenvolver os dispositivos de controle comuns. Tradicionalmente, para cada dispositivo, um driver ou uma interface customizada precisa ser escrita. O propósito do OPC é definir uma interface comum que é escrita uma vez, e reusada por qualquer aplicação.

Uma vez que um servidor OPC é desenvolvido para um dispositivo em particular, este pode ser usado por qualquer aplicação capaz de funcionar como um cliente OPC. Servidores OPC utilizam a tecnologia OLE (também conhecida como COM) para comunicar-se com os clientes. A tecnologia COM fornece um padrão para troca de informações em tempo real entre aplicações definidas. A figura 2.7 mostra um conjunto de aplicações trabalhando com vários servidores OPC.

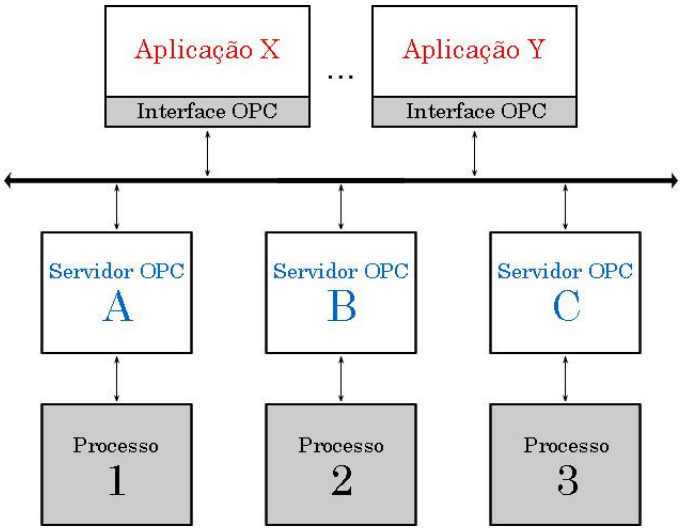

Figura 2.7: Aplicações trabalhando com vários servidores OPC.

2.3.1 OPC-DA

OPC Data Access é o nome dado a um grupo de padrões que fornecem especificações para comunicação de dados em tempo-real, a partir de dispositivos de aquisição de dados, tais como CLPs e IHMs. Tais especificações focam-se na comunicação contínua de dados.

O padrão OPC-DA trabalha exclusivamente com dados de tempo-real. Para dados históricos, utiliza-se o OPC-HDA (*OPC Historical Data Access*).

Existem basicamente três atributos associados à comunicação através do OPC-DA: o valor, a qualidade do valor (indica se o valor mostrado realmente corresponde ao real) e o *timestamp* (data e hora da leitura). A especificação do OPC-DA dita que esses três valores devem ser retornados a um cliente OPC após uma requisição.

Capítulo 3

Lógica *Fuzzy*

3.1 Introdução

Um ponto fundamental da lógica *booleana* é que ela impõe para qualquer objeto analisado, não importando o quão complexo este seja, uma de duas categorias pré-definidas, rígidas e complementares (alto/baixo, quente/frio, verdadeiro/falso etc).

Muitas vezes a lógica *booleana* se mostra como uma solução eficiente como, por exemplo, no projeto de circuitos lógicos, mas em alguns casos, não. Para ilustrar este problema, pensemos na estatura de um indivíduo. Podemos definir que uma pessoa alta é aquela que possui 1,80m ou mais. De acordo com a lógica *booleana*, uma pessoa que possui 1,79m de altura é considerada baixa, ou ainda, completamente baixa. A situação faz menos sentido ainda quando dizemos que uma pessoa de 1,79m é tão baixa quanto uma de 1,20m.

Em muitos casos, a precisão pode ser um tanto inútil, enquanto instruções não-precisas podem ser melhores interpretadas e, consequentemente, realizadas. Dois dos principais aspectos da imperfeição da informação são a imprecisão e a incerteza. As teorias mais conhecidas para tratar estes aspectos são a teoria dos conjuntos e a teoria das probabilidades, respectivamente, porém nem estas são completamente compatíveis com a riqueza das informações humanas.

Pensando em tratar do aspecto vago da informação, Lofti Zadeh desenvolveu, em meados da década de 1960 [ZADEH 1965], uma lógica que viola o conceito de que uma premissa precisa necessariamente ser completamente verdadeira ou completamente falsa. Zadeh criou uma teoria de conjuntos em que não há descontinuidades, e desta forma, não existe uma distinção abrupta entre os elementos pertencentes e não-pertencentes a um conjunto. Esses conjuntos são chamados de conjuntos nebulosos, conjuntos difusos ou simplesmente conjuntos *fuzzy*. A lógica *fuzzy* permite a representação de conceitos definidos por fronteiras difusas, como os que surgem na linguagem natural humana, ou conceitos qualitativos. Esta teoria, por ser menos restritiva, pode ser considerada mais adequada para o tratamento de informações fornecidas por seres humanos que a de probabilidades.

Zadeh notou que a modelagem de diversas atividades, sejam elas relacionadas a problemas biológicos, químicos, industriais etc., se tornariam demasiadamente complexas se implementadas de maneira convencional.

A lógica *fuzzy* tem sido cada vez mais usada em sistemas que utilizam informações fornecidas por seres humanos para automatizar procedimentos quaisquer, como, por exemplo, no controle de processos, no auxílio à decisão etc [CARULO 2008]. Eles foram utilizados, com sucesso, em algumas aplicações que se tornaram exemplos clássicos. A primeira aplicação que se tornou pública foi em 1974 na Inglaterra, onde o professor Mamdani implementou um controle de uma máquina a vapor baseado em lógica *fuzzy* [MAMDANI & ASSILIAN 1975].

Nos dias de hoje, estas teorias têm sido utilizadas em aplicações que vão do controle de eletrodomésticos ao controle de satélites, do mercado financeiro à medicina, e tendem a crescer cada vez mais.

3.2 Conjuntos *Fuzzy*

Um conjunto é uma coleção formada por vários objetos. Na teoria clássica de conjuntos, bem como na lógica de Boole, existem apenas duas possibilidades quanto à relação de um objeto qualquer a um conjunto: ou este objeto pertence ao conjunto, ou não pertence. A utilização da teoria clássica dos conjuntos na resolução de problemas propriamente humanos causa, em diversas situações, resultados incoerentes, fato este explicado pela limitação já conhecida sobre esta teoria.

A teoria dos conjuntos nebulosos ou conjuntos *fuzzy* permite que um objeto pertença a mais de um conjunto simultaneamente, sendo atribuídos a este objeto graus diferentes de pertinência para cada um dos conjuntos a que ele pertence.

Podemos formalizar a teoria dos conjuntos *fuzzy* estendendo a teoria clássica. Temos que um conjunto pode ser representado pela sua função indicadora ou função característica. Na teoria clássica de conjuntos, temos que um conjunto A, dentro do universo $\mathbf{U}$, pode ser definido por:

$$I_A : \mathbf{U} \to \{0,1\}, \text{ onde}$$
$$I_A(x) = \begin{cases} 1, & \text{se} \quad x \in A \\ 0, & \text{se} \quad x \notin A \end{cases}$$

Supondo que $\mathbf{U}$ seja o universo formado por todos os números reais não-negativos, e que A seja um subconjunto de $\mathbf{U}$, dentro de um intervalo fechado, a função característica é dada pelo gráfico que se segue na figura 3.1.

Caso o problema a ser resolvido seja, por exemplo, dentro do universo $\mathbf{U}$ definido em $\mathbb{R}$, encontrar quais números são primos e quais não são, o conceito de teoria clássica de conjuntos é suficiente, pois existe uma definição rigorosa de números primos. Entretanto, no caso em que $\mathbf{U}$ é definido em $\mathbb{R}$, e A é o conjunto dos números muito maiores que um valor x, o problema torna-se muito subjetivo. Esta subjetividade está ligada ao fato de que, dependendo do contexto, o critério que define quais números são muito maiores que x pode mudar.

Estendendo o conceito da teoria clássica, podemos formalizar a definição de um con-

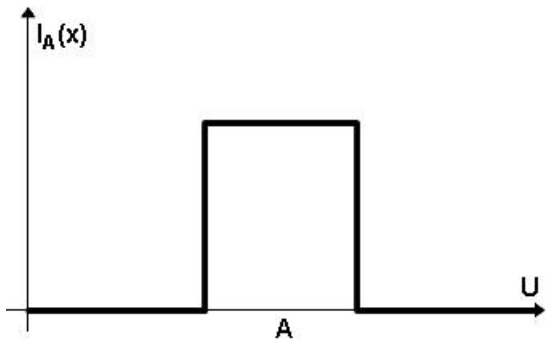

Figura 3.1: Função característica de A.

junto *fuzzy* A, dentro de um universo $\mathbf{U}$, descrevendo-o como

$$A = \{x \in \mathbf{U} \mid \mu_A(x) = \xi, 0 \leq \xi \leq 1\}$$

onde $\mu_A : \mathbf{U} \rightarrow [0,1]$ é a função que expressa o grau de pertinência de um objeto para um conjunto, ou seja, o quanto um objeto pertence a um determinado conjunto. A figura 3.2 ilustra um exemplo de função de pertinência para um conjunto *fuzzy* A.

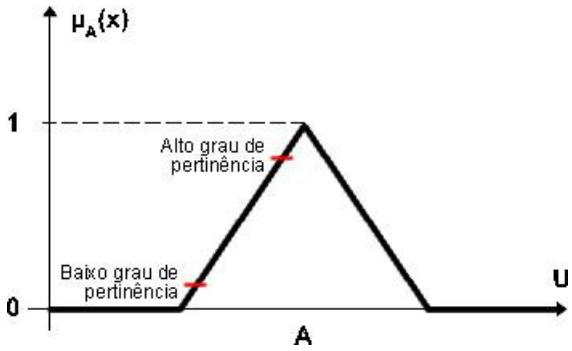

Figura 3.2: Função de pertinência de A.

Portanto, a teoria de conjuntos nebulosos ou conjuntos *fuzzy*, idealizada por Zadeh, busca formalizar as imprecisões encontradas nas definições dos problemas da natureza, quando estes são descritos em linguagens informais, propriamente humanas.

3.2.1 Operações Sobre Conjuntos *Fuzzy*

Na teoria clássica, dizemos que um elemento $x \in A \cup B$ quando este elemento está inserido no conjunto A ou no conjunto B. Caso o elemento x pertença simultaneamente aos conjuntos A e B, dizemos que $x \in A \cap B$. De forma geral, as operações básicas em teoria dos conjuntos são definidas como se segue:

$$A \cup B = \{x \in \mathbf{U} \mid x \in A \text{ ou } x \in B\}; \text{ união}$$

$$A \cap B = \{x \in \mathbf{U} \mid x \in A \text{ e } x \in B\}; \text{ interseção}$$

$$\overline{A} = \{x \in \mathbf{U} \mid x \notin A\}; \text{ complemento}$$

De forma análoga, na teoria dos conjuntos *fuzzy*, desejamos também fazer associações semânticas que resultem em interpretações úteis para expressões semelhantes. Desta maneira, temos que expandir as definições da teoria clássica, para que sejam possíveis operações com pertinência parcial para determinado conjunto.

Com base nesta concepção, Zadeh também definiu em seu trabalho algumas propriedades e operações com conjuntos *fuzzy*. Tais resultados têm o intuito de estender os conceitos da teoria clássica, incluindo-a como caso particular da teoria *fuzzy*.

Podemos então definir estas mesmas operações básicas, aplicando-as aos conjuntos *fuzzy*, simplesmente substituindo as funções características dos conjuntos clássicos pelas funções de pertinência dos conjuntos *fuzzy*. Assim, dados os conjuntos *fuzzy* A e B, dentro de um universo $\mathbf{U}$, associados às funções de pertinência μ_A e μ_B, definimos as seguintes operações básicas:

União

O conjunto *fuzzy* F é formado pela união dos conjuntos A e B ($F = A \cup B$), caso:

$$x \in \mathbf{U} \mid \mu_F(x) = max\{\mu_A(x),\ \mu_B(x)\}$$

Neste caso, o conjunto F representa o menor conjunto que contém os conjuntos A e B. Se um conjunto G contém F, então também contém a união de A e B. A figura 3.3 ilustra a operação de união entre dois conjuntos *fuzzy*, com $F = A \cup B$.

Interseção

O conjunto *fuzzy* F é formado pela interseção dos conjuntos A e B ($F = A \cap B$), caso:

$$x \in \mathbf{U} \mid \mu_F(x) = min\{\mu_A(x),\ \mu_B(x)\}$$

Logo, o conjunto F representa o maior conjunto *fuzzy* que está contido simultaneamente nos conjuntos A e B. De forma análoga à teoria clássica, os conjuntos A e B são disjuntos se $A \cap B = \emptyset$. A figura 3.4 ilustra a operação de interseção entre dois conjuntos *fuzzy*, $F = A \cap B$.

Complemento

O complemento de um conjunto *fuzzy* F, denotado por $\overline{F}$, é definido por:

$$x \in \mathbf{U} \mid \mu_{\overline{F}}(x) = 1 - \mu_F(x)$$

Neste caso, o conjunto $\overline{F}$ é formado pela coleção de todos os objetos que não pertencem

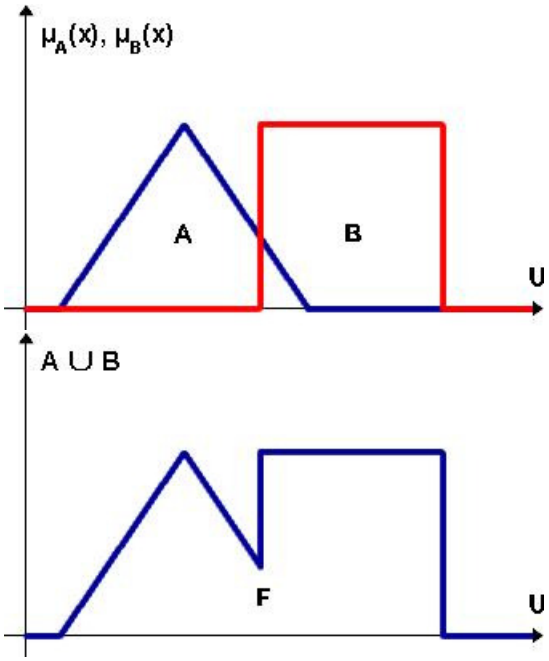

Figura 3.3: União de dois conjuntos *fuzzy* $F = A \cup B$.

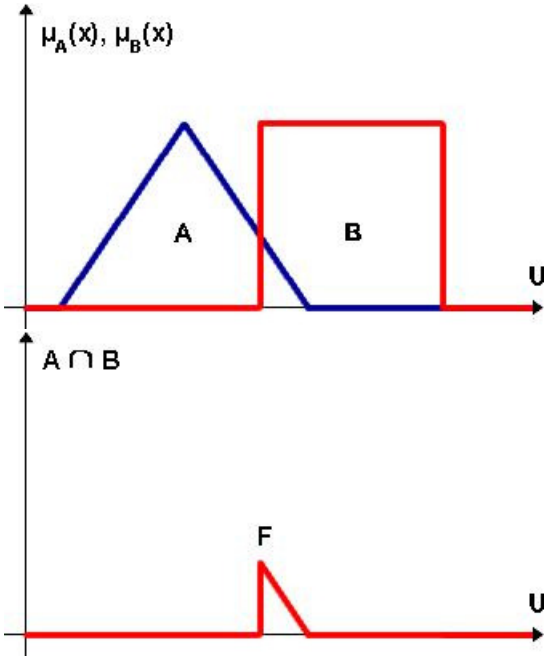

Figura 3.4: Interseção de dois conjuntos *fuzzy* $F = A \cap B$.

ao conjunto F. A figura 3.5 ilustra a operação de complemento $\overline{F}$ de um conjunto *fuzzy* F.

Devemos ressaltar que propriedades dos conjuntos clássicos como distributividade, associatividade, comutatividade, involução, leis transitiva e de De Morgan também valem

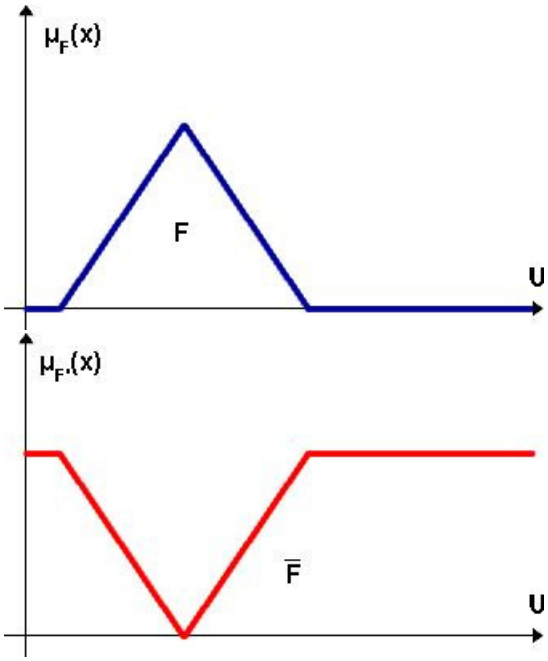

Figura 3.5: Complemento $\overline{F}$ de um conjunto F.

para as operações com conjuntos *fuzzy*.

Os operadores de união e interseção mostrados até o momento foram propostos inicialmente por Zadeh. Além destes, existe ainda uma ampla classe de modelos para conectores lógicos, denominados normas triangulares, e divididos em dois grupos distintos: *t-normas* e *s-normas* [PEDRYCZ 1993].

Uma t-norma é definida por uma função de dois argumentos

$$\mathbf{t} : [0,1] \text{ x } [0,1] \rightarrow [0,1]\,, \text{ tal que, para } x, y, w, z \in [0,1]\,, \text{ a função } \mathbf{t}$$

- é não decrescente em cada argumento: para $x \leq y, w \leq z,\ x\ \mathbf{t}\ w \leq y\ \mathbf{t}\ z$
- é comutativa: $x\ \mathbf{t}\ y = y\ \mathbf{t}\ x$
- é associativa: $(x\ \mathbf{t}\ y)\ \mathbf{t}\ z = x\ \mathbf{t}\ (y\ \mathbf{t}\ z)$
- satisfaz as condições de contorno: $x\ \mathbf{t}\ 0 = 0,\ x\ \mathbf{t}\ 1 = x$

Uma s-norma é definida por uma função de dois argumentos

$$\mathbf{s} : [0,1] \text{ x } [0,1] \rightarrow [0,1]\,, \text{ tal que, para } x, y, w, z \in [0,1]\,, \text{ a função } \mathbf{s}$$

- é não decrescente em cada argumento: para $x \leq y, w \leq z,\ x\ \mathbf{s}\ w \leq y\ \mathbf{s}\ z$
- é comutativa: $x\ \mathbf{s}\ y = y\ \mathbf{s}\ x$
- é associativa: $(x\ \mathbf{s}\ y)\ \mathbf{s}\ z = x\ \mathbf{s}\ (y\ \mathbf{s}\ z)$
- satisfaz as condições de contorno: $x\ \mathbf{s}\ 0 = x,\ x\ \mathbf{s}\ 1 = 1$

Podemos verificar que $min\{\mu_A(x),\ \mu_B(x)\}$ é uma t-norma e $max\{\mu_A(x),\ \mu_B(x)\}$ é uma s-norma, de modo que estas operações de união e interseção são instâncias particulares de

uma vasta gama de operações. Na tabela 3.1 são mostradas outras operações que satisfazem as propriedades das t-normas e s-normas [NASCIMENTO. & YONEYAMA 1999]; na figura 3.6 são ilustradas algumas das principais t-normas e na figura 3.7, algumas das principais s-normas.

Tabela 3.1: Definição de t-normas e s-normas.

Nome	**t-norma**	**s-norma**
Zadeh	$min\{x,y\}$	$max\{x,y\}$
Probabilístico	$x.y$	$x+y-xy$
Lukasiewicz	$max\{0,x+y-1\}$	$min\{1,x+y\}$
Yager, $w \in \{0,\infty\}$	$1-min\left\{[(1-x)^w+(1-y)^w]^{1/w},1\right\}$	$min\left\{[x^w+y^w]^{1/w},1\right\}$
Dubois e Prade, $\alpha \in \{0,1\}$	$\dfrac{x.y}{max\{x,y,\alpha\}}$	$\dfrac{x+y-xy-min\{x,y,1-\alpha\}}{max\{1-x,1-y,\alpha\}}$
Weber	$\begin{cases} x, & \text{se } y=1 \\ y, & \text{se } x=1 \\ 0, & \text{caso contrário} \end{cases}$	$\begin{cases} x, & \text{se } y=0 \\ y, & \text{se } x=0 \\ 1, & \text{caso contrário} \end{cases}$

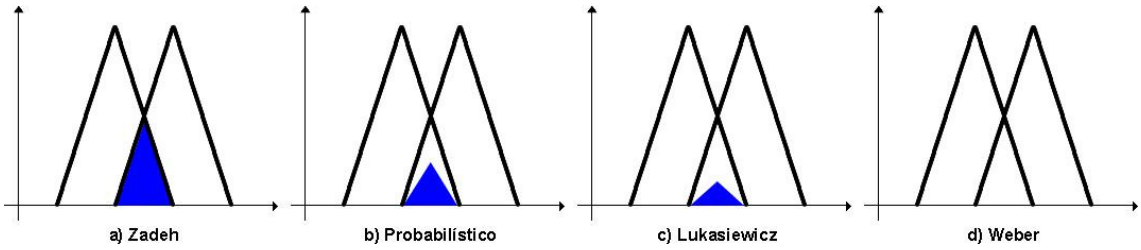

Figura 3.6: Ilustração de algumas das principais t-normas (azul).

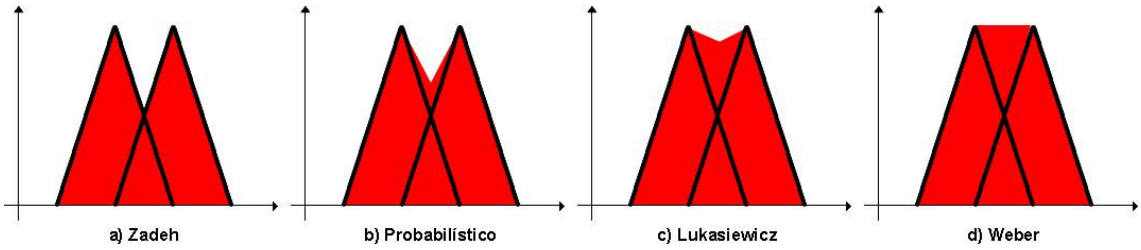

Figura 3.7: Ilustração de algumas das principais s-normas (vermelho).

3.3 Variáveis Linguísticas

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

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

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

O valor de uma variável linguística, ou valor *fuzzy*, é uma sentença composta por um termo primário (no exemplo anterior: *baixa, média, alta*), por conectores lógicos *(e, ou, não)* e por modificadores *(pouco, muito, extremamente)*. Podemos criar novos valores *fuzzy* adicionando conectores lógicos e modificadores aos valores *fuzzy* primários (por exemplo, podemos dizer que a velocidade do carro é *"não muito alta"*). A figura 3.8 ilustra a variável linguística velocidade com os valores *fuzzy* dados por *{Muito Baixa, Baixa, Média, Alta, Muito Alta}*.

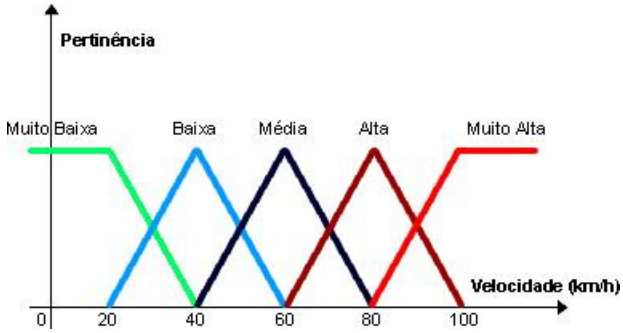

Figura 3.8: Valores *fuzzy* para a variável velocidade.

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

3.4 Lógica *Fuzzy*

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

3.4.1 Regras Se-Então

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

(Regra) Se x é A
(Consequência) Então y é B

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

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

Uma regra *fuzzy*, em diversos casos, pode possuir mais de um antecedente, como também mais de uma implicação. O exemplo a seguir ilustra esta afirmação:

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

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

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

3.4.2 Operadores de Implicação

Um operador de implicação é uma entidade usada para modelar regras de inferência do tipo Se-Então, e que pode ser definida por

$$\mathbf{I}\text{: } [0,1] \text{ x } [0,1] \rightarrow [0,1]$$

Considerando A e B dados por $\mu_A : X \rightarrow [0,1]$ *e* $\mu_B : Y \rightarrow [0,1]$, a relação $A \rightarrow B$ deve ser expressa por:

$$\mu_{A \rightarrow B}(x,y) = \mathbf{I}(\mu_A(x), \mu_B(y))$$

Apesar de as t-normas não serem operadores de implicação propriamente ditos, encontramos frequentemente aplicações em que estas são empregadas como implicação, notadamente em aplicações de controle *fuzzy*. A tabela 3.2 mostra alguns dos principais operadores duais de implicação encontrados na literatura [SANDRI & CORREA 1999], e a figura 3.9 ilustra o uso de alguns destes operadores.

Tabela 3.2: Operadores duais de implicação.

Nome	Operador de Implicação
Mamdani	$min\{x,y\}$
Larsen	$x.y$
Zadeh	$max\{1-x, min\{x,y\}\}$
Reichenbach	$1-x+xy$
Kleene	$max\{1-x,y\}$
Lukasiewicz	$min\{1-x+y,1\}$
Rescher - Gaines	$\begin{cases} 1, & \text{se } x \leq y \\ 0, & \text{caso contrário} \end{cases}$
Göedel	$\begin{cases} 1, & \text{se } x \leq y \\ y, & \text{caso contrário} \end{cases}$

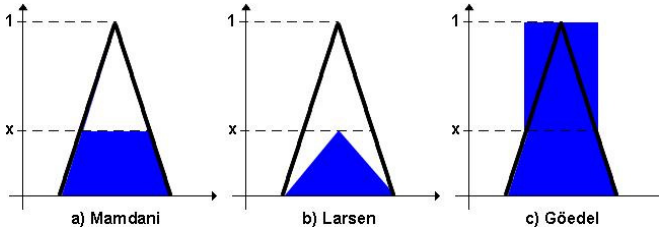

Figura 3.9: Ilustração das implicações $\mathbf{I}(x, \mu B(y))$ (em azul, as pertinências das consequências).

3.5 Funções de Pertinência

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

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

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

Função Triangular

Na função triangular, definem-se três parâmetros distintos: t_1, que representa o ponto a partir do qual a função assumirá valores maiores que 0; t_2, que representa o ponto no eixo x em que a função terá pertinência igual a 1; t_3, que representa o ponto a partir do qual a função assumirá sempre valores iguais a 0. A função triangular é representada na

figura 3.10.

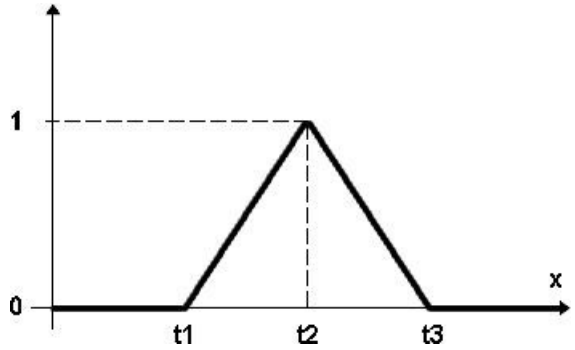

Figura 3.10: Função Triangular, modelada pelos parâmetros t_1, t_2 e t_3.

Matematicamente, definimos a função de pertinência triangular como se segue:

$$fF(x) = \begin{cases} \frac{x-t_1}{t_2-t_1}, & \text{se } t_1 \leq x \leq t_2 \\ \frac{t_3-x}{t_3-t_2}, & \text{se } t_2 \leq x \leq t_3 \\ 0, & \text{caso contrário} \end{cases}$$

Função Trapezoidal

Na função trapezoidal, existem quatro parâmetros para definição da função: r_1, que representa o ponto a partir do qual a função assumirá valores maiores que 0; r_2 e r_3, que representam o intervalo de constância da pertinência no valor 1; r_4, que representa o ponto a partir do qual a função assumirá sempre valores iguais a 0. A função trapezoidal é representada na figura 3.11.

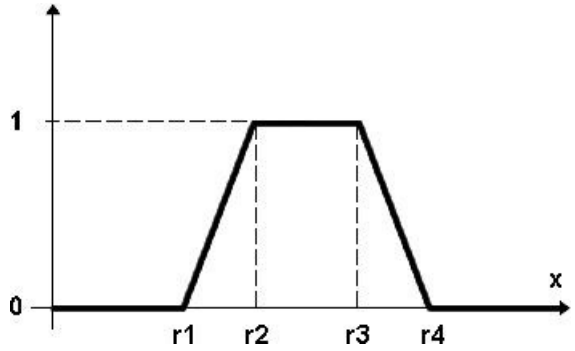

Figura 3.11: Função Trapezoidal, modelada pelos parâmetros r_1, r_2, r_3 e r_4.

Matematicamente, definimos a função de pertinência trapezoidal como se segue:

$$fF(x) = \begin{cases} \frac{x-r_1}{r_2-r_1}, & \text{se } r_1 \leq x \leq r_2 \\ 1, & \text{se } r_2 \leq x \leq r_3 \\ \frac{r_4-x}{r_4-r_3}, & \text{se } r_3 \leq x \leq r_4 \\ 0, & \text{caso contrário} \end{cases}$$

Função Gaussiana

Também conhecida como distribuição de Gauss. Nesta função, geralmente são utilizados 3 parâmetros para definição da função: g_1 e g_2, que representam os limites da função; g_3, que representa a variância da distribuição. Visualmente, a variância está diretamente relacionada à convexidade do sino formado pela função. A função gaussiana é representada na figura 3.12.

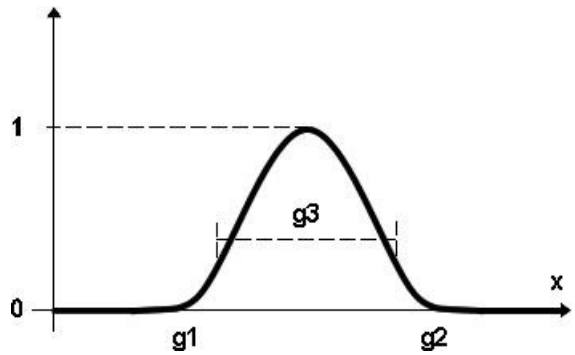

Figura 3.12: Função Gaussiana, modelada pelos parâmetros g_1, g_2, e g_3.

Matematicamente, e para fins de desenvolvimento do sistema, definimos a função de pertinência gaussiana como se segue:

$$fF(x) = \frac{5}{2\sqrt{2\pi}} \exp\left(\frac{(-x+0.5g_2+0.5g_1)^2}{2g_3}\right)$$

3.6 Sistemas de Inferência *Fuzzy*

Os sistemas de inferência *fuzzy* são ferramentas computacionais utilizadas nas mais diversas áreas da engenharia e da ciência, e agregam os conceitos de conjuntos *fuzzy*, variáveis lingüísticas e raciocínio aproximado, processando dados por meio de mecanismo de inferência. A estrutura básica de um sistema de inferência é mostrada na figura 3.13.

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

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

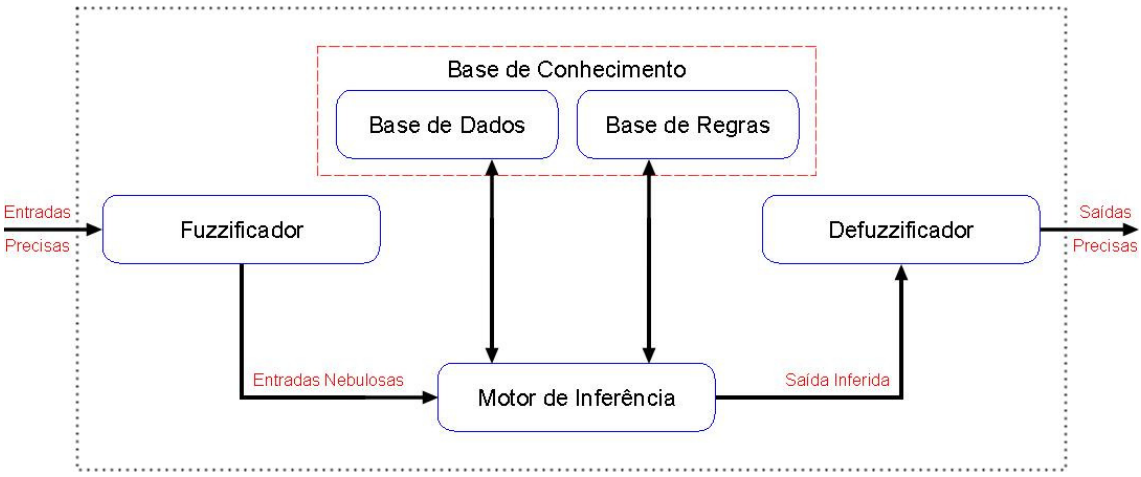

Figura 3.13: Sistema de Inferência *Fuzzy*.

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

Com base no grau de ativação determina-se a *consequência* produzida por uma determinada regra. Na maioria das vezes um sistema de inferência *fuzzy* possui mais de uma regra. Cada regra produz uma consequência e o resultado global da etapa inferência dependerá da combinação dessas conseqüências. Esta etapa é chamada de *agregação*, a qual tem por resultado um conjunto *fuzzy*, estando este diretamente relacionado com o operador de agregação aplicado. De forma similar ao uso de s-normas, a etapa de agregação pode ser realizada utilizando métodos de valor máximo, soma, métodos probabilísticos, dentre outros.

A próxima etapa é chamada de *defuzzificação*. O procedimento compreende a identificação do domínio das variáveis de saída num correspondente no universo de discurso, e com a saída nebulosa inferida evolui-se uma saída não nebulosa. Existem na literatura diversos métodos para defuzzificação, cada um apresentando melhores resultados para determinados tipos de problemas. Os métodos de defuzzificação mais utilizados são:

- **Método do Centróide:** O valor de saída é o centro de gravidade da função de distribuição de possibilidades de saída. É um dos médotos de defuzzificação mais utilizados [TANSCHEIT 1999] e geralmente apresenta bons resultados.
- **Método do Bissetor:** A saída é dada pelo ponto que divide a figura da função agregada em duas figuras de áreas exatamente iguais. Geralmente, o valor de saída deste método é bem próximo da saída do método do centróide.

- **Média dos Máximos (MOM):** Encontra o ponto médio entre os valores que têm o maior grau de pertinência inferido pelas regras.
- **Primeiro Máximo (SOM):** Encontra o valor de saída através do ponto em que o grau de pertinência da distribuição atinge o primeiro valor máximo.
- **Último Máximo (LOM):** O valor de saída deste método é o ponto em que o grau de pertinência da função de distribuição de possibilidades atinge o último valor máximo.

A figura 3.14 exemplifica a utilização de alguns dos principais métodos de defuzzificação.

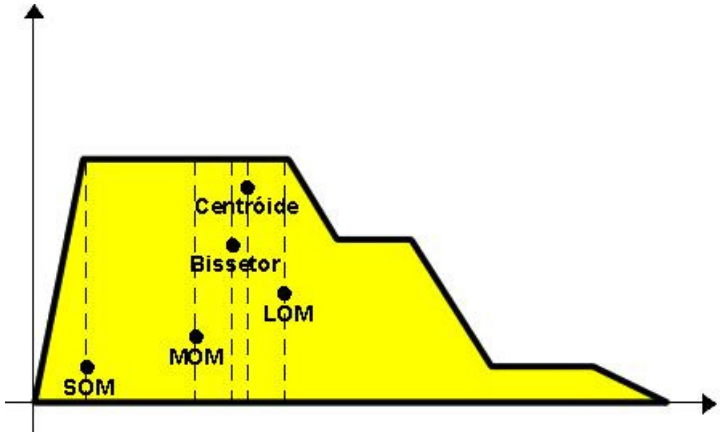

Figura 3.14: Principais métodos de defuzzificação.

3.6.1 Modelo *Fuzzy* de Mamdani

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

O trabalho de Mamdani foi baseado no artigo de Lotfi Zadeh, publicado em 1973 [ZADEH 1973], que tratava do estudo de algoritmos *fuzzy* para sistemas complexos e processos de decisão.

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

Um exemplo completo de inferência baseada no modelo de Mamdani é mostrado na figura 3.15, onde ilustram-se as etapas de fuzzificação, implicação, agregação e fuzzificação.

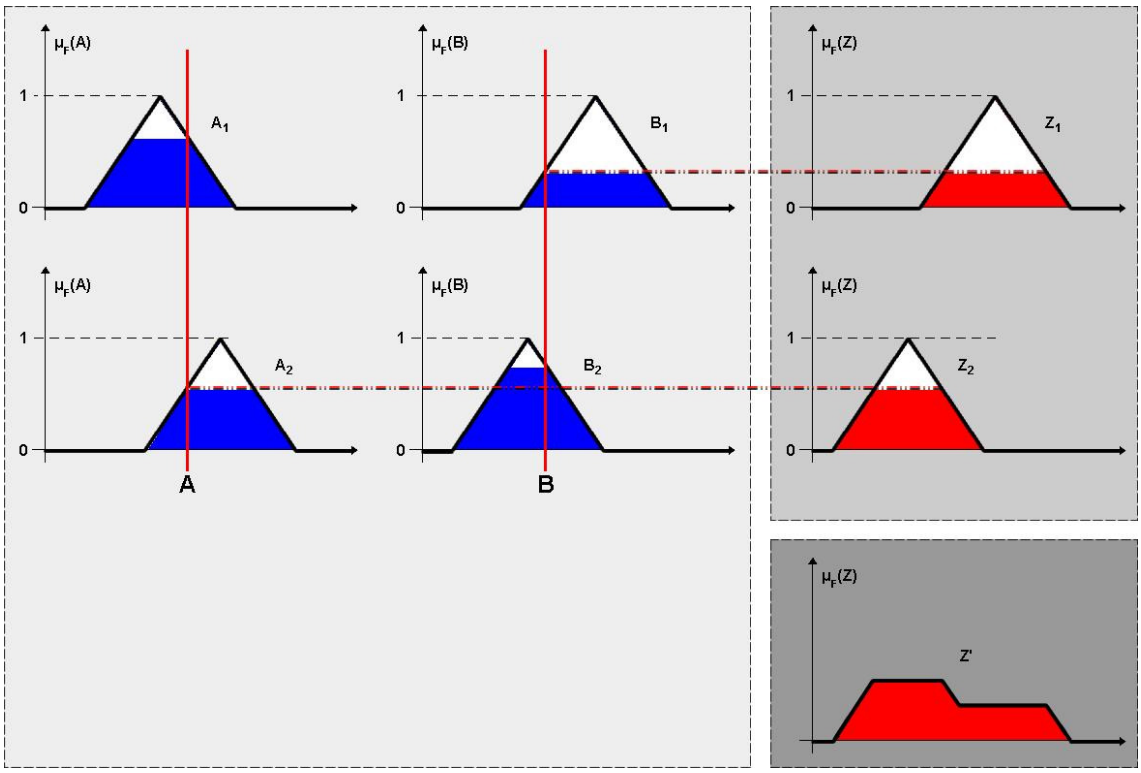

Figura 3.15: Modelo clássico de Mamdani. Método de mínimo para união e método de máximo para agregação.

3.6.2 Modelo *Fuzzy* de Takagi-Sugeno

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

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

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

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

$$\begin{gathered}\text{Se } x \text{ é } A \text{ e } y \text{ é } B \\ \text{Então } z = aA + bB + c\end{gathered}$$

Para um sistema de ordem zero, o nível de saída z é uma constante, dados $a = b = 0$.

Desta forma, uma função constante nada mais é que um peso atribuído a cada regra, de acordo com o grau de importância da mesma para a saída do sistema.

Diferentemente do modelo de Mamdani, um sistema Takagi-Sugeno não necessita de uma interface de defuzzificação, dado que a sua saída é calculada diretamente. O nível de saída z do sistema de inferência é calculado através do grau de ativação w_i de cada regra i, bem como pelo peso z_i atribuído a esta regra [SOUZA 2006]. A etapa de agregação é simplesmente dada pela média ponderada das regras. Assim, o nível de saída final do sistema é dado por:

$$z = \frac{\sum_{i=1}^{n} w_i z_i}{\sum_{i=1}^{n} w_i}$$

onde n é o número total de regras do sistema. Na figura 3.16, ilustra-se a inferência através do método de Takagi-Sugeno utilizando-se duas regras de inferência.

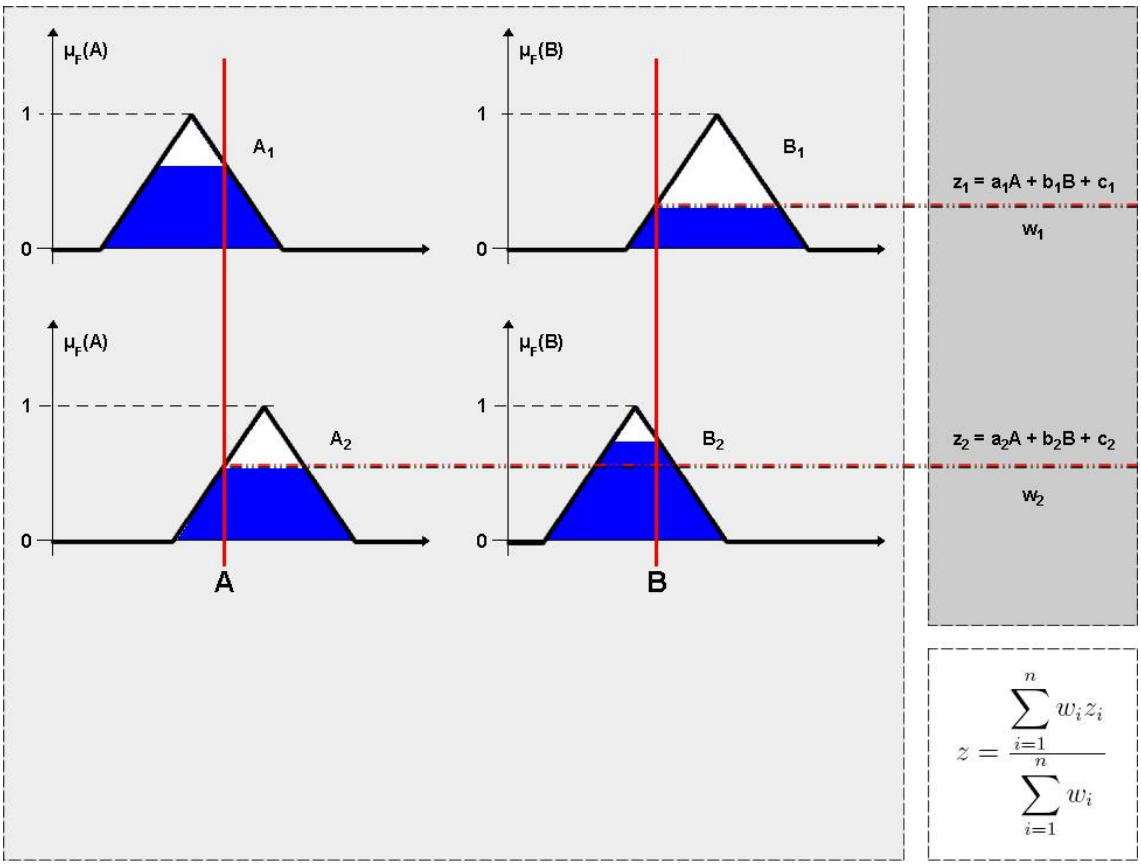

Figura 3.16: Modelo de interpolação de Takagi-Sugeno. Método de mínimo para união.

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

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

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

Vantagens do Modelo de Mamdani

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

Vantagens do Modelo de Takagi-Sugeno

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

No próximo capítulo é apresentada a arquitetura proposta neste trabalho para implementação de aplicações *fuzzy*. Essa arquitetura foi desenvolvida em linguagem Java e possui suporte de comunicação OPC-DA (*OLE for Process Control - Data Access*), que permite o seu emprego em aplicações industriais reais.

Capítulo 4

Arquitetura do JFuzZ

O sistema proposto neste trabalho, inicialmente chamado de JFuzZ, provê ferramentas para criação e edição de sistemas de inferência *fuzzy*. Uma vez criados e modelados, os sistemas de inferência podem ser utilizados no controle de processos industriais em geral, através do protocolo de comunicação OPC.

Devido à sua grande usabilidade, o *Fuzzy Logic Toolbox* (`www.mathworks.com`), ferramenta integrada ao *MathWorks Matlab*, serviu de modelo no projeto e desenvolvimento do JFuzZ. O sistema desenvolvido mantém características estruturais e visuais semelhantes à do *software* da *MathWorks*.

4.1 Arquitetura do Sistema

O JFuzZ, ambiente de criação e edição de sistemas *fuzzy* industriais, é uma ferramenta desenvolvida em linguagem *Java*. O sistema objetiva o uso em ambientes industriais reais, através do protocolo de comunicação OPC-DA.

De uma visão geral, o sistema JFuzZ está dividido em cinco módulos distintos e bem definidos, como ilustra a figura 4.1.

4.1.1 Módulo de Edição *Fuzzy* / Configurador

É o módulo responsável pela configuração do sistema de inferência *fuzzy*. Este módulo manipula informações básicas do sistema de inferência, tais como tipo, variáveis de entrada e saída, nomes, funções de pertinência, *ranges*, métodos de inferência, *tags* ligadas às variáveis etc.

Através do módulo de configuração, o sistema de inferência é modelado, sendo ele capaz de fornecer as informações necessárias à inferência realizada pelo motor *fuzzy*.

4.1.2 Módulo de *Engine Fuzzy*

Podemos dizer que este é o principal módulo do sistema, pois o *engine fuzzy* é a entidade responsável pela execução propriamente dita de todo o processo de inferência.

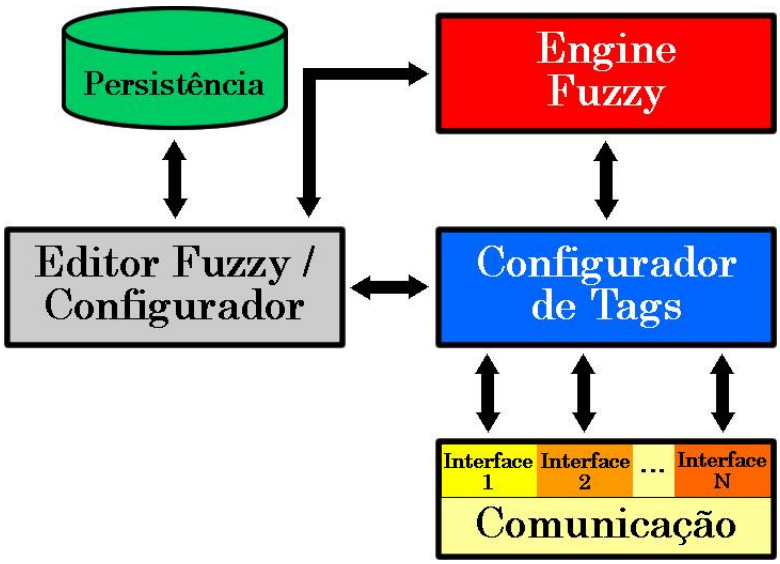

Figura 4.1: Arquitetura do sistema JFuzZ.

Esse módulo tem a função de coletar as informações do sistema de inferência a partir do módulo de configuração e, através dos parâmetros coletados, realizar as operações matemáticas sobre os valores de entrada, gerando um conjunto de saídas numéricas.

A *engine fuzzy* implementa os métodos mais comuns encontrados na literatura para sistemas de inferência *fuzzy*. A figura 4.2 mostra os métodos de t-norma, s-norma, implicação, agregação e defuzzificação para cada um dos tipos de sistema implementados no JFuzZ.

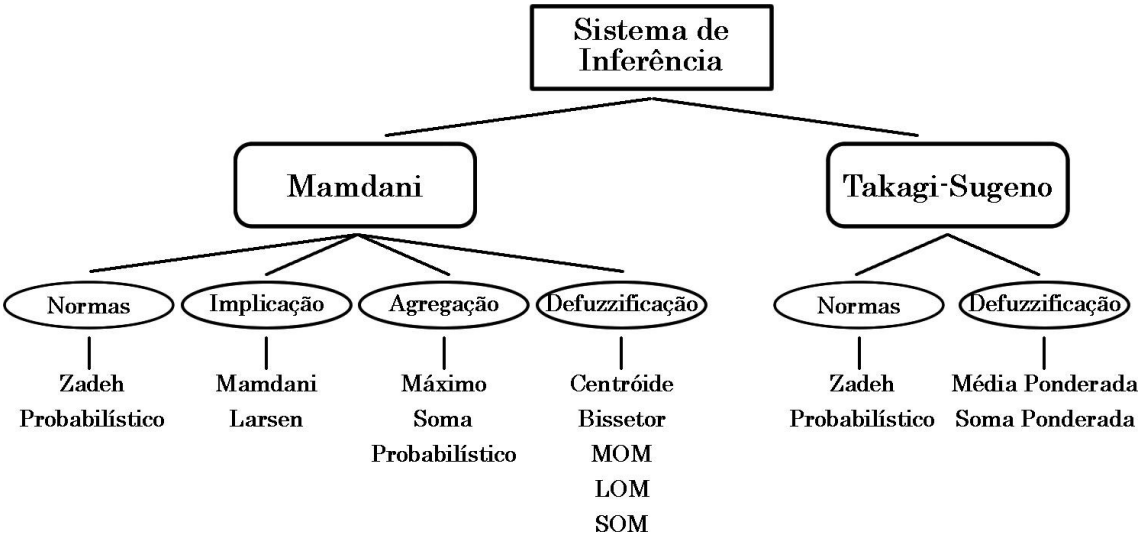

Figura 4.2: Métodos implementados no sistema.

4.1.3 Módulo de Comunicação

Este é o módulo de mais baixo nível do sistema e é responsável pela troca de informações entre o sistema de inferência e o processo.

Este módulo é composto por uma lista de interfaces de comunicação industrial, sendo cada uma delas capaz de comunicar-se com um conjunto de dispositivos habilitados para tal protocolo.

Aqui aplica-se diretamente o conceito de transparência: o sistema de inferência não precisa preocupar-se com detalhes de baixo nível ou detalhes de comunicação. A combinação de várias fontes de dados é realizada neste módulo, e para as camadas mais altas do sistema de inferência, elas comportam-se como variáveis genéricas e semelhantes.

No desenvolvimento deste trabalho, o esforço foi voltado para a implementação da interface de comunicação OPC, justificado pelo fato de este ser um padrão na comunicação industrial. Devido à estrutura modular do sistema, outras interfaces de comunicação podem ser agregadas no futuro, de maneira que o conjunto de interfaces de comunicação do JFuzZ está limitado apenas ao desenvolvimento de cada uma delas.

4.1.4 Módulo Configurador de Tags

Em ambientes industriais, torna-se quase obrigatória a conexão do sistema a um processo real. Esta comunicação é realizada através de um determinado protocolo.

Em alguns casos, a simples ligação de uma variável do processo a uma variável do sistema de inferência é suficiente para a execução da tarefa de controle, mas em alguns casos esta solução não é válida. Muitas vezes é necessário um pré-tratamento da informação antes que ela chegue ao sistema de inferência, adequando assim o dado à solução da tarefa.

Um exemplo prático deste problema é capturar o erro atual do sistema. Em geral, o erro não é uma simples variável do processo, mas sim a diferença entre o *setpoint* e o valor lido de uma variável. Neste caso, uma operação aritmética de subtração é necessária para que o erro seja calculado. O módulo de configuração de *tags* é a entidade responsável por manipular uma ou mais *tags* através de operações aritméticas simples, publicando esta manipulação em uma variável do sistema JFuzZ.

Da forma descrita, o sistema JFuzZ não lê diretamente os valores das variáveis das plantas. Este processo é realizado através de uma interface entre a *engine fuzzy* e o módulo de comunicação.

É importante ressaltar que só é possível manipular variáveis de entrada (no futuro, esta característica será expandida para as variáveis de saída). Dessa maneira, o valor atribuído à variável de saída do sistema é escrito diretamente do processo. Vale lembrar ainda que é possível a combinação de variáveis de diferentes fontes de dados e diferentes protocolos de comunicação. Para o sistema de inferência, será gerada uma variável genérica.

4.1.5 Módulo de Persistência

Esta é a entidade responsável pelo armazenamento e recuperação de todas as informações do sistema de inferência *fuzzy*. Por não haver a necessidade de utilização do sis-

tema de forma distribuída, bem como pela simplicidade dos dados a serem armazenados, definiram-se tais armazenamentos na forma de arquivos.

Todos os parâmetros de configuração para o sistema de inferência *fuzzy* em execução são gravados em um arquivo estrutural, salvo com a extensão ".FUZ". Os dados salvos e carregados na memória para a execução do sistema podem ser divididos em quatro grupos distintos, como mostrado na figura 4.3.

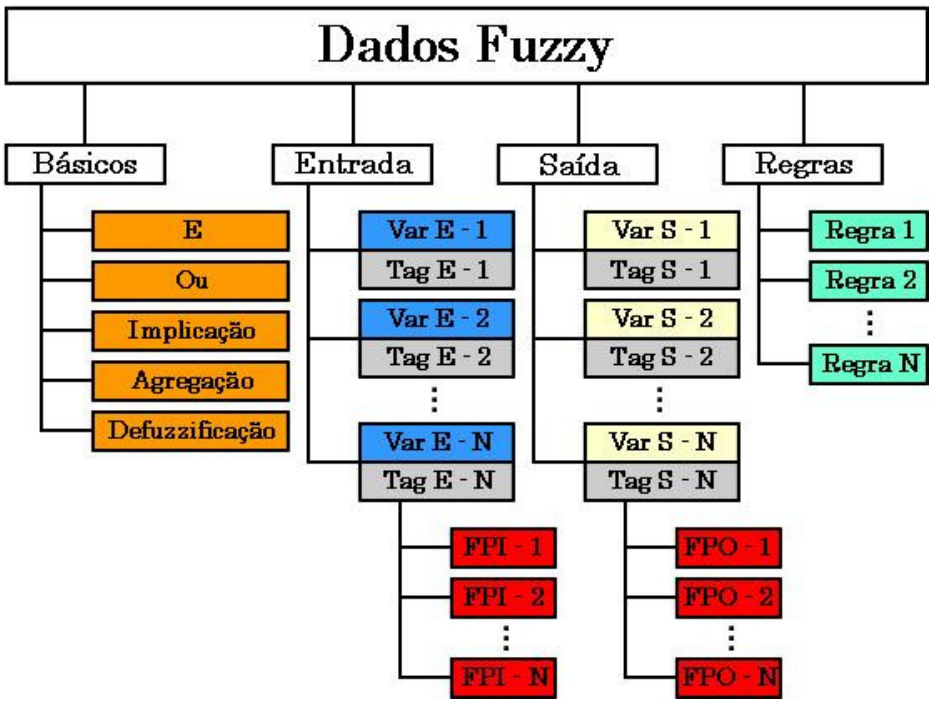

Figura 4.3: Estrutura de dados do JFuzZ.

- **Dados Básicos:** dados que tratam das configurações básicas do sistema de inferência *fuzzy*, tais como tipos de normas, métodos de implicação, agregação e defuzzificação.
- **Entrada:** bloco de dados a respeito das variáveis de entrada. Guarda informações tais como nomes, *ranges*, *tags* pré-tratadas do sistema e informações de servidores e variáveis do processo. Além disso, este bloco é responsável pelo armazenamento das informações de todas as funções de pertinência associadas a cada variável de entrada.
- **Saída:** bloco de dados a respeito das variáveis de saída. Guarda informações tais como nomes, *ranges*, *tags* pré-tratadas do sistema e informações de servidores e variáveis do processo. Além disso, este bloco é responsável pelo armazenamento das informações de todas as funções de pertinência associadas a cada variável de saída.
- **Regras:** dados a respeito das regras de inferência definidas no sistema *fuzzy*.

4.2 Modelagem do Sistema

No processo de modelagem de *software*, algumas ferramentas são utilizadas no auxílio à análise e implementação dos sistemas. Uma destas ferramentas, chamada de UML (*Unified Modeling Language*), apresenta uma linguagem padrão que permite modelar um sistema, bem como visa dotar o mercado de orientação a objetos de uma linguagem única de modelagem.

Um dos principais diagramas da linguagem UML é chamado de diagrama de classes. Esse diagrama é utilizado para representar as diversas classes de objetos do sistema, bem como as associações entre cada uma delas (herança, generalização, composição, agregação) [SOUZA 2005].

O sistema JFuzZ foi desenvolvido utilizando-se os conceitos básicos de programação orientada a objetos. A figura 4.4 ilustra o diagrama de classes resumido do sistema. As principais classes do projeto são detalhadas a seguir:

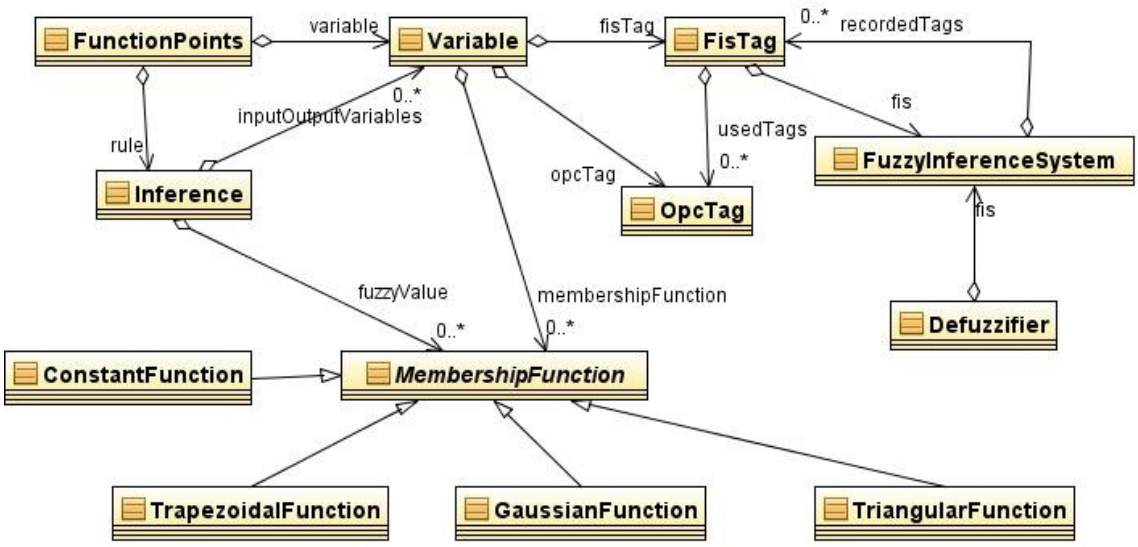

Figura 4.4: Diagrama de classes do JFuzZ.

- ***FuzzyInferenceSystem:*** classe principal do sistema. É a classe que guarda todos os atributos do sistema de inferência, tais como configurações básicas, tipo de sistema, variáveis de entrada e saída, regras, métodos de inferência etc.
- ***Variable:*** classe que modela as variáveis de entrada e saída do sistema. Trata de informações como nome, *range*, funções de pertinência e *tags* ligadas à variável.
- ***MembershipFunction:*** classe de funções de pertinência. Guarda os atributos das funções de pertinência de cada variável, tais como tipo, limites e nome do valor *fuzzy*. *MembershipFunction* é uma classe abstrata, e as classes *ConstantFunction*, *TriangularFunction*, *TrapezoidalFunction* e *GaussianFunction* (classes que modelam os tipos de funções de pertinência implementadas) herdam da mesma.

- ***Inference:*** classe que modela as regras de inferência do sistema *fuzzy*. Guarda atributos das variáveis de entrada e saída envolvidas em cada regra, bem como os valores *fuzzy* e os conectores lógicos associados.
- ***Defuzzifier:*** classe que compreende o processo de defuzzificação de um sistema de inferência. A classe *Defuzzifier* implementa todos os métodos de defuzzificação detalhados anteriormente, e é instanciada a partir de um objeto *FuzzyInferenceSystem*.
- ***OpcTag:*** classe que guarda os atributos das *tags* do protocolo OPC, tais como configurações de nomes e endereços do servidor, nomes das variáveis etc.
- ***FisTag:*** classe responsável pela interface entre as *tags* OPC e as variáveis do sistema. Essa classe realiza a combinação aritmética entre diferentes variáveis da planta e exporta tal combinação em uma variável do sistema.

4.3 Funcionalidades do Sistema

O sistema JFuzZ foi completamente concebido com base em janelas e ambientes gráficos simplificados, tornando intuitiva a criação de sistemas de inferência *fuzzy*.

O sistema desenvolvido possui basicamente cinco ambientes ambientes gráficos principais para edição, visualização e execução dos sistemas de inferência:

- Editor básico de sistemas de inferência *fuzzy*;
- Editor de funções de pertinência;
- Editor de regras de inferência;
- Visualizador de regras e
- Módulo de controle.

Tais ambientes gráficos são dinamicamente ligados, e todas as mudanças realizadas no sistema de inferência utilizando um dos ambientes serão transmitidas aos outros.

4.3.1 Editor básico de sistemas de inferência *fuzzy*

O editor de sistemas de inferência trabalha com os problemas de alto-nível do sistema, tais como o número de variáveis de entrada e saída, os nomes das variáveis, os *links* de variáveis do sistema com variáveis OPC e as configurações básicas do sistema de inferência. O JFuzZ não limita o número de variáveis, entretanto, este valor máximo estará diretamente relacionado à capacidade da máquina.

A figura 4.5 ilustra o ambiente gráfico de edição de sistemas de inferência *fuzzy*. A seguir, são especificadas as numerações que aparecem na figura.

1. **Barra de títulos:** exibe o tipo do sistema de inferência e o nome do arquivo aberto atualmente. Se o sistema de inferência ainda não foi salvo, o título mostrado é *Untitled.fuz*.
2. **Barra de menus:** permite abrir, salvar e editar sistemas de inferência *fuzzy*, bem como acessar os outros ambientes gráficos.
3. **Variáveis de entrada:** exibe as variáveis de entrada do sistema, permitindo a adição, remoção e edição de variáveis.

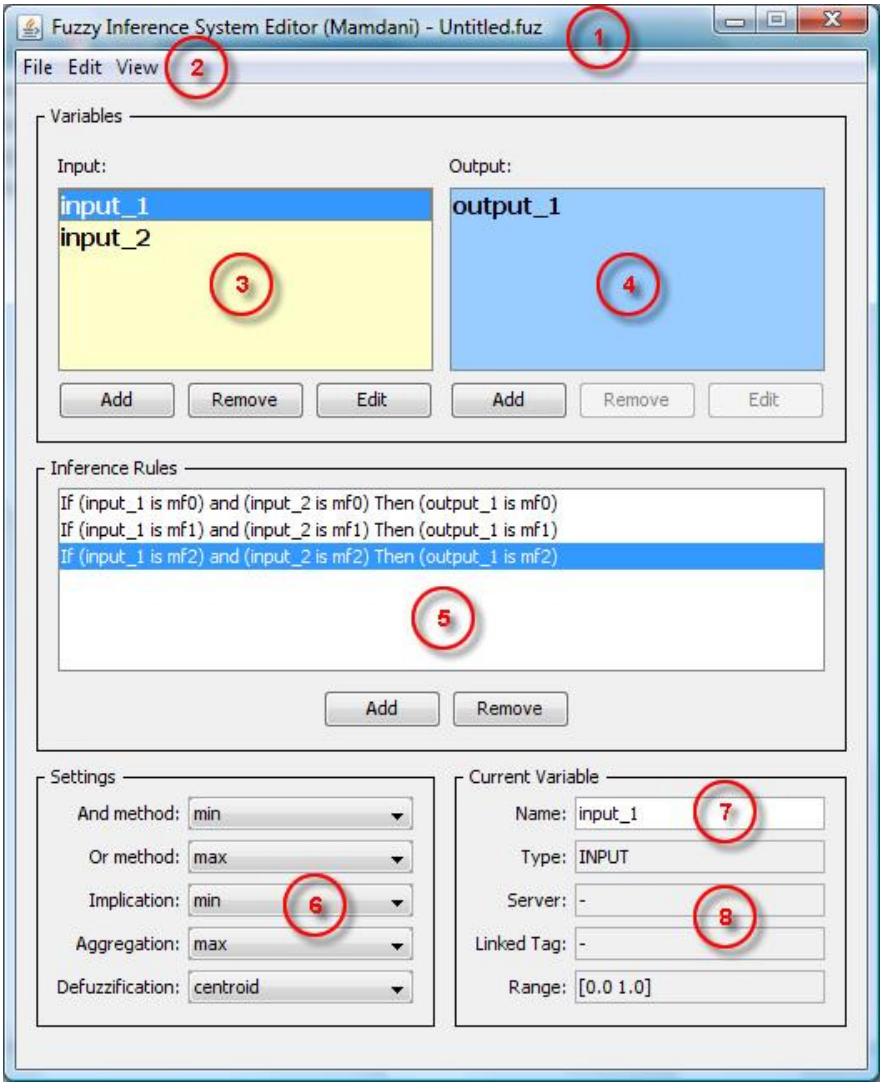

Figura 4.5: Ambiente de edição de sistemas de inferência.

4. **Variáveis de saída:** exibe as variáveis de saída do sistema, permitindo a adição, remoção e edição de variáveis.
5. **Regras de inferência:** exibe as regras de inferência do tipo *Se-Então* do sistema, permitindo a adição e remoção de regras, e dá acesso ao ambiente gráfico de edição de regras.
6. **Configurações do sistema:** seleção de métodos e ajustes das funções de inferência.
7. **Nome da variável:** este campo de texto é usado para mostrar e modificar o nome de uma variável de entrada ou saída do sistema.
8. **Campos informativos:** exibem outras informações sobra a variável, tais como o seu tipo, a *tag* à qual ela está ligada, seus limites de operação etc.

4.3.2 Editor de funções de pertinência

O editor de funções de pertinência é a ferramenta que permite a visualização e edição de todas as funções de pertinência associadas a todas as variáveis de entrada e saída do sistema. Nesse ambiente é possível modelar as funções de acordo com os tipos disponíveis e com os parâmetros de configuração e de *range*.

A figura 4.6 ilustra o ambiente gráfico de edição funções de pertinência. A seguir, são especificadas as funcionalidades numeradas nessa figura.

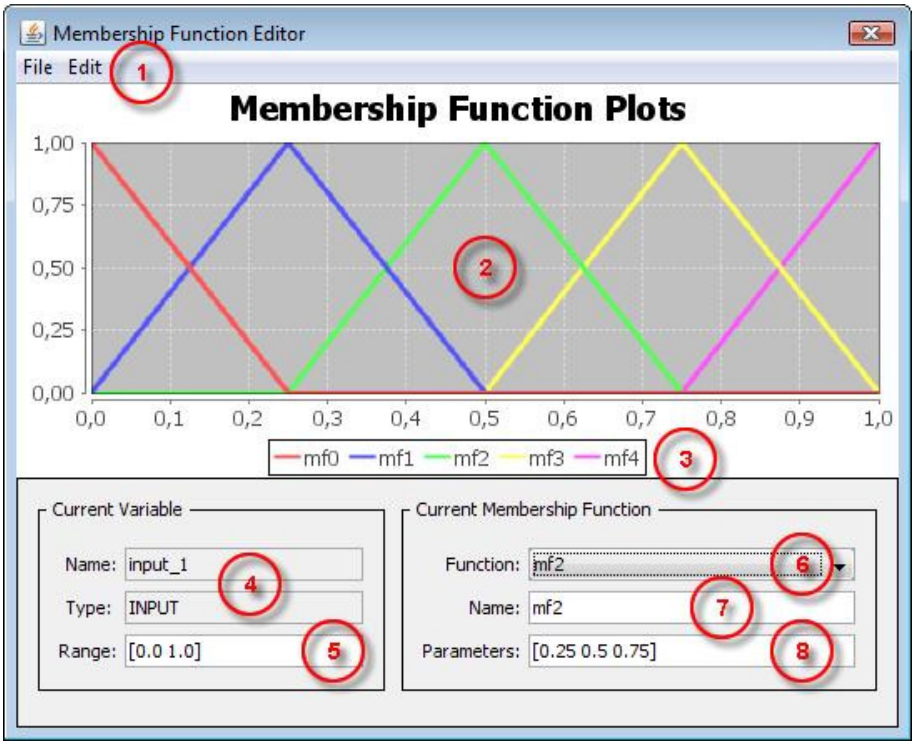

Figura 4.6: Ambiente de edição funções de pertinência.

1. **Barra de menus:** permite adicionar, remover e salvar funções de pertinência do sistema de inferência.
2. **Gráfico principal:** este gráfico exibe todas as funções de pertinência da variável selecionada.
3. **Legenda:** relaciona as funções mostradas no gráfico aos seus respectivos nomes.
4. **Nome e tipo da variável:** campos de texto para exibição do nome e tipo da variável selecionada.
5. **Range:** define os limites mínimo e máximo de valores que a variável poderá atingir.
6. **Função atual:** caixa de seleção para as funções de pertinência existentes.
7. **Nome da função:** este campo de texto é usado para mostrar e modificar o nome da função de pertinência selecionada.
8. **Parâmetros:** campo utilizado para configuração dos parâmetros de modelagem do comportamento da função. O número de parâmetros depende do tipo da função de pertinência, e devem ser sempre circundados por colchetes.

4.3.3 Editor de regras de inferência

O ambiente foi concebido de forma que a construção de regras de inferência a partir do ambiente gráfico fosse uma tarefa simples. Baseado nas descrições das variáveis de entrada e saída definidas no editor básico, o editor de regras permite criar regras automaticamente, apenas selecionando e associando cada variável a um valor *fuzzy*.

Clicando sobre a caixa que representa cada variável, o editor de regras irá negar o valor *fuzzy* escolhido, através da operação de complemento. A associação de um valor *fuzzy* a uma variável é opcional. No caso em que nenhum valor é escolhido, a variável em questão é ignorada para aquela regra na etapa de inferência.

A figura 4.7 ilustra o ambiente gráfico de edição de regras de inferência. As funcionalidades numeradas nessa figura são especificadas a seguir:

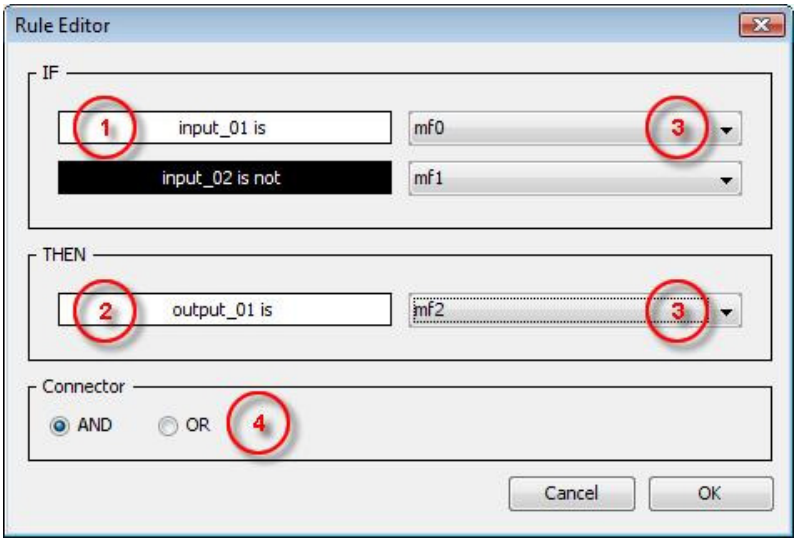

Figura 4.7: Ambiente de edição regras de inferência.

1. **Variáveis de entrada:** caixas de texto contendo os nomes de todas as variáveis de entrada do sistema para a criação da regra de inferência.
2. **Variáveis de saida:** caixas de texto contendo os nomes de todas as variáveis de saída do sistema para a criação da regra de inferência.
3. **Valores *fuzzy*:** caixas contendo todos os valores *fuzzy* disponíveis para cada variável.
4. **Conector lógico:** conector de ligação das associações das variáveis de entrada com os seus respectivos valores *fuzzy*.

4.3.4 Visualizador de regras

O visualizador de regras apresenta um roteiro interativo do processo completo de inferência. Este módulo está diretamente ligado ao sistema de inferência *fuzzy* criado.

O ambiente de visualização de regras possui uma interface dinâmica, que apresenta os antecedentes e consequentes de cada regra. Para cada uma delas, os valores *fuzzy*

selecionados para as variáveis de entrada e saída são ilustrados na forma de gráficos, de acordo com as funções de pertinência associadas.

Além dos antecedentes e consequentes de cada regra, o ambiente de visualização gera a forma que representa a operação de agregação, calculando e identificando os valores para as variáveis de saída, de acordo com as entradas do sistema e dos parâmetros gerais de configuração. Os valores para as variáveis de entrada podem ser atribuídos dinamicamente, sendo o sistema capaz de mostrar em tempo real as alterações sofridas em todas as etapas da inferência.

O visualizador de regras permite ao usuário interpretar um processo de inferência *fuzzy* inteiro de uma única vez. Este ambiente também ilustra graficamente como a forma de determinadas funções de pertinência influenciam no resultado final da inferência. Como este módulo ilustra graficamente cada parte de cada regra, a visualização como um todo pode se tornar inviável para sistemas muito grandes.

A figura 4.8 ilustra o ambiente gráfico de visualização de regras de inferência, possuindo os pontos numerados os seguintes significados:

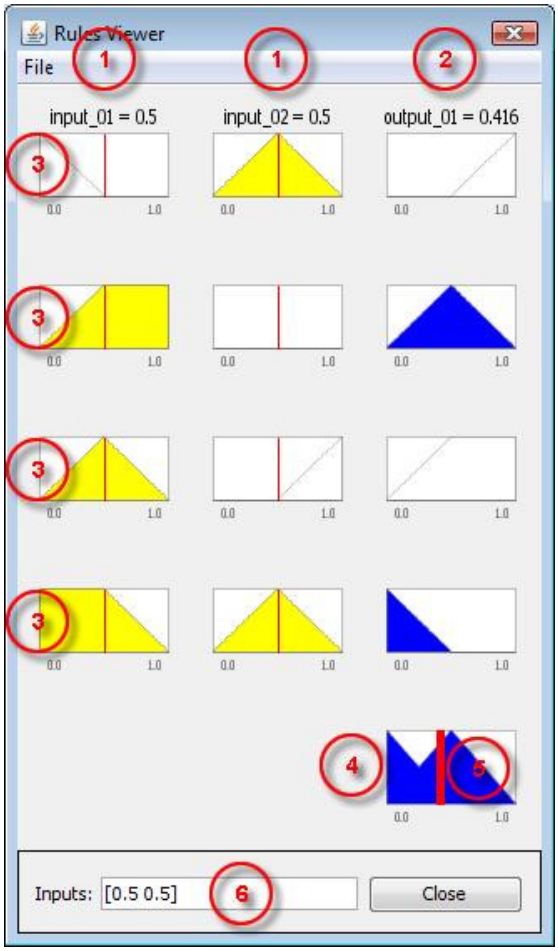

Figura 4.8: Ambiente de visualização regras de inferência.

1. **Antecedentes:** cada coluna gráfica amarela mostra como a variável de entrada é

usada em cada regra. Os valores para cada variável de entrada são mostrados no topo da coluna.

2. **Consequentes:** cada coluna gráfica azul mostra como a variável de saída é usada em cada regra. Os valores para cada variável de saída são mostrados no topo da coluna.
3. **Regras de inferência:** cada linha de gráficos representa uma regra de inferência.
4. **Forma de saída agregada:** o último gráfico da tela mostra como a saída de cada regra é combinada para formar a saída agregada.
5. **Defuzzificação:** a linha indica o valor de saída do defuzzificador para a forma de saída gerada.
6. **Entradas:** caixa de texto para inserção de valores de entradas para o sistema.

4.3.5 Módulo de Controle

O módulo de controle é o ambiente de execução dos sistemas de inferência *fuzzy* aplicados a um determinado processo industrial. Após criado no editor básico de sistemas de inferência, tal sistema *fuzzy* deve ser carregado no módulo de controle para início da execução.

Nesse ambiente, o operador pode monitorar o processo de controle através de informações numéricas e gráficos de tendências em tempo real. Além disso, o operador pode alterar os valores de referência do sistema a qualquer momento, sendo o controlador capaz de ajustar-se para seguir a nova referência.

A figura 4.9 ilustra o ambiente gráfico do módulo de controle *fuzzy*, sendo que os campos numerados têm os seguintes significados:

1. **Configuração do sistema de inferência atual:** botão de acesso ao editor básico de sistemas de inferência. A partir dele, o editor carregará o sistema de inferência atualmente utilizado no módulo de controle. Se nenhum sistema de inferência *fuzzy* encontra-se carregado, o editor básico criará um novo.
2. **Carregamento de sistema de inferência existente:** botão de acesso ao editor básico de sistemas de inferência. A partir dele, o módulo de controle carregará um sistema de inferência previamente definido, a partir de um arquivo ".FUZ".
3. **Informações básicas:** informações sobre o sistema de inferência carregado.
4. **Setpoint:** campo de definição da referência do sistema de controle.
5. **Dados de variáveis:** valores de tempo-real das variáveis do sistema.
6. **Período de atualização:** intervalo de atualização do sistema de controle, dado em milissegundos. Pode ser limitado à taxa de atualização do meio de comunicação.
7. **Execução da tarefa:** inicia e finaliza a execução da tarefa de controle, de acordo com o período de atualização definido.
8. **Variáveis ativas:** lista de variáveis ativas, e passíveis de monitoramento.
9. **Geração de gráficos:** a partir de seleção realizada na lista das variáveis, gera um gráfico de tendência com os valores das variáveis, atualizado a cada ciclo de controle. Vários gráficos podem ser gerados, com diferentes combinações de variáveis, estando esta funcionalidade limitada à capacidade de processamento da máquina.

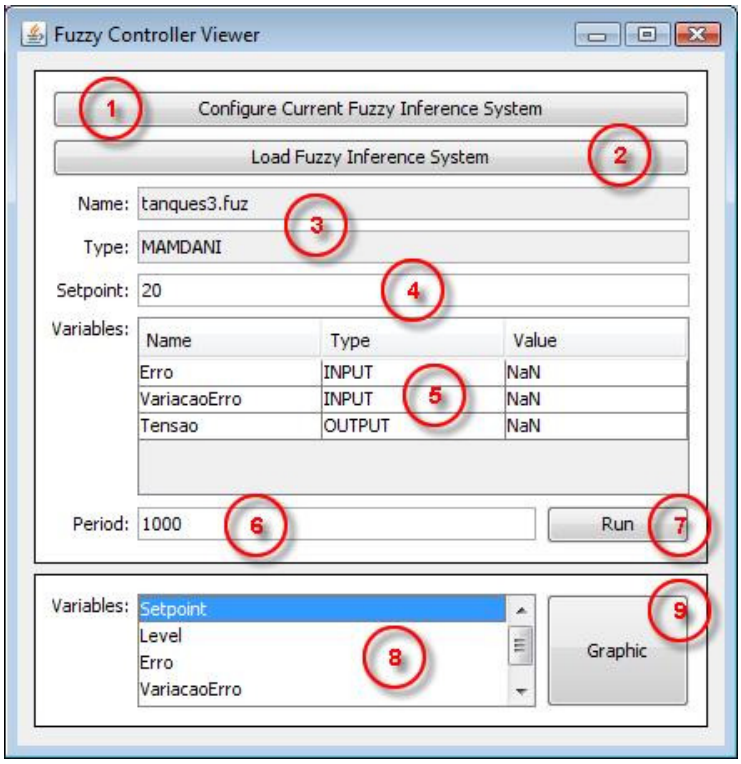

Figura 4.9: Ambiente de visualização do controlador *fuzzy*.

Capítulo 5

Validação e Resultados Obtidos

Neste capítulo serão apresentados e analisados alguns resultados obtidos na validação do JFuzZ. Tais resultados têm o objetivo de verificar a exatidão das saídas dos sistemas de inferência criados, bem como a eficácia do sistema quando utilizado em problema de controle real.

5.1 Validação Estática

Para a validação preliminar do *software* desenvolvido, fez-se utilização do *Matlab Fuzzy Logic Toolbox*, servindo como ferramenta de comparação. Uma aplicação *fuzzy* real foi criada no *toolbox* do Matlab e, em seguida, no ambiente JFuzZ. Todos os parâmetros de configuração foram atribuídos de forma idêntica às duas aplicações, e um conjunto de entradas aleatórias foi gerado e inserido em ambos os ambientes. As saídas foram plotadas em gráficos para comparação.

5.1.1 Descrição do Problema

O problema de exemplo proposto [SANKAR & KUMAR n.d.] foca-se em uma nova alternativa para os sistemas de freios de trens, baseada na lógica *fuzzy*. Normalmente, as companhias de trens indianas utilizam duas pessoas para operar o sistema, e empregam um procedimento manual para parar o trem em cada estação. O controlador *fuzzy* deve ajudar na redução do trabalho humano na operação dos trens.

A aplicação *fuzzy* deve gerar uma única saída, correspondente à força do freio (variando de 0 a 100%), a partir da distância do trem à estação e da velocidade em que o trem se encontra.

5.1.2 Modelagem do Sistema

O sistema de inferência *fuzzy* para controle de freio de trens, é modelado a seguir. Para o exemplo proposto, todas as funções de pertinência foram modeladas a partir de funções trapezoidais.

Esta seção detalha a modelagem do sistema de inferência no ambiente JFuzZ. O detalhamento da modelagem do sistema no ambiente do Matlab foge do escopo deste trabalho.

Tipos de Sistema

Mamdani e Takagi-Sugeno.

Variáveis de Entrada

Distância e Velocidade.

Variáveis de Saída

Força do freio (%).

Funções de Pertinência

1. Distância
   - Limites: [0.0 500.0];
   - Valores fuzzy: muito perto, perto, longe, muito longe;
   - Parâmetros: $[-500.0\ -10.0\ 0.0\ 20.0]$, [10.0 30.0 50.0 120.0], [60.0 110.0 160.0 350.0], [250.0 400.0 500.0 1000.0];
   - Funções estão ilustradas na figura 5.1.
2. Velocidade
   - Limites: [0.0 100.0];
   - Valores fuzzy: muito devagar, devagar, rápido, muito rápido;
   - Parâmetros: $[-100.0\ -10.0\ 0.0\ 5.0]$, [2.5 20.0 20.0 30.0], [20.0 40.0 45.0 60.0], [45.0 55.0 110.0 200.0];
   - Funções estão ilustradas na figura 5.2.
3. Força do freio (%)
   - Limites: [0.0 100.0];
   - Valores fuzzy: muito leve, leve, pesado, muito pesado;
   - Parâmetros: $[-100.0\ -10.0\ 8.0\ 20.0]$, [0.0 20.0 20.0 50.0], [40.0 60.0 60.0 80.0], [70.0 90.0 110.0 200.0];
   - Funções estão ilustradas na figura 5.3.

Regras de Inferência

1. Se distância é **MUITO PERTO** e velocidade é **MUITO DEVAGAR** então freio é **LEVE**
2. Se distância é **MUITO PERTO** e velocidade é **DEVAGAR** então freio é **PESADO**
3. Se distância é **MUITO PERTO** e velocidade é **RÁPIDO** então freio é **MUITO PESADO**
4. Se distância é **MUITO PERTO** e velocidade é **MUITO RÁPIDO** então freio é **LEVE**
5. Se distância é **PERTO** e velocidade é **MUITO DEVAGAR** então freio é **LEVE**

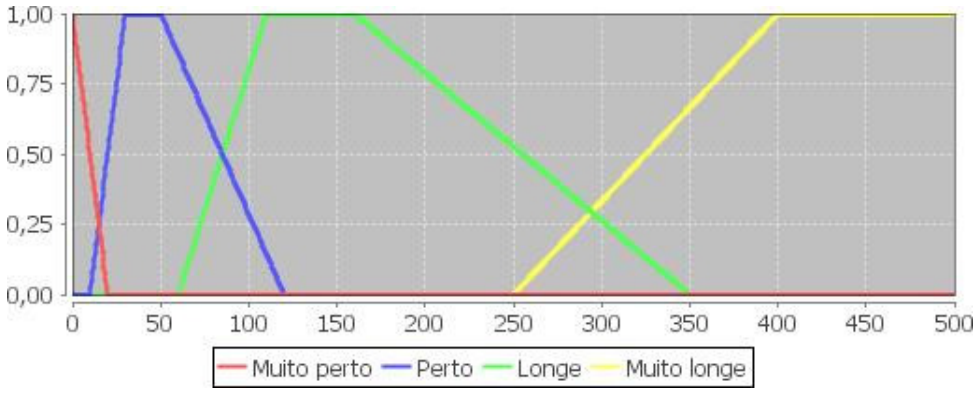

Figura 5.1: Funções de pertinência da variável *Distância*.

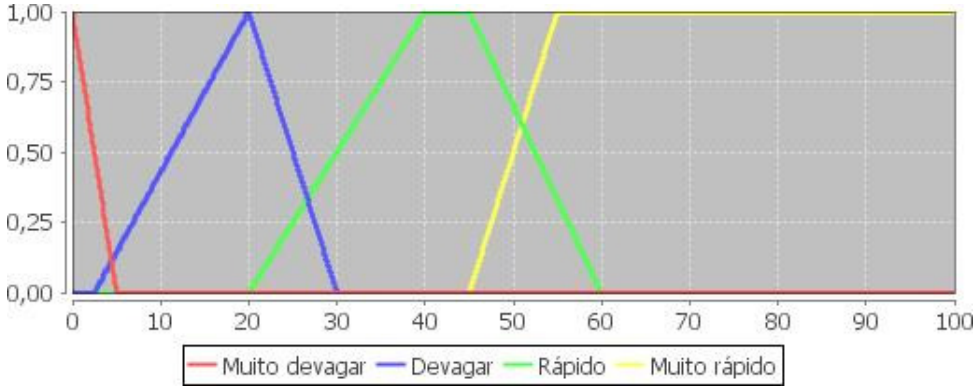

Figura 5.2: Funções de pertinência da variável *Velocidade*.

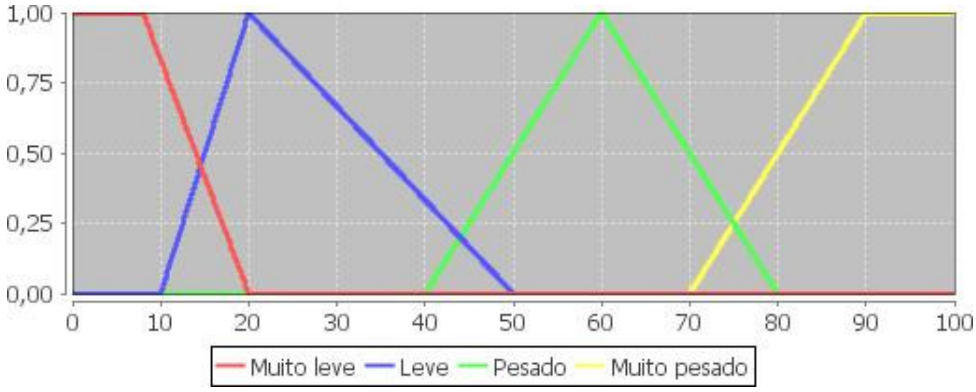

Figura 5.3: Funções de pertinência da variável *Força do freio (%)*.

6. Se distância é **PERTO** e velocidade é **DEVAGAR** então freio é **LEVE**
7. Se distância é **PERTO** e velocidade é **RÁPIDO** então freio é **PESADO**
8. Se distância é **PERTO** e velocidade é **MUITO RÁPIDO** então freio é **MUITO PESADO**
9. Se distância é **LONGE** e velocidade é **MUITO DEVAGAR** então freio é **LEVE**
10. Se distância é **LONGE** e velocidade é **DEVAGAR** então freio é **MUITO LEVE**

11. Se distância é **LONGE** e velocidade é **RÁPIDO** então freio é **LEVE**
12. Se distância é **LONGE** e velocidade é **MUITO RÁPIDO** então freio é **PESADO**
13. Se distância é **MUITO LONGE** e velocidade é **MUITO DEVAGAR** então freio é **MUITO LEVE**
14. Se distância é **MUITO LONGE** e velocidade é **DEVAGAR** então freio é **MUITO LEVE**
15. Se distância é **MUITO LONGE** e velocidade é **RÁPIDO** então freio é **LEVE**
16. Se distância é **MUITO LONGE** e velocidade é **MUITO RÁPIDO** então freio é **LEVE**

Parâmetros de Configuração para Sistema de Inferência de Mamdani

- Método de Conjunção: *minimum* (Zadeh);
- Método de Disjunção: *maximum* (Zadeh);
- Método de Implicação: *minimum* (Mamdani);
- Método de Agregação: *maximum*;
- Método de Defuzzificação: *centroid*.

Parâmetros de Configuração para Sistema de Inferência de Takagi-Sugeno

- Método de Conjunção: *minimum* (Zadeh);
- Método de Disjunção: *maximum* (Zadeh);
- Método de Ponderação: *weighted average* (Média Ponderada).

5.1.3 Procedimentos Realizados e Resultados Obtidos

Para efeito de comparação, um conjunto de 100 pares de entradas aleatórias (100 para cada variável de entrada do sistema) foi gerado e aplicado a cada um dos sistemas de inferência, em ambos os *software*s. O processo de geração de valores procurou abranger grandes variações dentro do *range* de cada variável de entrada. O conjunto de entradas aplicado é mostrado a seguir:

0: [326.14, 73.82]	1: [146.77, 76.95]	2: [173.23, 14.75]	3: [210.20, 17.64]
4: [475.17, 17.83]	5: [69.21, 39.42]	6: [455.57, 60.13]	7: [68.24, 91.63]
8: [254.08, 89.01]	9: [309.96, 27.08]	10: [394.74, 4.58]	11: [193.37, 77.47]
12: [261.72, 15.34]	13: [244.49, 94.49]	14: [401.07, 25.70]	15: [493.99, 91.70]
16: [248.61, 80.77]	17: [320.59, 28.27]	18: [56.65, 63.14]	19: [484.16, 34.17]
20: [355.01, 9.39]	21: [18.4, 88.18]	22: [235.73, 89.79]	23: [239.39, 25.16]
24: [94.62, 73.67]	25: [353.22, 66.95]	26: [198.31, 85.20]	27: [13.64, 42.57]
28: [70.22, 39.42]	29: [189.14, 87.70]	30: [205.68, 36.71]	31: [347.47, 90.72]
32: [32.62, 85.68]	33: [255.58, 83.36]	34: [331.88, 18.67]	35: [122.58, 57.59]
36: [19.25, 62.17]	37: [384.78, 27.09]	38: [269.69, 25.14]	39: [142.71, 41.97]
40: [433.66, 72.27]	41: [112.44, 7.85]	42: [55.25, 56.41]	43: [446.04, 57.29]
44: [52.32, 94.49]	45: [409.34, 70.32]	46: [259.90, 84.85]	47: [179.14, 43.61]
48: [70.02, 80.23]	49: [436.41, 63.23]	50: [323.19, 50.34]	51: [487.52, 44.61]
52: [420.18, 90.77]	53: [340.82, 65.34]	54: [325.86, 60.74]	55: [2.96, 31.07]
56: [231.50, 61.34]	57: [235.73, 83.13]	58: [126.64, 97.84]	59: [132.53, 77.52]
60: [407.07, 5.22]	61: [131.21, 39.30]	62: [411.67, 54.06]	63: [479.57, 23.74]
64: [437.53, 95.73]	65: [208.95, 96.41]	66: [330.31, 50.39]	67: [351.58, 77.71]
68: [184.38, 88.92]	69: [458.38, 84.57]	70: [16.39, 10.94]	71: [435.08, 41.64]
72: [494.89, 75.05]	73: [316.63, 27.51]	74: [418.39, 94.46]	75: [342.68, 67.79]
76: [134.98, 71.55]	77: [80.07, 85.99]	78: [487.67, 8.23]	79: [362.22, 75.64]
80: [151.17, 95.42]	81: [479.94, 62.89]	82: [398.47, 54.74]	83: [363.25, 87.06]
84: [398.31, 78.02]	85: [439.65, 38.96]	86: [98.47, 86.51]	87: [440.78, 61.22]
88: [93.94, 10.07]	89: [156.27, 33.44]	90: [113.76, 69.50]	91: [1.49, 35.71]
92: [386.12, 29.73]	93: [273.97, 8.68]	94: [157.14, 80.42]	95: [122.43, 5.69]
96: [45.61, 91.80]	97: [41.73, 62.92]	98: [213.39, 88.38]	99: [137.24, 97.57]

Após o processo de inferência realizado para cada tipo de sistema, em cada um dos ambientes de *software*, o comportamento da variável de saída foi plotado em gráficos de linha. A figura 5.4 representa as saídas para o sistema de inferência do tipo Mamdani. A figura 5.5 representa as saídas para o sistema de inferência do tipo Takagi-Sugeno.

O módulo do erro calculado entre as saídas dos dois *software*s foi plotado em gráficos de linha. A figura 5.6 ilustra o gráfico de erro para o sistema de inferência do tipo Mamdani. A figura 5.7 ilustra o gráfico de erro para o sistema de inferência do tipo Takagi-Sugeno.

Os erros médios calculados foram de 0.086% para o sistema do tipo Mamdani e de 0.071% para o sistema do tipo Takagi-Sugeno.

5.2 Tarefa de Controle Real

Após a validação estática do sistema, uma tarefa de controle real foi implementada com o intuito de verificar o comportamento do *software* para entradas dinâmicas, bem como a sua integração com uma planta real.

Figura 5.4: Saídas para o sistema de inferência do tipo Mamdani.

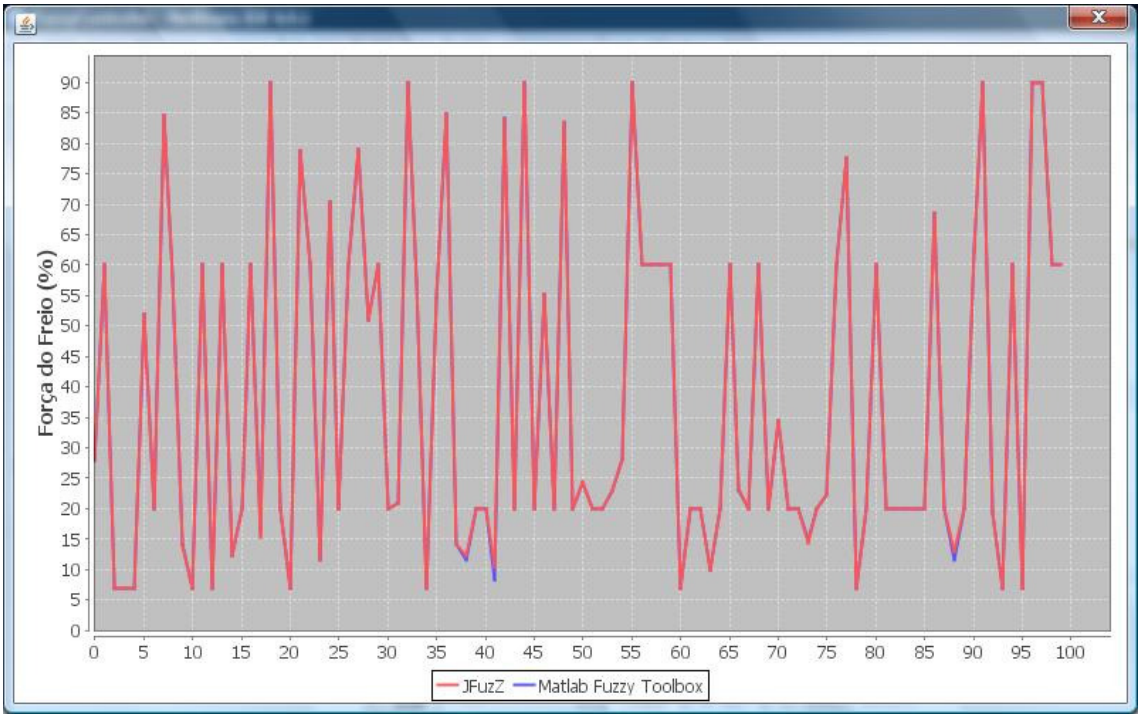

Figura 5.5: Saídas para o sistema de inferência do tipo Takagi-Sugeno.

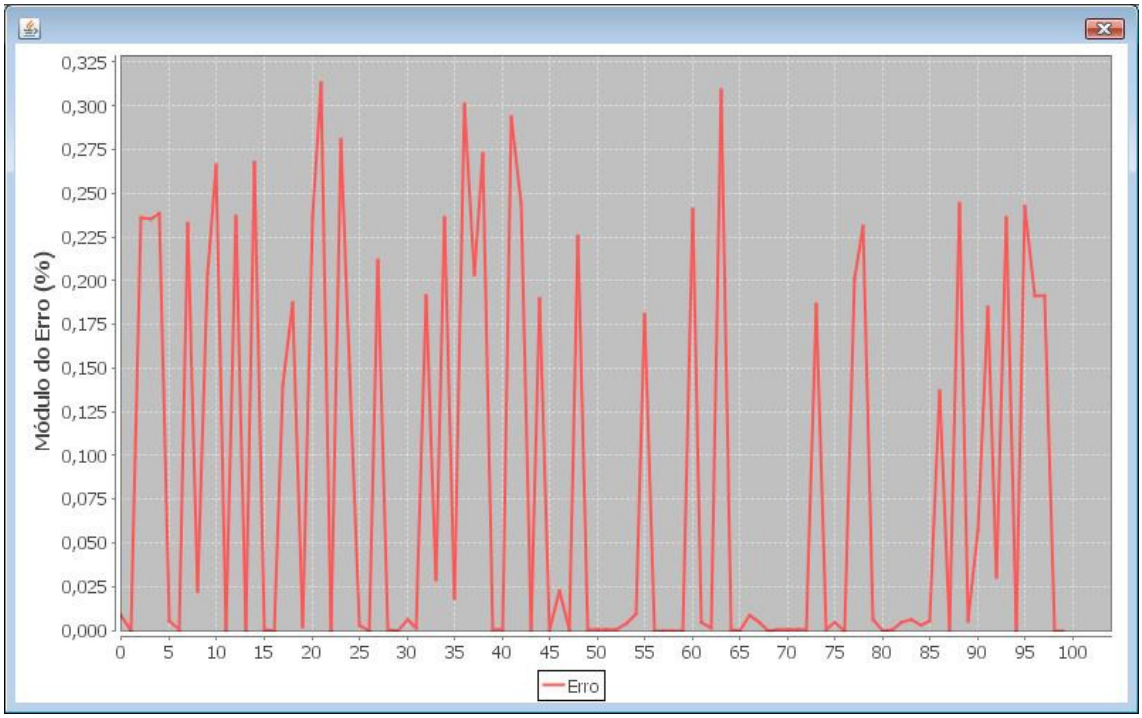

Figura 5.6: Erro para o sistema de inferência do tipo Mamdani.

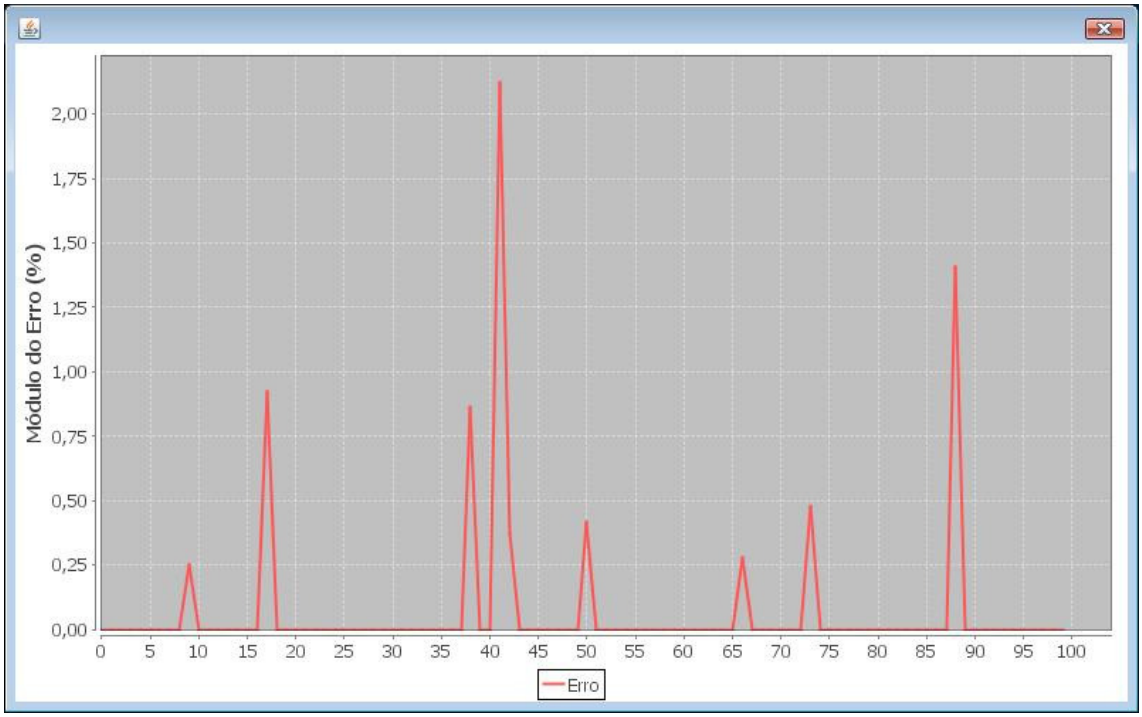

Figura 5.7: Erro para o sistema de inferência do tipo Takagi-Sugeno.

5.2.1 Descrição do Problema

Um problema de controle comum na indústria petroquímica é o controle de nível de líquidos em tanques de armazenamento e recipientes de reações. Em proporções me-

nores, podemos trabalhar com este problema fazendo uso de uma planta experimental, denominada Tanques de Água Acoplados da Quanser [QUANSER 2004].

O módulo de dois tanques consiste de uma bomba acoplada a uma bacia de água. A bomba puxa a água verticalmente para dois pequenos orifícios. Dois tanques montados na face frontal do equipamento são configurados de tal forma que o fluxo do primeiro tanque desce para o segundo, que por sua vez, despeja a água na bacia.

O objetivo desta etapa de validação é o projeto de um sistema de controle inteligente que regule o nível da água no segundo tanque (sistema de segunda ordem), com o *set-point* igual a 10cm. Para esse problema, três soluções foram implementadas, analisadas e comparadas entre si: o controle PID, o controle *fuzzy* direto e o controle adaptativo.

5.2.2 Arquitetura da Aplicação

O sistema de controle implementado é formado por elementos de *hardware* e *software*. Dentre os elementos de *hardware* podemos citar um CLP, uma interface de conversão entre corrente e tensão, um sensor de pressão do módulo de tanques, uma bomba e um módulo de potência. Os principais elementos de *software* do sistema são o JFuzZ e o servidor OPC.

A figura 5.8 ilustra a arquitetura do sistema de controle implementado. Um detalhe neste esquema é a presença de uma interface de conversão corrente-tensão. Tal circuito foi adicionado devido ao fato de a saída do CLP utilizado gerar valores de corrente, e a entrada do módulo de potência receber valores de tensão.

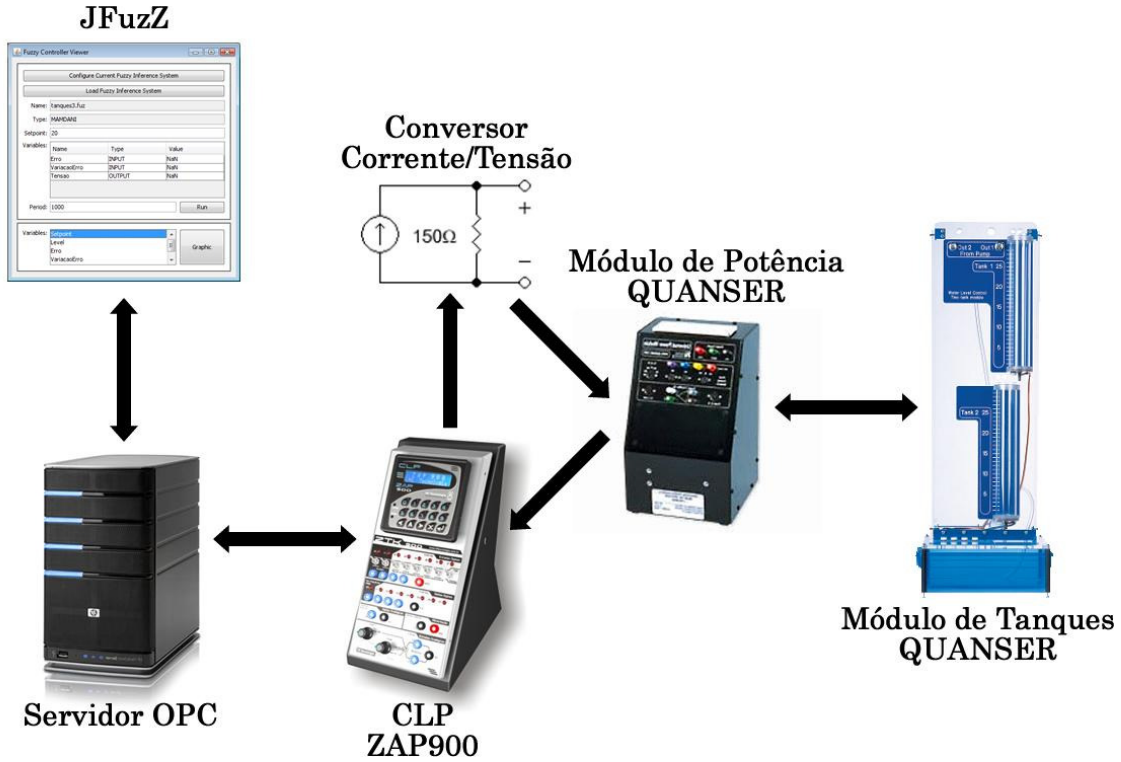

Figura 5.8: Arquitetura da aplicação de controle.

Nessa aplicação, o CLP recebe os dados de medição do sensor do tanque e realiza um controle PID, através de um controlador implementado no próprio CLP. O sinal de controle gerado pelo CLP corresponde a um valor de corrente que varia entre $4mA$ e $20mA$ e que, convertido em tensão, pode chegar ao máximo de $3V$. Esta tensão é aplicada ao módulo de potência que, por sua vez, aplica à bomba um valor cinco vezes maior que o da sua entrada. Para garantir um funcionamento seguro, a tensão máxima aplicável à bomba é de $15V$.

5.2.3 Configurações de Controle no CLP

Para realizar o controle do sistema de tanques de água, utilizou-se um Controlador Lógico Programável da HI Tecnologia (`http://www.hitecnologia.com.br`), modelo ZAP900. Com este CLP, além de muitas outras características, é possível a criação de controladores PID através da linguagem Ladder. O *software* de desenvolvimento de programas em Ladder disponibilizado pela HI Tecnologia é o SPDSW.

O primeiro passo para configuração do CLP é a habilitação das entradas e saídas analógicas, permitindo assim, a comunicação entre o CLP e os sensores e atuadores da planta. Esta tarefa é realizada informando ao CLP a existência de um módulo extra com esse propósito, chamado de HMX500. A etapa é ilustrada na figura 5.9.

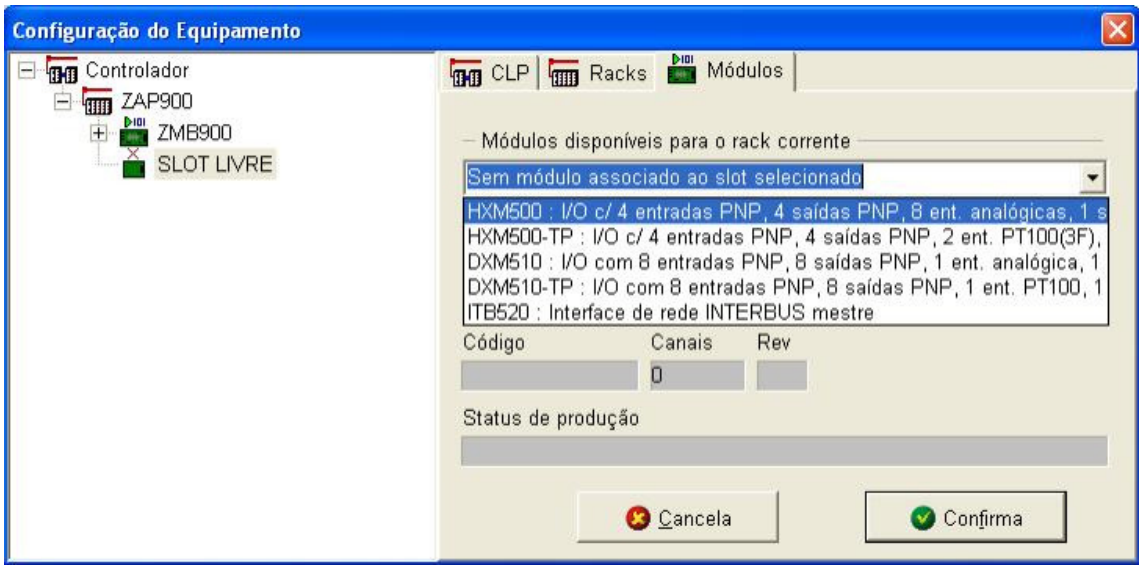

Figura 5.9: Janela de configuração do SPDSW, utilizada para habilitação das entradas e saídas analógicas.

Com o módulo configurado, o passo seguinte é a construção de um programa em Ladder capaz de receber nas entradas analógicas do CLP os valores de tensão dos sensores do módulo de tanques, executando um controle PID sobre a planta através de uma tensão de controle aplicada à bomba d'água. A figura 5.10 ilustra o programa desenvolvido.

A chave I0 é a responsável pela ativação das entradas analógicas e do controlador PID. Os valores de entrada dos sensores de pressão dos tanques de cima e de baixo são

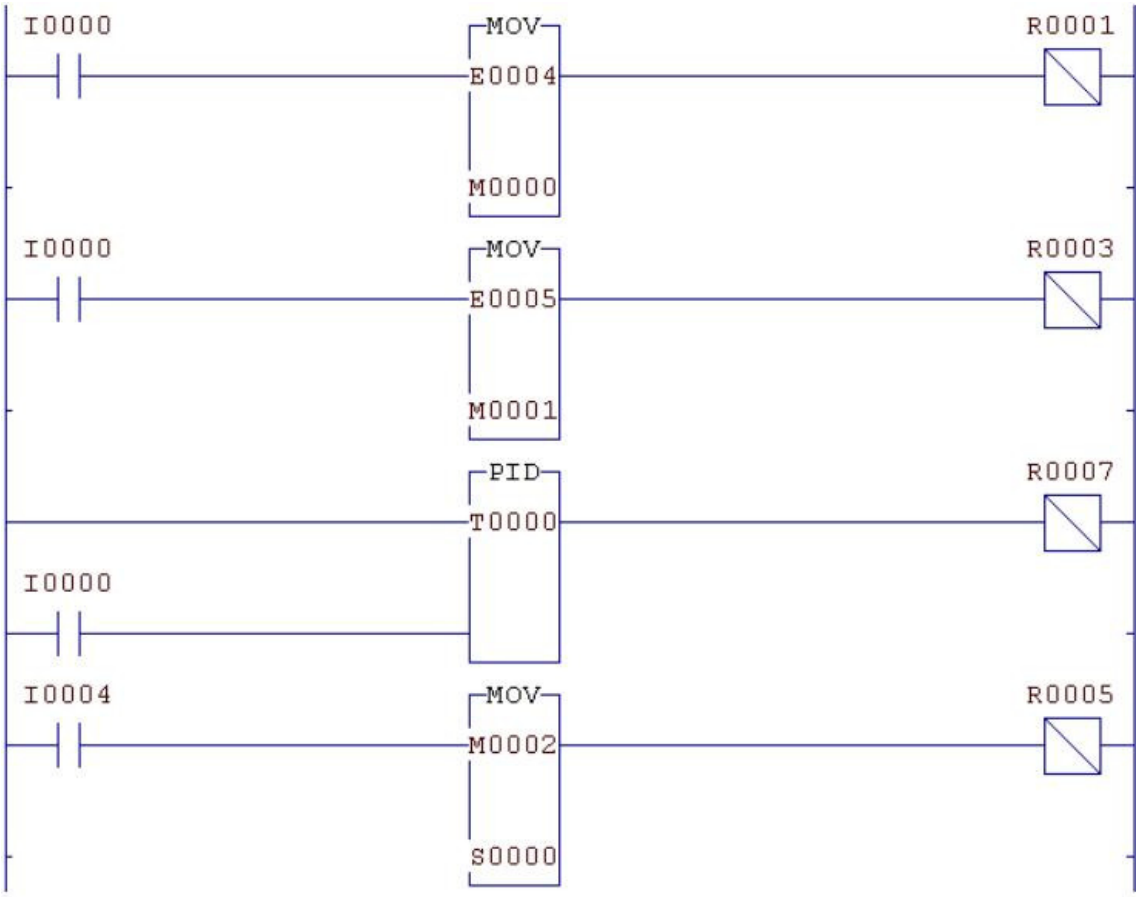

Figura 5.10: Programa em Ladder para controle PID do módulo de tanques.

movidos para as memórias *inteiras* M0 e M1, respectivamente. Após ativado o contato I0, o controle da planta é executado com intervalo fixo de 100*ms*. O contato I4 ativa a saída analógica S0, que é alimentada com o valor da memória M2.

A etapa seguinte foca-se na configuração do controlador PID, implementado no CLP. A figura 5.11 mostra a janela de configuração do controlador no SPDSW. Tal janela dá acesso à configuração de diversos parâmentros do controlador, tais como ganhos, limites, tipo de controle e tipo de saída.

Na caixa *Parâmetros* é possível *linkar* as variáveis de processo ao controlador. Isso é feito atribuindo-se os endereços de memória das variáveis aos parâmetros requeridos no *software*. Nesta implementação, o parâmetro PV (variável do processo) está ligado à posição de memória M0, o parâmetro OV (variável de saída) está ligado ao endereço M2 e o parâmetro SP (*setpoint*) está ligado à posição M4.

Na caixa *Ganhos* indicamos os endereços da memória do CLP responsáveis pelo armazenamento dos valores de ganho proporcional, tempo integrativo e tempo derivativo (GP, TI e TD, respectivamente). Diferentemente do item anterior, as memórias ligadas aos ganhos devem ser to tipo *real*. Nesta aplicação, as memórias selecionadas são endereçadas em D1, D2 e D3.

Os itens de *Limites*, *Ativação de Bumpless*, *Tipo de Controle* e *Tipo de saída* não são

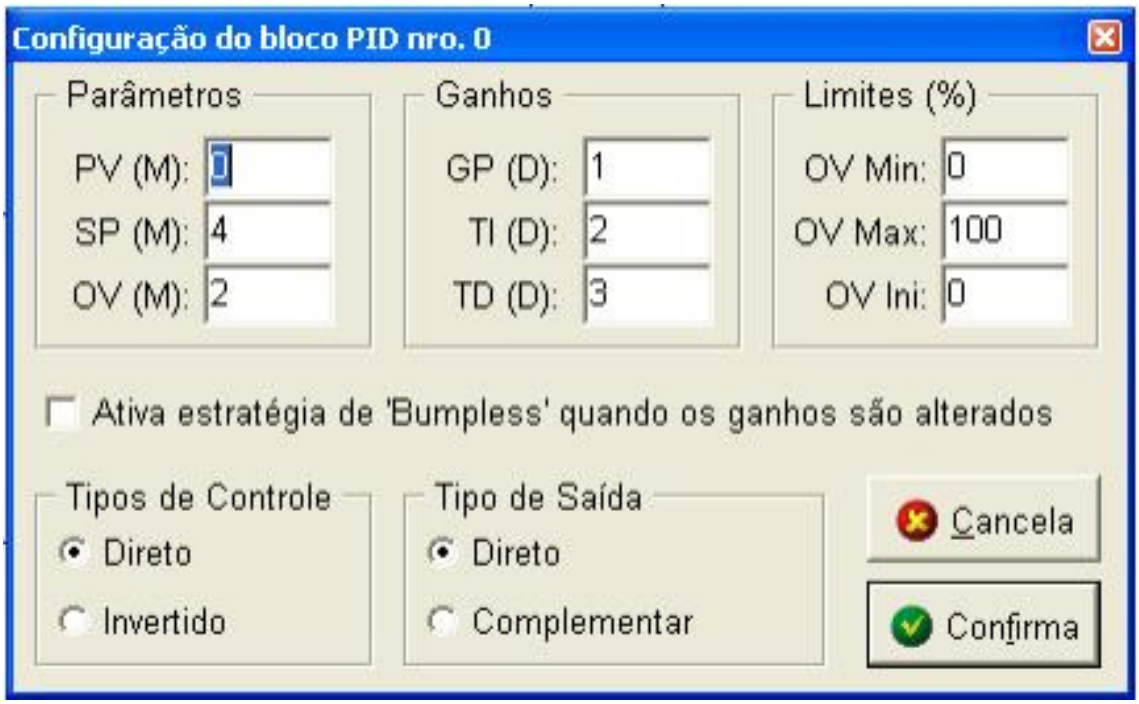

Figura 5.11: Janela de configuração do controlador PID no CLP ZAP 900.

modificados, permanecendo com os valores padrão.

5.2.4 Configuração do Servidor OPC

Após carregar o programa Ladder no CLP, deve-se realizar a configuração e habilitação do servidor OPC, de modo que outras aplicações tenham acesso às variáveis do CLP.

A HI Tecnologia disponibiliza um *software* com tal propósito, chamado de *HS1 Power Tool*. A figura 5.12 ilustra a tela do *software* em questão. Nessa janela são definidas as informações a respeito do canal de comunicação, tais como a porta utilizada, o modo de comunicação, a velocidade da transferência, o número de bits, os bits de parada e a paridade.

Depois de configurado e ativado o canal de comunicação, um dispositivo padrão deve ser criado e também ativado. A partir daí, adicionam-se blocos de dados, objetos que representam as *tags* do processo no servidor OPC. A figura 5.13 mostra a janela de configuração dos blocos de dados. Nessa janela é possível definir nome, descrição, endereço, tipo e taxa de atualização do bloco.

Após configurado o servidor, este deve ser habilitado através do botão *Play*. Desta maneira, qualquer cliente OPC terá acesso aos dados do servidor, estando habilitado a ler e escrever nas variáveis do processo.

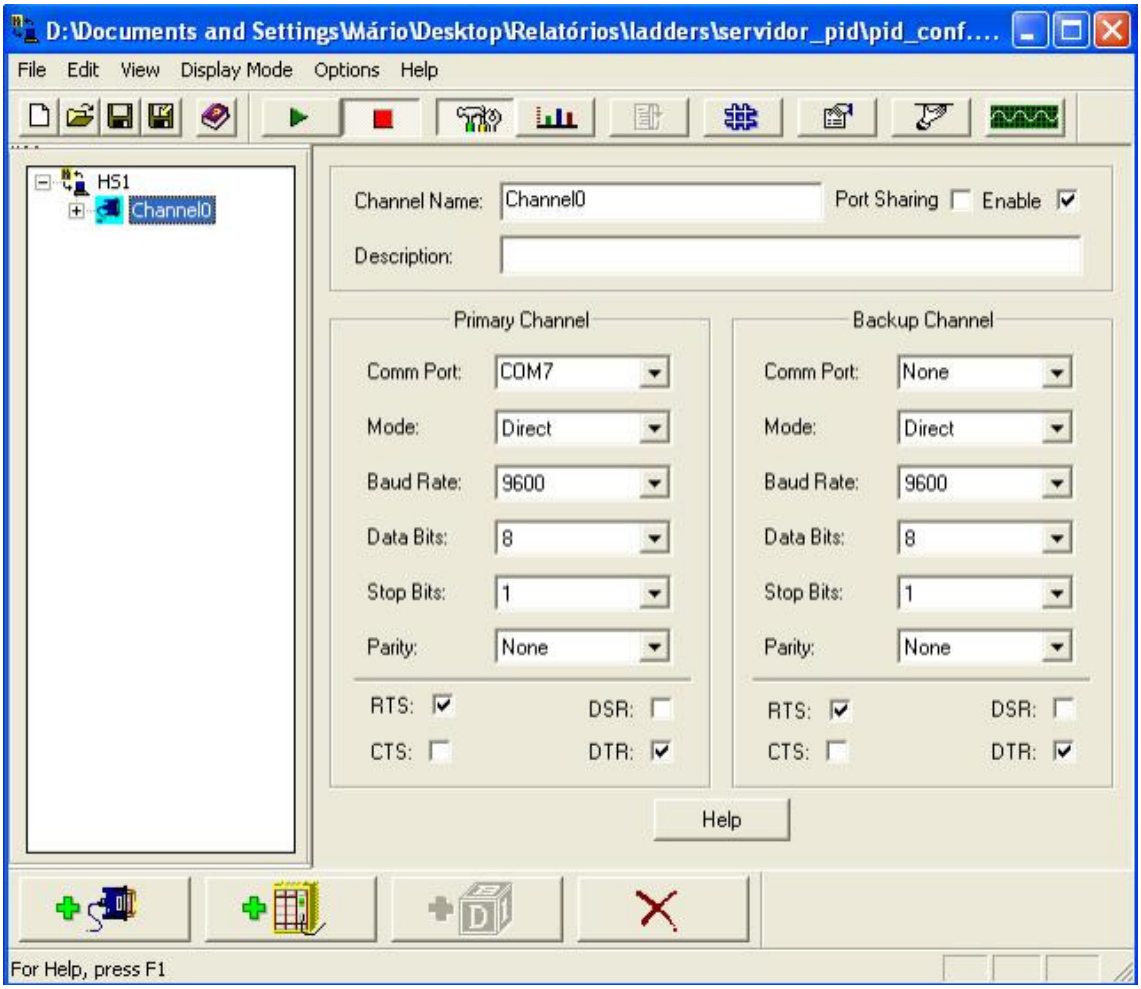

Figura 5.12: Janela de configuração do servidor OPC.

5.2.5 Implementação das Estratégias Controle

Após configurados os elementos de hardware e *software* para implementação do controlador PID no CLP, devemos focar na implementação das três estratégias de controle citadas anteriormente. Cada uma delas é descrita a seguir.

Controle PID

A primeira estratégia de controle para o sistema de tanques implementada é a do controle PID. Apesar de esta solução não estar diretamente relacionada ao foco do trabalho, a coleta dos dados desse experimento será importante para comparação posterior com as outras soluções.

A ação de controle utilizada foi a Proporcional-Integrativa (PI), desta maneira, eliminando a parte derivativa da ação ($TD = 0$). Esta opção foi escolhida com o objetivo de minimizar o erro de regime do sistema de tanques.

Nessa tarefa inicial, a constante de tempo integrativo TI foi fixada em $10s$, com o

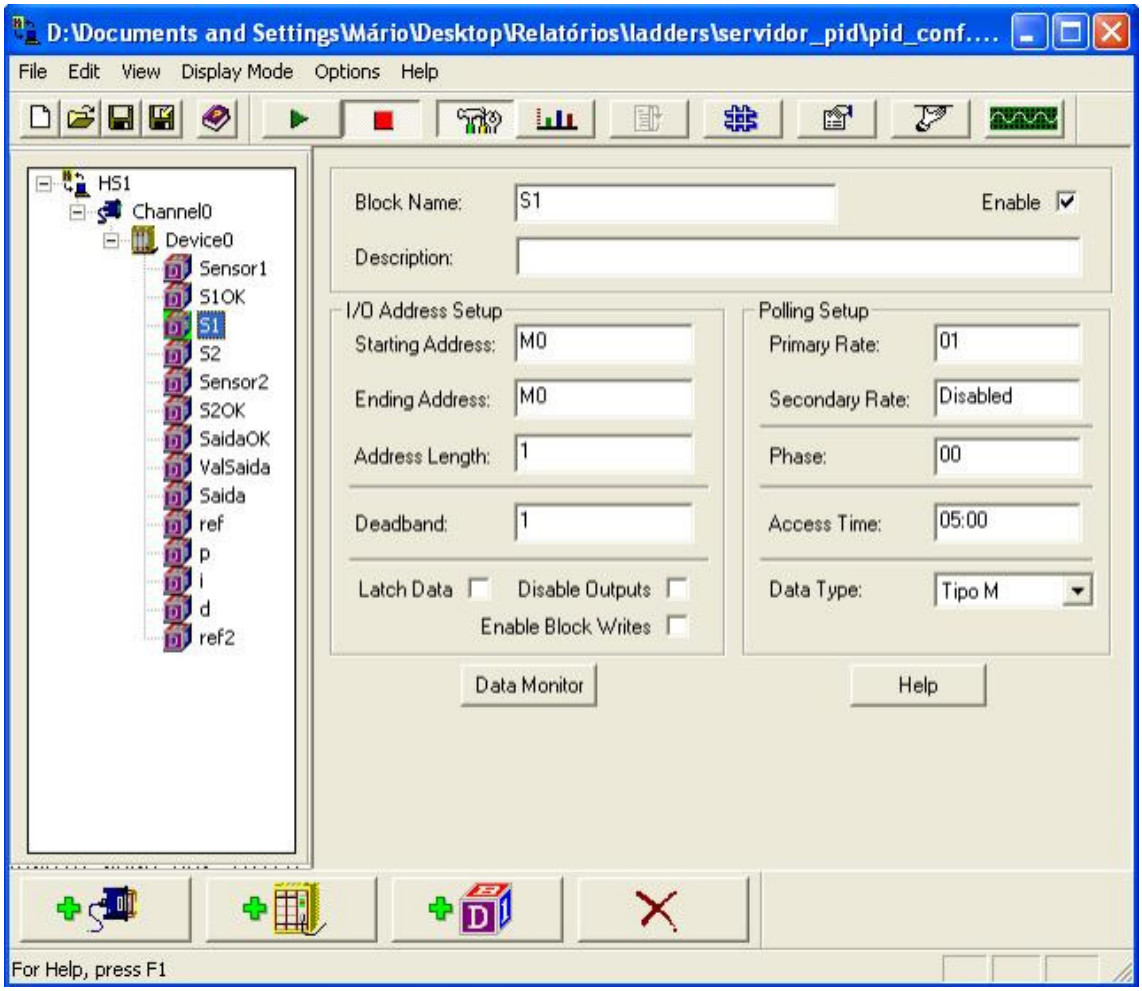

Figura 5.13: Janela de configuração dos blocos de dados no servidor OPC.

ganho proporcional GP igual a 10 e 2, sucessivamente. A figura 5.14 ilustra o gráfico de tendência do nível do segundo tanque para $GP = 2$. A figura 5.15 apresenta o gráfico de tendência do nível para $GP = 10$. Vale salientar que este trabalho não visa a busca e implementação de estratégias de controle ótimas. O principal objetivo desta seção é a validação da ferramenta JFuzZ, como uma alternativa para construção de controladores baseados em lógica *fuzzy*.

Observando os gráficos notamos que um maior ganho proporcional leva a um maior valor de *overshoot*. Por sua vez, o controle com ganho proporcional igual a 10 apresentou um menor tempo de subida que o controle com ganho proporcional igual a 2.

Além dos gráficos de nível, foram gerados gráficos que apresentam o comportamento dos sinais de controle e dos módulos dos erros. O sinal de controle nessa tarefa é a tensão aplicada ao módulo de potência, que pode variar entre $0.6V$ e $3V$, sendo multiplicado em 5 vezes na saída do módulo para alimentação da bomba. Tal *range* de valores de tensão se dá devido ao fato de que a bomba deve ser alimentada uma tensão máxima de $15V$ (o que explica o limite máximo de $3V$ na entrada do módulo de potência), bem como pelo

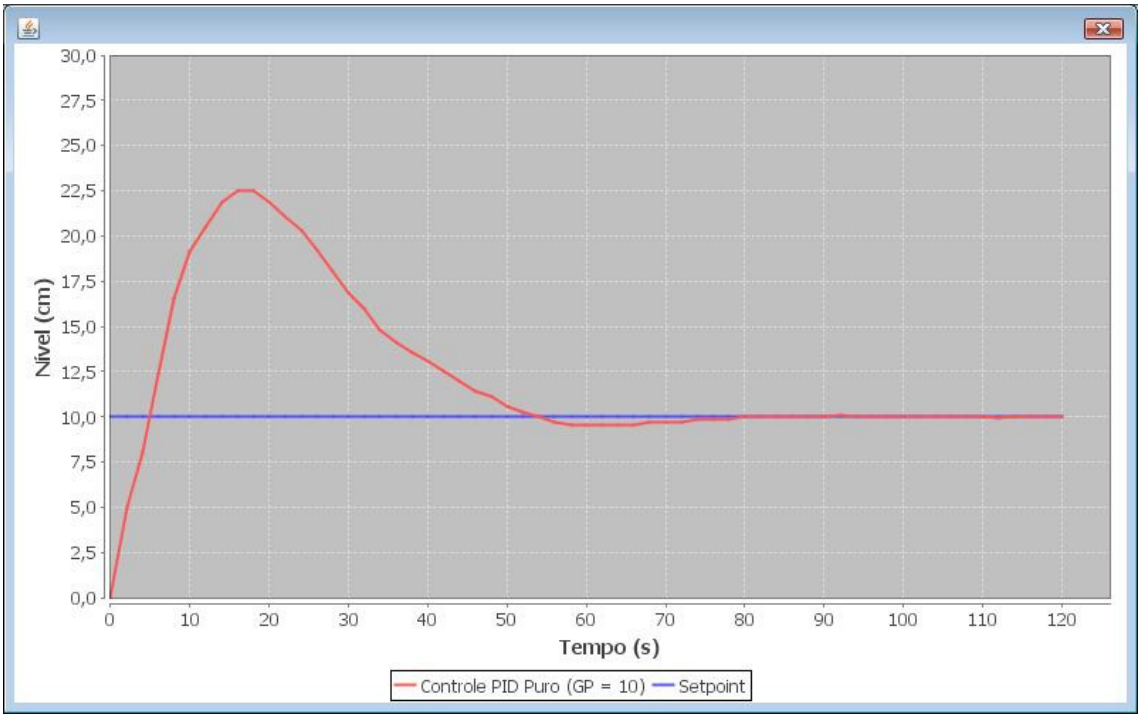

Figura 5.14: Nível do tanque para GP = 10.

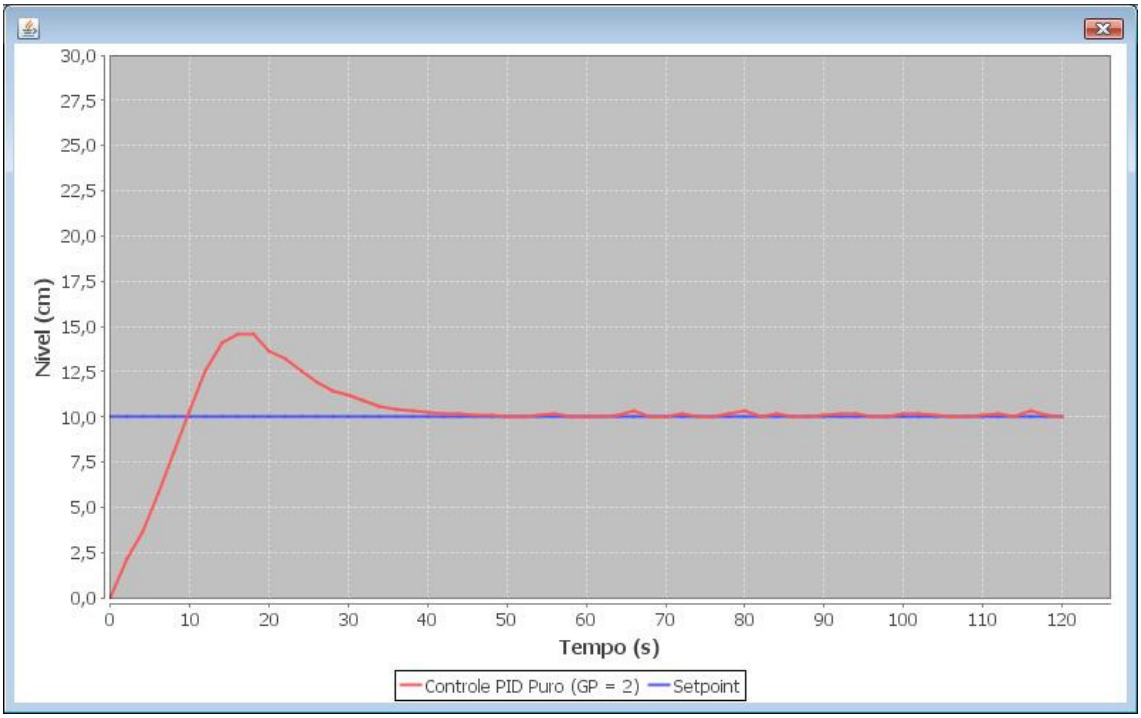

Figura 5.15: Nível do tanque para GP = 2.

fato de a saída analógica do CLP utilizado não ser capaz de gerar uma corrente inferior a

$4mA$ (pela Lei de Ohm, para tal corrente e um resistor de 150Ω, a tensão resultante é de $0.6V$). A figura 5.16 apresenta o gráfico de tendência do sinal de controle do controlador PID com $GP = 10$ e a figura 5.17 o gráfico para $GP = 2$. A figura 5.18 ilustra o módulo do erro (diferença entre o valor da variável do processo e a referência) para o controlador PID com $GP = 10$ e a figura 5.19 o gráfico para $GP = 2$.

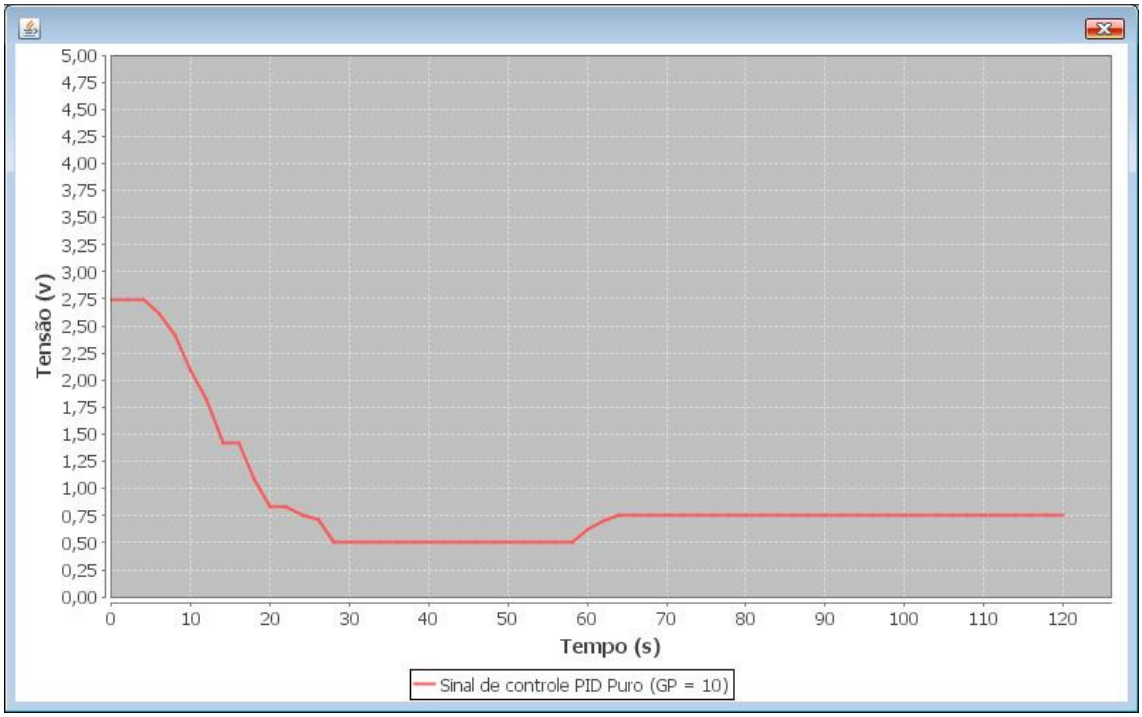

Figura 5.16: Sinal de controle para GP = 10.

Controle *Fuzzy* Direto

Como primeira etapa para utilização do JFuzZ em uma tarefa de controle, utilizou-se a idéia de controle *fuzzy* direto. Nessa aplicação, um controlador *fuzzy* simples tem como variáveis de entrada o erro atual do sistema e a variação do erro (diferença entre o erro atual e o erro do ciclo anterior). A partir de tais valores, o controlador gera diretamente um valor para o sinal de controle do sistema, nesse caso, a tensão aplicada à entrada do módulo de potência.

Para o controle *fuzzy* direto, as funções de pertinência das variáveis de entrada foram modeladas empiricamente, e são mostradas a seguir, sendo: a figura 5.20 ilustra as funções de pertinência para a variável de entrada *Erro*, a figura 5.21 mostra as funções de pertinência para a variável de entrada *Variação do Erro* e a figura 5.22 apresenta o gráfico de funções de pertinência para a variável de saída *Tensão*.

Além das funções de pertinência mostradas, o controlador foi construído utilizando as seguintes regras de inferência:

1. Se Erro é **NEGATIVO ALTO** e Variação do Erro é **ZERO** então Tensão é **ALTA**

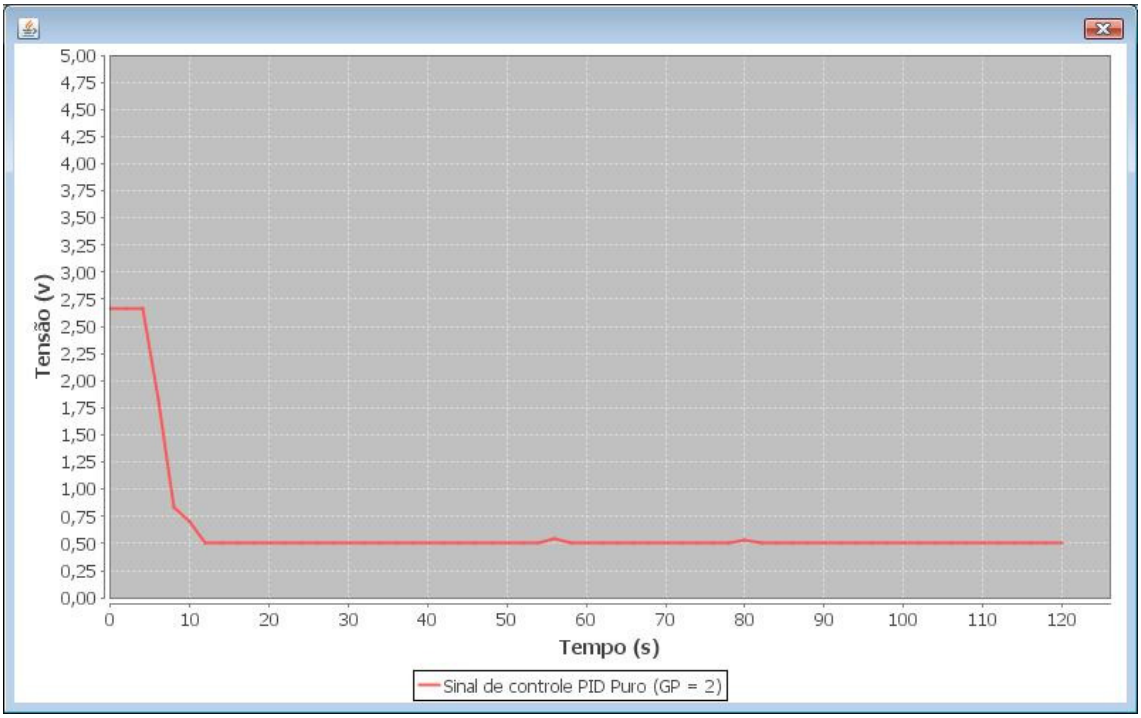

Figura 5.17: Sinal de controle para GP = 2.

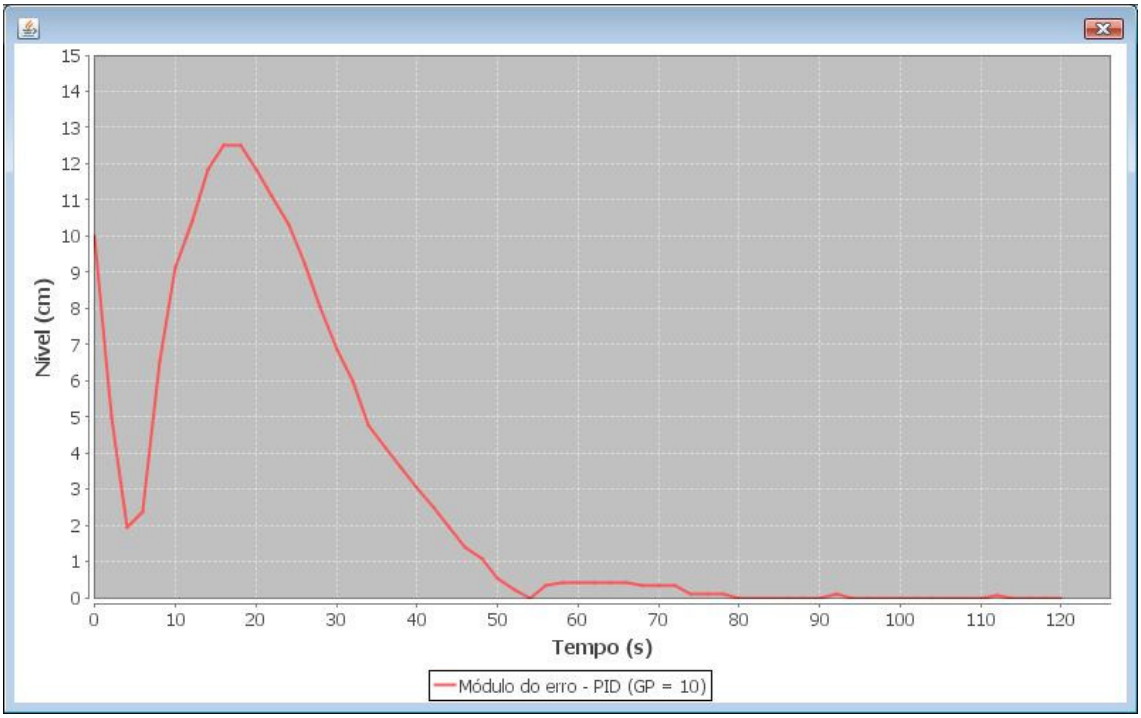

Figura 5.18: Módulo do erro para GP = 10.

2. Se Erro é **NEGATIVO ALTO** e Variação do Erro é **POSITIVO** então Tensão é

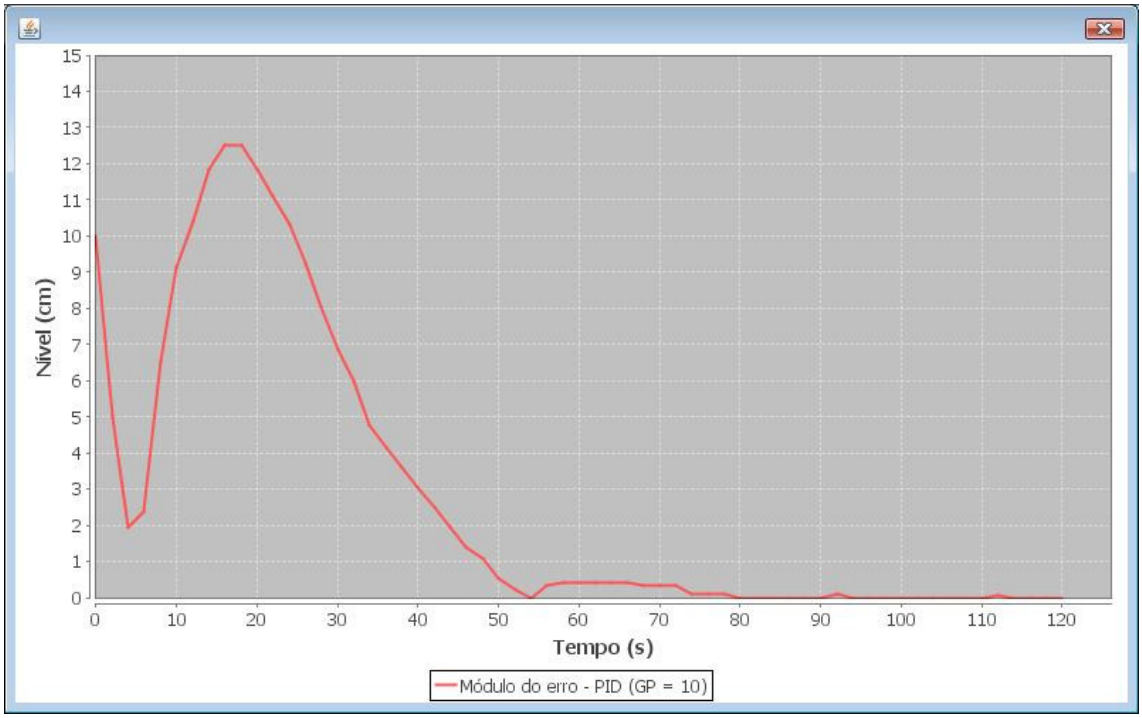

Figura 5.19: Módulo do erro para GP = 2.

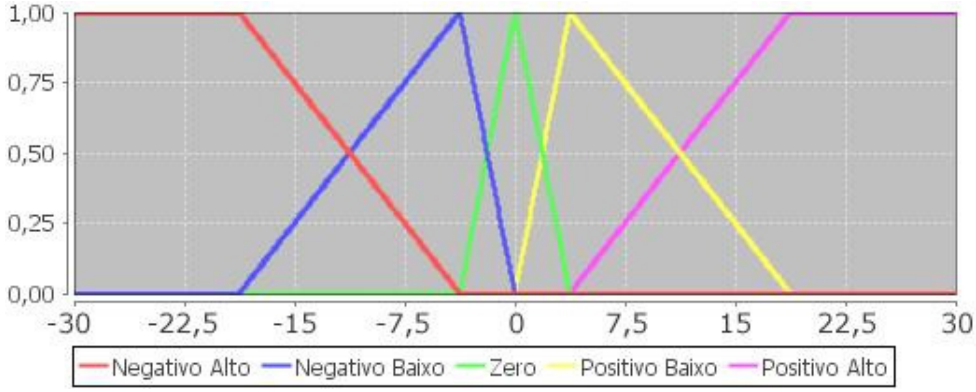

Figura 5.20: Funções de pertinência da variável *Erro*.

BAIXA

3. Se Erro é **NEGATIVO ALTO** e Variação do Erro é **NEGATIVO** então Tensão é **ALTA**
4. Se Erro é **NEGATIVO BAIXO** e Variação do Erro é **NEGATIVO** então Tensão é **BAIXA**
5. Se Erro é **NEGATIVO BAIXO** e Variação do Erro é **ZERO** então Tensão é **BAIXA**
6. Se Erro é **NEGATIVO BAIXO** e Variação do Erro é **POSITIVO** então Tensão é **ZERO**
7. Se Erro é **ZERO** e Variação do Erro é **NEGATIVO** então Tensão é **BAIXA**
8. Se Erro é **ZERO** e Variação do Erro é **ZERO** então Tensão é **ZERO**

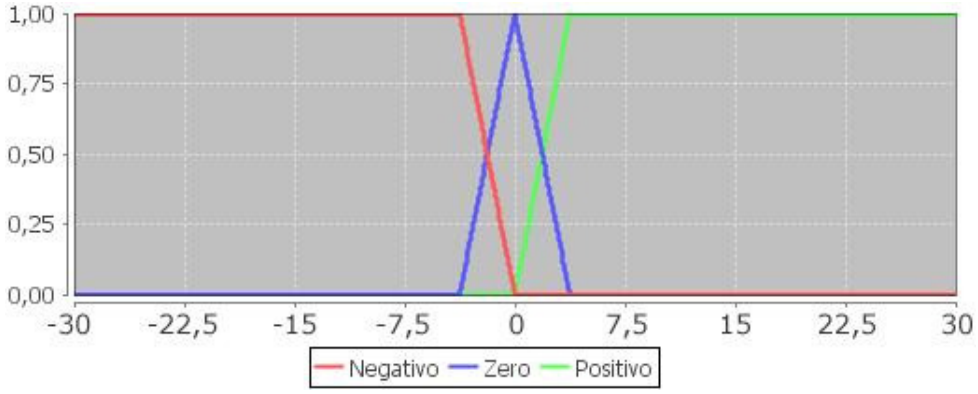

Figura 5.21: Funções de pertinência da variável *Variação do Erro*.

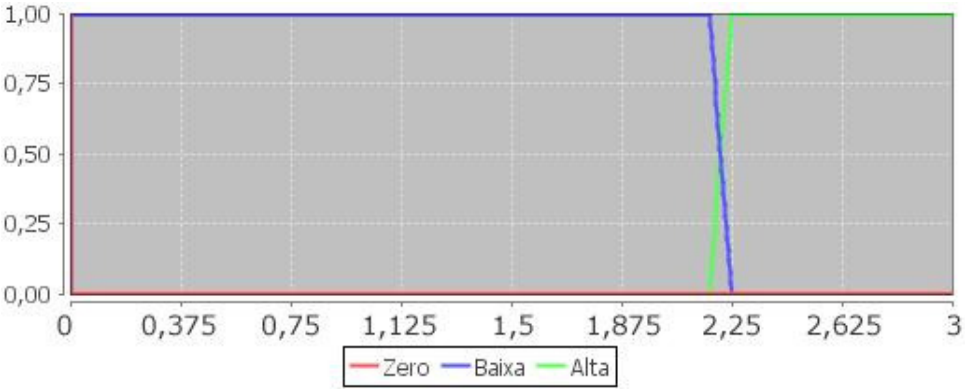

Figura 5.22: Funções de pertinência da variável *Tensão*.

9. Se Erro é **ZERO** e Variação do Erro é **POSITIVO** então Tensão é **ZERO**
10. Se Erro é **POSITIVO BAIXO** e Variação do Erro é **NEGATIVO** então Tensão é **ZERO**
11. Se Erro é **POSITIVO BAIXO** e Variação do Erro é **POSITIVO** então Tensão é **ZERO**
12. Se Erro é **POSITIVO BAIXO** e Variação do Erro é **ZERO** então Tensão é **ZERO**
13. Se Erro é **POSITIVO ALTO** e Variação do Erro é **ZERO** então Tensão é **ZERO**
14. Se Erro é **POSITIVO ALTO** e Variação do Erro é **POSITIVO** então Tensão é **ZERO**
15. Se Erro é **POSITIVO ALTO** e Variação do Erro é **NEGATIVO** então Tensão é **ZERO**

As configurações básicas do controlador são mostradas a seguir.

- **Método de Conjunção**: *min*;
- **Método de Disjunção**: *max*;
- **Método de Implicação**: *min*;
- **Método de Agregação**: *max*;
- **Método de Defuzzificação**: *centroid*.

Após construído o controlador, a tarefa de controle *fuzzy* direto foi realizada e os gráficos foram gerados. A figura 5.23 ilustra a tendência do nível do tanque para a aplicação. A figura 5.24 apresenta o gráfico do sinal de controle para este controle. Por fim, a figura 5.25 mostra o módulo do erro durante a execução da aplicação.

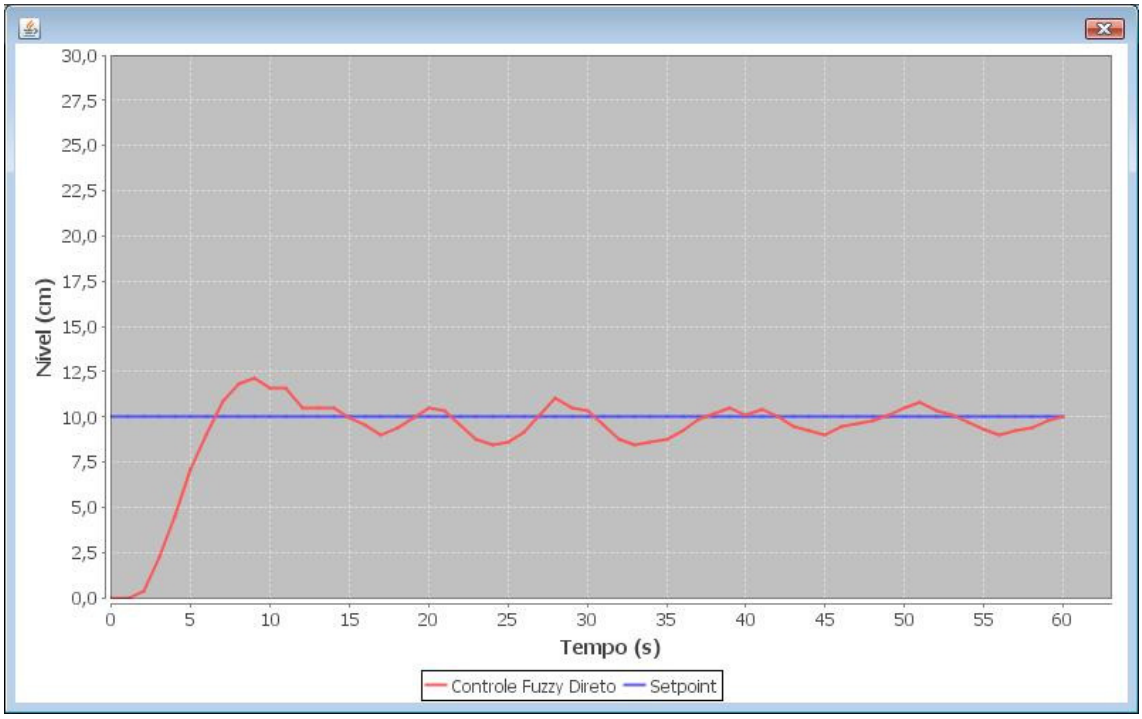

Figura 5.23: Nível do tanque para a estratégia de controle *fuzzy* direto.

Controle Adaptativo

Controle adaptativo é a denominação atribuída à ação de controle de sistemas capazes de modificar seus próprios parâmetros em resposta a alterações verificadas em algum módulo que esteja sendo monitorado. Nesse sentido, adaptar-se significa mudar o comportamento em resposta a novas circunstâncias de operação, com o objetivo de manter um nível esperado de desempenho.

Nesta experiência, o papel do mecanismo de adaptação é executado pelo controlador *fuzzy*, que por sua vez é responsável pela sintonização dos parâmetros do controlador PID criado anteriormente.

Observando os resultados do controle PID puro, é notório que cada um dos valores da constante de ganho proporcional utilizados melhora em algum aspecto o desempenho do controle. Sendo assim, a tarefa do controlador *fuzzy* é alterar o valor de *GP* de acordo com o valor do erro atual do sistema. Essa estratégia pode tornar-se mais eficiente através da construção de um mecanismo de adaptação mais robusto, porém, a nível de validação da ferramenta JFuzZ, decidiu-se pela criação de um controlador simples que sintoniza apenas o parâmetro *GP*.

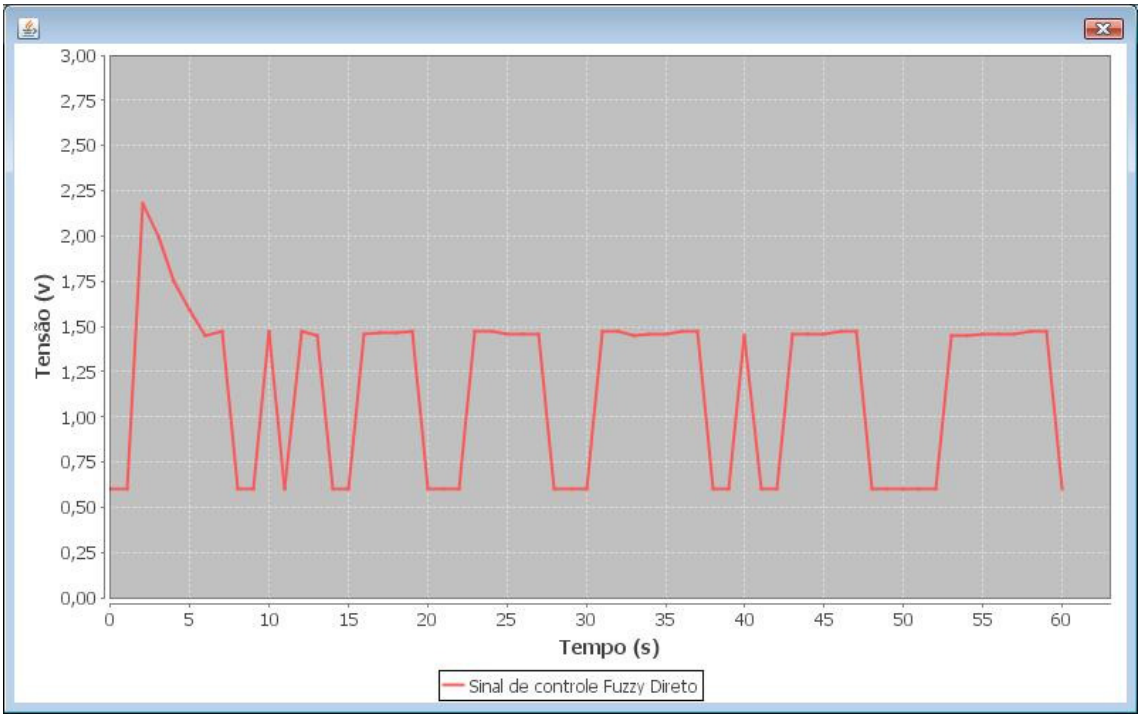

Figura 5.24: Tensão aplicada ao módulo de potência no controle *fuzzy* direto.

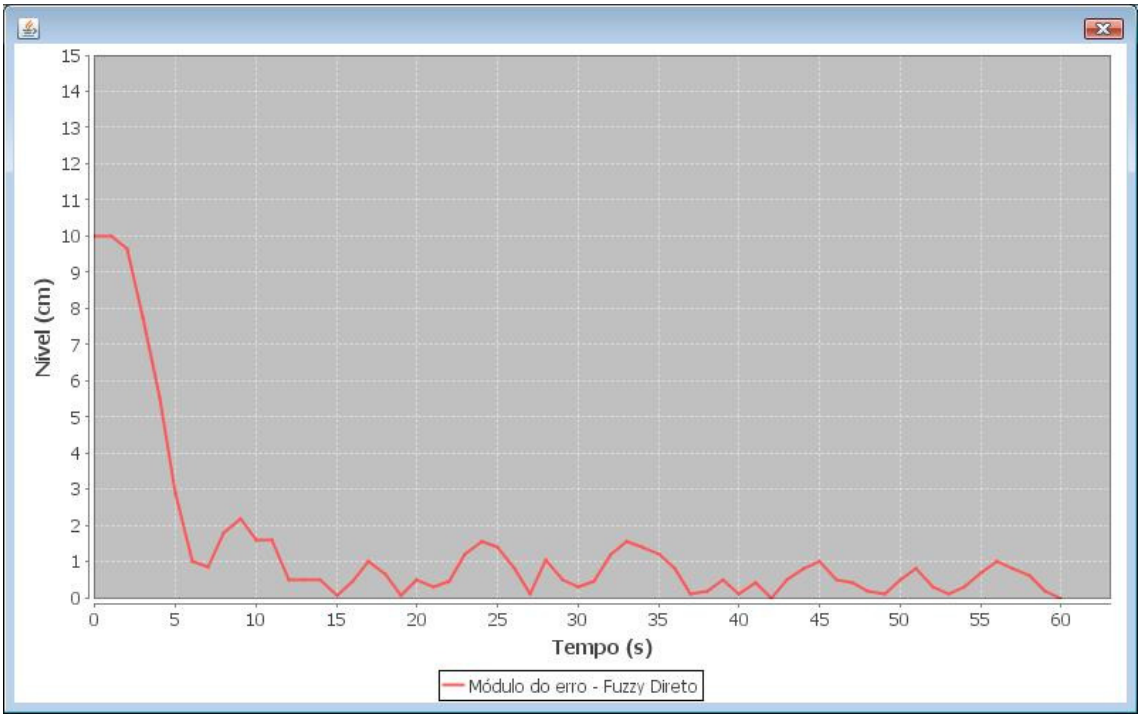

Figura 5.25: Módulo do erro no controle *fuzzy* direto.

De acordo com os resultados anteriores, verificamos que para $GP = 10$ o sistema

apresenta um maior *overshoot* e um menor tempo de subida que o controle com $GP = 2$. Dessa forma, o controlador *fuzzy* deve variar de forma que, para um erro maior (mais distante do *setpoint*), maior será o ganho proporcional (mais rápido ele se aproximará da referência). A figura 5.26 ilustra as funções de pertinência para a variável de entrada *Erro* e a figura 5.27 apresenta o gráfico de funções de pertinência para a variável de saída *GP*.

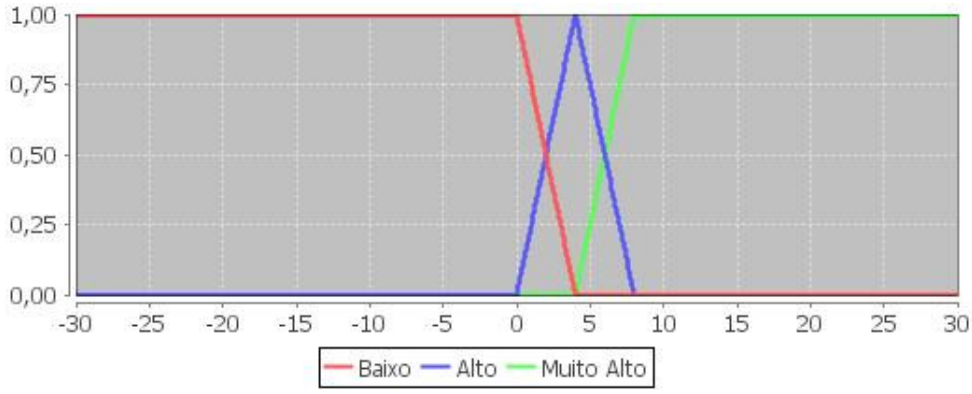

Figura 5.26: Funções de pertinência para a variável *Erro* do controlador adaptativo.

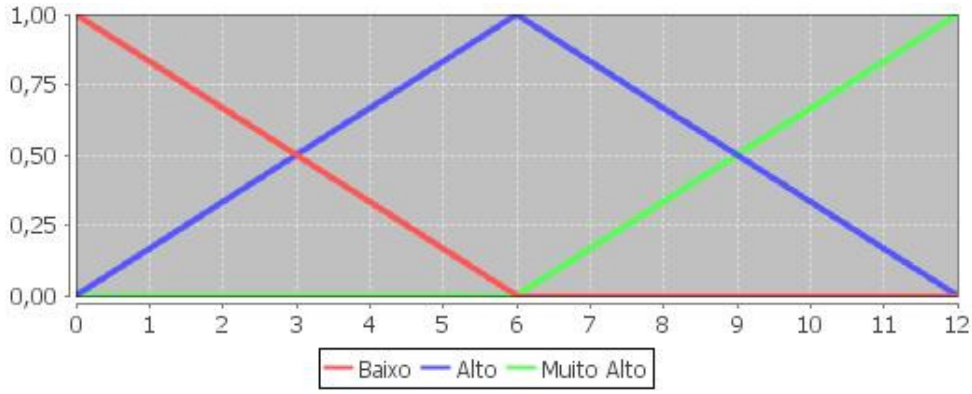

Figura 5.27: Funções de pertinência para a variável *GP* do controlador adaptativo.

O conjunto de regras de inferência para o controlador é mostrado a seguir, e as configurações básicas do controlador construído permanecem como no exemplo anterior.

1. Se Erro é **BAIXO** então GP é **BAIXO**
2. Se Erro é **ALTO** então GP é **ALTO**
3. Se Erro é **MUITO ALTO** então GP é **MUITO ALTO**

Após a realização do controle do sistema, os dados foram coletados para geração dos gráficos. A figura 5.28 ilustra a tendência do nível do tanque para a aplicação. A figura 5.29 apresenta o gráfico do sinal de controle para este controle. Por fim, a figura 5.30 mostra o módulo do erro durante a execução da aplicação.

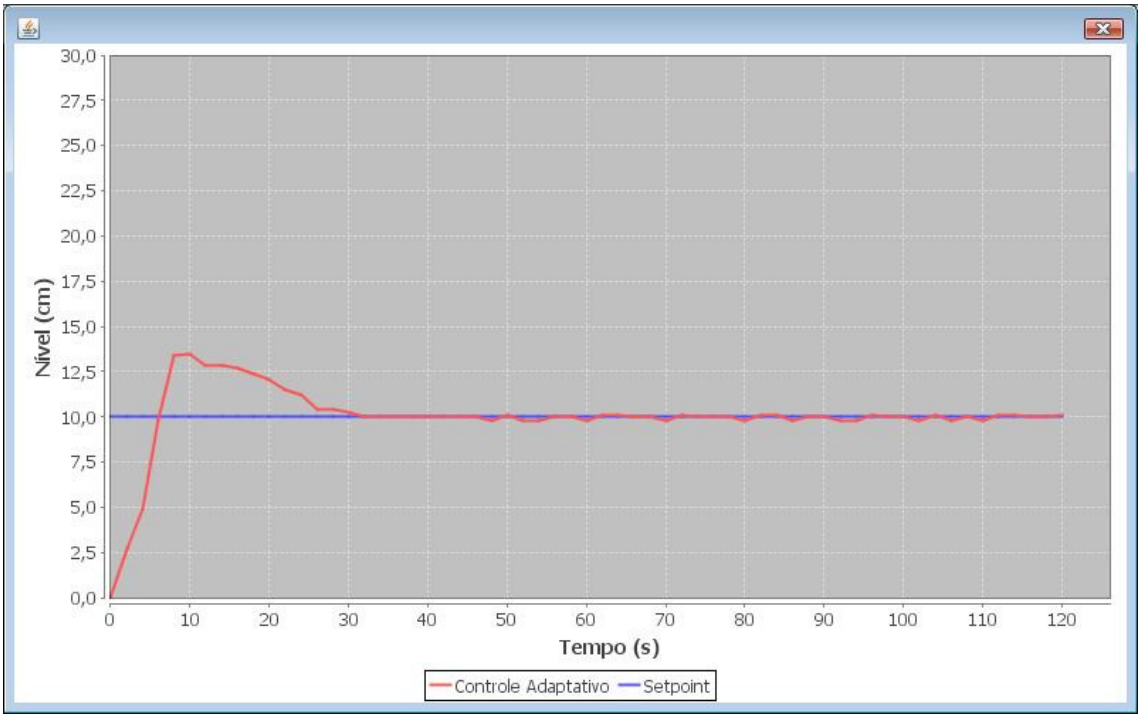

Figura 5.28: Nível do tanque para a estratégia de controle PID adaptativo.

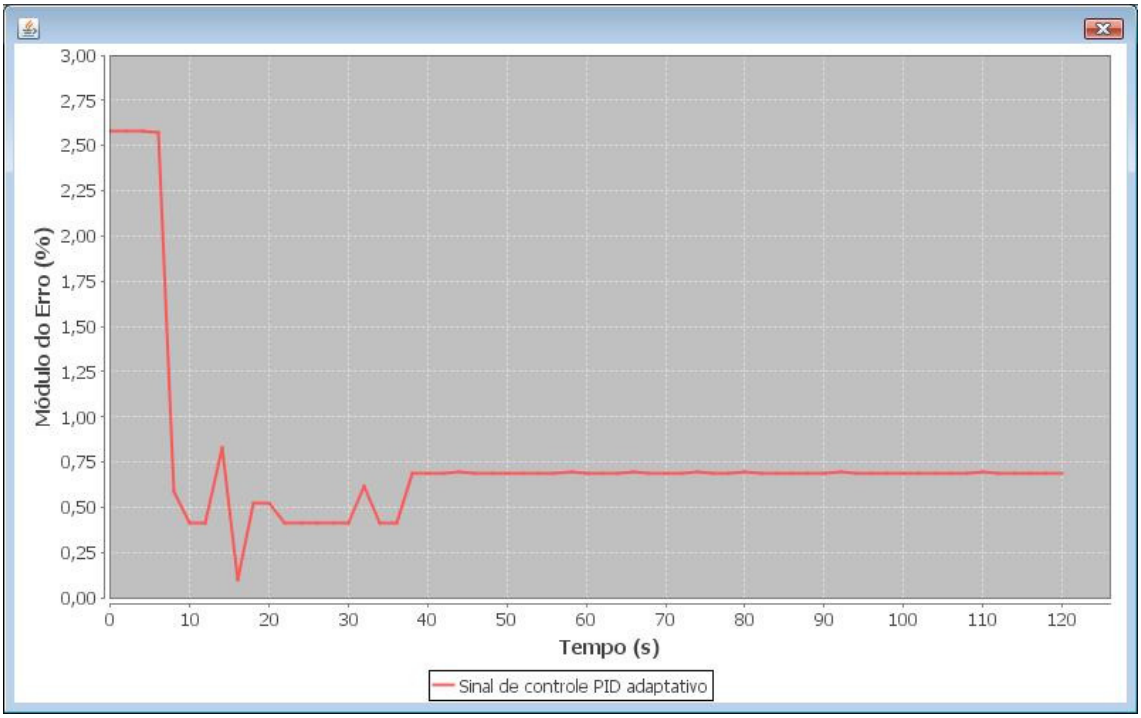

Figura 5.29: Tensão aplicada ao módulo de potência no controle PID adaptativo.

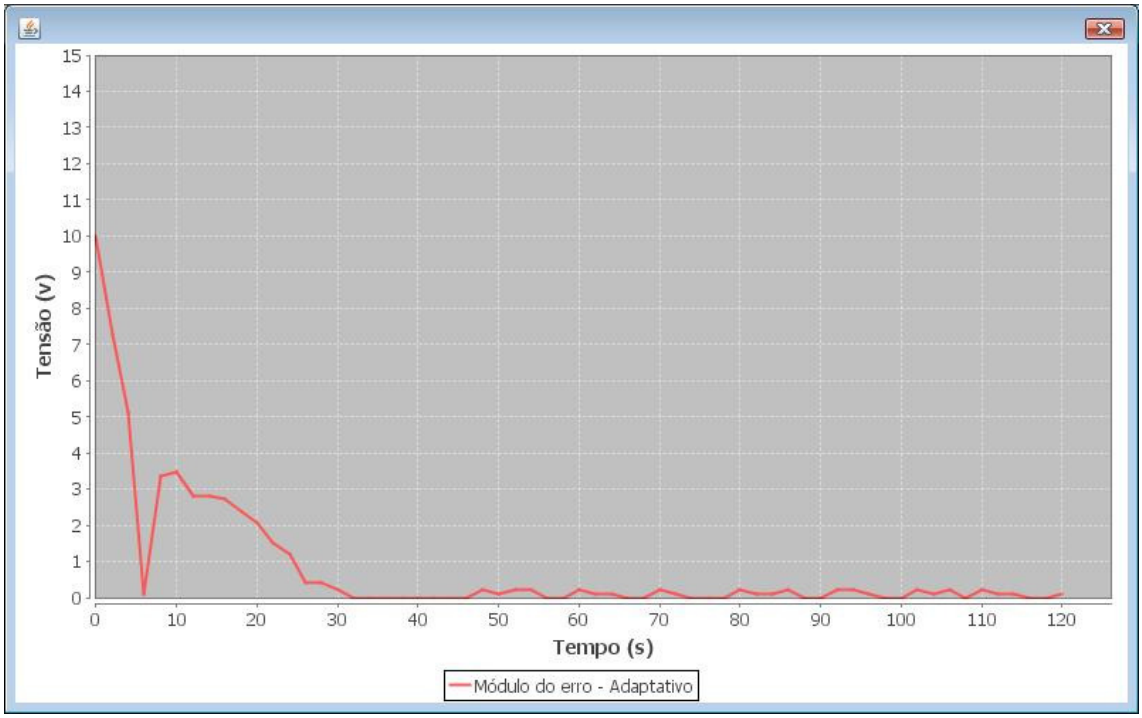

Figura 5.30: Módulo do erro no controle PID adaptativo.

Tabela 5.1: Características das estratégias utilizadas no controle do módulo de tanques

Controlador	***Overshoot* (%)**	**TE a 2% (s)**	**TS (s)**
PID com $GP = 10$	12.5	74.0	5.0
PID com $GP = 2$	4.63	41.8	9.9
Fuzzy Direto	2.17	∞	6.5
Adaptativo	3.48	30.2	6.0

5.2.6 Resultados Obtidos

É possível perceber graficamente as características de cada aplicação de controle implementada. Os gráficos da estratégia de controle PID puro mostraram que um ganho proporcional alto resulta em um bom tempo de subida, porém um *overshoot* muito alto e, consequentemente, um tempo de estabilização também muito elevado. De forma análoga, a redução do ganho proporcional do controlador implica em maior tempo de subida.

A estratégia de controle *fuzzy* direto deixa a desejar, principalmente pelo fato de o sistema não conseguir estabilizar-se. Este problema pode ser reduzido através da construção de controladores mais robustos, porém essa estratégia, por natureza, torna o sistema demasiadamente oscilatório.

Das soluções apresentadas, o controle adaptativo apresentou-se como a mais satisfatória em termos de *overshoot*, tempo de subida (TS) e tempo de estabilização (TE), superando as outras estratégias nos três atributos. A tabela 5.1 apresenta os resultados detalhados dos testes.

Capítulo 6

Considerações Finais

No decorrer deste documento, foram apresentados os detalhes de desenvolvimento da ferramenta JFuzZ, um ambiente para criação de aplicações *fuzzy* industriais. Também foi realizado um estudo sobre a lógica *fuzzy* e as principais estratégias de controle de processos utilizando a lógica *fuzzy*.

Viu-se que a teoria *fuzzy* é uma ferramenta importante na implementação de estruturas de controle inteligente, contribuindo para o alcance de melhores resultados nos sistemas de controle regulatório. Neste caso, a ferramenta JFuzZ mostrou-se como uma alternativa amigável e de fácil operação na construção de sistemas de inferência *fuzzy*, em geral.

Além disso, o trabalho destacou a importância do protocolo de comunicação OPC, atualmente um dos padrões para troca de informações entre dispositivos industriais, bem como a interface disponível no JFuzZ com este propósito.

Foram implementadas estratégias de validação para o sistema JFuzZ, de modo a determinar se a ferramenta apresenta níveis satisfatórios de estabilidade, confiabilidade e robustez de *software*, bem como determinar se o sistema é adequado ao meio industrial.

Em primeira instância, a validação foi realizada de maneira estática, através da construção de sistemas de inferência *fuzzy* idênticos em ambientes diferentes. Verificou-se que o mesmo conjunto de entradas implicava no mesmo conjunto de saídas para o JFuzZ e o *Matlab Fuzzy Logic Toolbox*, ferramenta amplamente utilizada e, consequentemente, validada.

Em segundo momento, foram aplicadas técnicas de controle a fim de regular o nível da água no módulo de tanques da Quanser. Tais estratégias foram implementadas com o auxílio total ou parcial do JFuzZ. Verificou-se que o controle adaptativo, através da sintonização dos parâmetros do controlador PID a partir de um controlador *fuzzy*, surtiu melhores efeitos na planta, diminuindo o tempo de subida, o tempo tempo de estabilização e o *overshoot*.

Desta maneira, o JFuzZ atendeu de forma eficiente às expectativas quanto à exatidão das saídas dos sistemas de inferência criados, bem como à interação com processos de controle industrial reais. Dados os resultados, verificamos que o JFuzZ é uma ferramenta precisa, e melhor aplicável ao ambiente industrial que as outras soluções de *software* analisadas neste trabalho.

Os resultados obtidos neste trabalho servem de motivação para a expansão da ferramenta. Existe atualmente a necessidade de adicionar novos módulos e funcionalidades ao

JFuzZ, tais como:

- **Módulo de pós-processamento:** de forma análoga ao módulo de pré-processamento apresentado, é necessário o desenvolvimento de um módulo capaz de manipular as variáveis de saída do sistema, de modo a possibilitar a realização de operações nos valores gerados pelo controlador *fuzzy* (Ex.: aplicação de ganhos).
- **Manipulação gráfica das funções de pertinência:** possibilidade de edição das funções de pertinência através de cliques no mouse.
- **Inclusão de outras interfaces de comunicação:** expansão da lista de interfaces industriais disponíveis no sistema. A próxima interface em vista é a de comunicação com o PI (*Plant Information*).
- **Documentação:** desenvolvimento da documentação completa do *software*, tais como manuais, diagramas de sequencia, diagramas de colaboração etc.

Além da continuação do desenvolvimento do *software* existente, os resultados motivam a criação e expansão de outras ferramentas, tentando atingir novas abordagens além do controle de processos, onde a lógica *fuzzy* vem a otimizar determinadas aplicações do meio industrial. O próximo passo será focado na aplicação do JFuzZ e da lógica *fuzzy* na detecção de falhas incipientes em equipamentos industriais, como proposta de tese de doutoramento.

Referências Bibliográficas

ANDRADE, ALEXANDRE A. DE (2001), Emprego de sistemas especialistas em sistemas supervisórios de automação industrial, Dissertação de mestrado, Universidade de São Paulo, USP, São Paulo, SP.

CARULO, MARCELLE F. (2008), Desenvolvimento e Implementação de Controlodor Fuzzy Aplicado à Produção de Biodiesel, Tese de doutorado, Universidade Estadual de Campinas, Campinas, SP.

CHUAN, FENG, SUN ZENGGI & SU LING (2004), 'Design and implementation of scilab fuzzy logic tooibox', *IEEE Intemational Symposium on Computer Aided Control Systems Design* .

COSTA, BRUNO S. JALES (2008), 'Desenvolvimento de componentes visuais para monitoração de processos na indústria de petróleo de gás', Universidade Federal do Rio Grande do Norte. Monografia.

COSTA, WENDEL T., ERIVELTON G. NEPOMUCENO & ORIANE M. NETO (n.d.), 'Controle nebuloso de sistemas não-lineares via método takagi-sugeno: Uma abordagem didática usando simulink', Universidade Federal de São João Del-Rei - MG.

COUGHANOWR, DONALD R. (1991), *Process Systems Analysis and Control*, McGraw-Hill, Drexel, EUA.

FANG, XINLI, TAO SHEN, XIAOHONG WANG & ZHIQUN ZHOU (n.d.), 'Application and research of fuzzy pid in tank systems', *Fourth International Conference on Natural Computation* .

FERNANDES, FRANCISCO G., JOSÉ S. B. LOPES, ANDRE L. MAITELLI, FÁBIO M. U. DE ARAÚJO & LUIZ AFFONSO H. G. DE OLIVEIRA (n.d.), 'Implementação de controladores pid utilizando lógica fuzzy e instrumentação industrial', *Simpósio Brasileiro de Automação Inteligente* .

FLORESCU, ADRIANA, VASILE ALEXANDRU & DAN A. STOICHESCU (2001), 'Fuzzy control of ac-dc converters supplying high precision dc drive systems', *24th International Spring Seminar on Electronics Technology* .

GHARIEB, W. & G. NAGIB (n.d.), 'Fuzzy intervention in pid controller desing', *IEEE ISIE* .

HALL, LAWRENCE O. & RICHARD J. HATHAWAY (1996), 'Fuzzy logic toolbox: Software review', *IEEE Transactions on Fuzzy Systems* **4**(1), 82–85.

JONES, CLARENCE T. (1996), *Programmable Logic Controllers - The Complete Guide to the Technology*, Patrick-Turner Publishing Company, United States.

KAZUO TANAKA, HUA O. WANG (2001), *Fuzzy Control Systems Design and Analysis: A Linear Matrix Inequality Approach*, Wiley-Interscience.

LIU, JUN, KHIANG W. LIM, WENG K. HO, KAY C. TAN, ARTHUR TAY & RAJAGOPALAN SRINIVASAN (2005), 'Using the opc standard for real-time process monitoring and control', *IEEE Software* **22**(6), 54–59.

LOPES, JOSÉ S. B., FÁBIO M. U. DE ARAÚJO, ANDRÉ L. MAITELLI & OSCAR G. FILHO (n.d.), 'Controle avançado de processos industriais de petróleo utilizando um sistema de inferência neuro-fuzzy para geração dinâmica de set-points', Universidade Federal do Rio Grande do Norte.

MAMDANI, EBRAHIM H. & S. ASSILIAN (1975), 'An experiment in linguistic synthesis with a fuzzy logic controller', *International Journal of Man-Machine Studies* **7**, 1–13.

MOZELLI, LEONARDO A. (2008), Controle fuzzy para sistema takagi-sugeno: Condições aprimoradas e aplicações, Dissertação de mestrado, Universidade Federal de Minas Gerais, UFMG, Belo Horizonte, MG.

NASCIMENTO., CAIRO L. & TAKASHI YONEYAMA (1999), *Inteligência Artificial em Controle e Automação*, Edgard Blücher, São José dos Campos, SP.

PAWELCZYK, MAREK (n.d.), 'Integrating different control modes in a fuzzy controller', Institute of Automatic Control, Silesian Technical University, POLAND.

PEDRYCZ, WITOLD (1993), *Fuzzy Control and Fuzzy Systems*, Research Studies Press, Taunton, England.

PEDRYCZ, WITOLD & FERNANDO A. C. GOMIDE (1998), *An Introduction to Fuzzy Sets: Analysis and Design (Complex Adaptive Systems)*, MIT Press, Boston.

QUANSER (2004), *Coupled tanks user manual*.

SANDRI, SANDRA & CLÁUDIO CORREA (1999), *Lógica Nebulosa*, V Escola de Redes Neurais, Conselho Nacional de Redes Neurais, São José dos Campos, SP.

SANKAR, G. & S. S. KUMAR (n.d.), 'Fuzzy logic based automatic braking system in trains', *Proceedings of India International Conference on Power Electronics* .

SANTOS, ALMIR B. DOS, LUIZ OCTÁVIO & ABRÃO A. JUNIOR (2006), 'A influência da automação industrial na distribuição, logística e transportes de derivados no brasil', *Rio Oil and Gas Expo and Conference* .

SOMESSARI, SAMIR L., ANSELMO FEHER, FRANCISCO E. SPRENGER, MARIA E. C. M. ROSTELLATO, FABIO E. COSTA & WILSON A. P. CALVO (2009), 'Development of an automation system for iodine-125 brachyterapy seed production by nd: Yag laser welding', *International Nuclear Atlantic Conference - INAC, Rio de Janeiro* .

SOUZA, ALESSANDRO (2005), Sistema de gerência de informação de processos industriais via web, Dissertação de mestrado, Universidade Federal do Rio Grande do Norte, UFRN, Natal, RN.

SOUZA, RAFAEL B. M. DE (2006), Projeto de reguladores fuzzy takagi-sugeno utilizando as condições iniciais da planta, Dissertação de mestrado, Universidade Estadual Paulista, UNESP, Ilha Solteira, SP.

TAKAGI, T. & M. SUGENO (1985), 'Fuzzy identification of system and its applications to modeling and control', *IEEE Transactions on Systems, Man and Cybernetics* **15**(1), 116–132.

TANSCHEIT, R. (1999), 'Fundamentos de lógica fuzzy e controle fuzzy', Pontifícia Universidade Católica - RJ. Apostila de Aula do Departamento de Engenharia Eletrônica.

YAP, ANTHONY E. & ARTHUR H. DEPOI (2009), 'Method and apparatus for minimizing forces on a web'.

ZADEH, LOFT. A. (1965), 'Fuzzy sets', *Information and Control 8* pp. 338–353.

ZADEH, LOFT. A. (1973), 'Outline of a new approach to the analysis of complex systems and decision processes', *IEEE Transactions on Systems, Man and Cybernetics SMC-3* pp. 28–44.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Administração
Baixar livros de Agronomia
Baixar livros de Arquitetura
Baixar livros de Artes
Baixar livros de Astronomia
Baixar livros de Biologia Geral
Baixar livros de Ciência da Computação
Baixar livros de Ciência da Informação
Baixar livros de Ciência Política
Baixar livros de Ciências da Saúde
Baixar livros de Comunicação
Baixar livros do Conselho Nacional de Educação - CNE
Baixar livros de Defesa civil
Baixar livros de Direito
Baixar livros de Direitos humanos
Baixar livros de Economia
Baixar livros de Economia Doméstica
Baixar livros de Educação
Baixar livros de Educação - Trânsito
Baixar livros de Educação Física
Baixar livros de Engenharia Aeroespacial
Baixar livros de Farmácia
Baixar livros de Filosofia
Baixar livros de Física
Baixar livros de Geociências
Baixar livros de Geografia
Baixar livros de História
Baixar livros de Línguas

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo