ESCOLA FEDERAL DE ENGENHARIA DE ITAJUBÁ

# SISTEMA COMPILADOR FUZZY

Área de Automação e Sistemas Elétricos Industriais

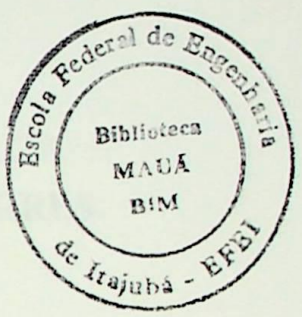
LEONARDO DE MELLO HONÓRIO

ITAJUBÁ - MG
2000


**EFEI**

**Escola Federal de Engenharia de Itajubá**

**Instituto de Engenharia Elétrica**
**Departamento de Eletrônica**


# Sistema Compilador Fuzzy

**Área de Automação e Sistemas Elétricos Industriais**


**Eng. Leonardo de Mello Honório**

# Sistema Compilador Fuzzy

LEONARDO DE MELLO HONÓRIO

ORIENTADOR: DR. LUIZ EDIVAL DE SOUZA
COORIENTADOR: PHD. GERMANO L. TORRES

DISSERTAÇÃO APRESENTADA À ESCOLA FEDERAL DE ENGENHARIA DE ITAJUBÁ PARA A OBTENÇÃO DO TÍTULO DE MESTRE EM CIÊNICAS EM ENGENHARIA ELÉTRICA

ITAJUBÁ
ESTADO DE MINAS GERAIS – BRASIL
2000

Honório, Leonardo de Mello.

Sistema Compilador Fuzzy / Leonardo de Mello Honório – EFEI – 120p

Monografia apresentada a EFEI para obtenção do grau de mestre em ciências em engenharia.

1. Fuzzy 2. Compilador 3. Controle Fuzzy

I. Título.

Banca Examinadora

.....

.....

.....

.....

# *Mensagem*

"Sim eu acredito que Deus é como uma usina de força, como as que geram eletricidade, uma espécie de usina elétrica. E que ELE é um poder supremo, que não é bom nem ruim, nem de esquerda nem de direita, nem negro nem branco. ELE simplesmente É. E é possível Ter acesso a essa fonte de força para um fim desejado. Assim como a eletricidade pode matar gente em cadeira elétrica ou iluminar uma sala. Eu acho que "Deus É." A gente tem que agradecer a DEUS ou seja o que for que está lá em cima pelo fato de sobrevivermos. ...

Houve uma grande mudança e vamos rumo a um futuro desconhecido. Mas ainda estamos aqui. Enquanto há vida há esperança."

John Lennon

# *Agradecimentos*

*A meus pais, que sempre me apoiaram.*

*Prof. Luiz Edival de Souza – pela orientação, disponibilidade e incentivo para a realização do trabalho.*

*Prof. Germano L. Torres – pelas sugestões pertinentes, acompanhamento e apoio que contribuíram para o embasamento desse projeto.*

*Prof. Antônio Carlos Zambroni – pelas informações, esclarecimentos e opiniões profissionais tão importantes.*

*Aos colegas de estudo e trabalho, principalmente os eng.s Eric Morais Abrahao, Eduardo Mendes e Anderson Henriques pela ajuda, convivência e pelo humor nas situações difíceis.*

*Aos professores e alunos do GAII / Labfield - EFEI pela colaboração.*

*Aos funcionários da EFEI pela atenção.*

# *Dedicatória*

***A minha família e amigos.***

# *Resumo*

Cada vez mais utilizados em todas as áreas, os sistemas de controle que utilizam a lógica difusa (fuzzy), se mostraram como uma alternativa confiável. Diferente das formas tradicionais de controle, fuzzy, através da linguagem natural, utiliza o conhecimento heurístico humano para gerar complexos modelos matemáticos. Porém a implementação desses sistemas não é trivial. Para tanto, esse trabalho desenvolveu uma ferramenta compiladora que permite a rápida implementação da lógica difusa, transformando os campos de pertinência, as regras e o processo de inferência em código fonte da linguagem C, ou Matlab®.

Tal sistema compilador fuzzy é dotado de um completo ambiente gráfico que interage com o operador e possui as funções básicas necessárias para o desenvolvimento, facilitando assim a implementação da lógica difusa. Permite também, que o operador possa interagir com o processo de compilação, gerando ou modificando as funções correntes. Possibilita assim, além da implementação de novas metodologias de desenvolvimento da lógica difusa a criação de rotinas para acesso a hardwares externos, como controladores lógico programáveis, placas de aquisição de dados ou dispositivos de saída e entrada. O que torna o sistema compilador fuzzy uma ferramenta não só versátil para o aprendizado e pesquisa, como para a implementação de sistemas reais de controle.

# 1

# Introdução

## 1.1 Considerações Iniciais

Desenvolvida nos anos 60 e utilizada nas indústria a partir dos anos 70, sistemas que utilizam a lógica difusa (FUZZY) [1,2] mostraram-se poderosos aliados no que tange a facilidade de configuração em relação aos métodos tradicionais de controle e com resultados expressivos tratando os problemas sob um novo paradigma. Diferente da então tradicional lógica booleana que trata o mundo real como tendo apenas duas classes (verdadeiro ou falso), FUZZY atribui às variáveis reais (temperatura, pressão, tensão, etc) classes de conjuntos associados a termos lingüísticos (alto, baixo, quente, frio, etc) denominados pela literatura de MEMBERSHIP. Com isso a determinação das regras de controle de um dado sistema é feito através de um conjunto de regras formado pela união dessas classes que formam a representação de todos os estados das grandezas utilizadas no sistema. Assim FUZZY ganha a grande capacidade de trabalhar os problemas utilizando uma linguagem natural não necessitando o equacionamento matemático de complexos sistemas. Isso permite dimensi-

onar e modelar problemas utilizando o conhecimento heurístico e o bom senso do projetista.

Por essa facilidade FUZZY é difundido na economia , medicina , engenharia ou qualquer outro meio científico que necessite de um sistema capaz de trabalhar de forma robusta com as incertezas [3, 4].

Porém, apesar da grande facilidade de desenvolvimento da Lógica Fuzzy, sua implementação em linguagens de programação para que realmente possa ser utilizada não é trivial. Em vista disso foram desenvolvidos vários softwares capazes de desenvolver tal lógica. Dentre eles, dois se sobressaem; a Toolbox Fuzzy do Matlab® [5] e o aplicativo FuzzyTech. Apesar de ambos serem poderosas ferramentas de desenvolvimento pecam quanto a facilidade de interação com o do usuário. Algumas características dos softwares estudados são visualizadas abaixo;

**FuzzyTech [6]**

- Possuiu pacotes fechados quanto ao número de entradas, campos, regras e saídas.
- Muitas telas simultâneas para a entrada e edição de regras, complicando o operador.
- Possui um módulo de otimização de regras.
- Possui módulo de simulação e otimização.
- Não gera banco de dados relacionados com as entradas executadas pelo usuário.
- Pode ser utilizado em hardwares industriais.

**ToolBox Fuzzy do MatLab® [5]**

- Número ilimitado de entradas e saídas, campos e regras.
- Boa interface para entrada de dados.
- Aberto à implementação de novas funções (de forma restrita).
- Possui módulo de simulação (simulink).
- Não gera banco de dados relacionados com as entradas executadas pelo usuário.

## 1.2 Utilização de Fuzzy

Não é objetivo desse trabalho expor a matemática que existe por trás de fuzzy. Porém, para um melhor entendimento, será desenvolvido em termos gerais sua forma de implementação [2, 7, 8, 9].

Como dito anteriormente a lógica fuzzy trabalha com termos lingüisticos, ou seja, com a linguagem natural de comunicação dos seres humanos. Um exemplo pode ser claramente visualizado na frase abaixo;

*"Se a temperatura estiver muito quente, ligar o ar condicionado em muito frio"*

Quando analisamos a frase acima, podemos perceber dois pontos;

- A sensibilidade de quente e frio é diferente para cada indivíduo.
- Independentemente da sensibilidade individual, a regra sempre é válida.

Ou seja, ao trabalhar com termos lingüísticos ao invés de valores pré determinados trabalha-se com um conjunto de regras referentes as ações não sofrerão alterações. Extrapo-

lando esse raciocínio para um sistema complexo de controle pode-se claramente ver essa vantagem. Onde, uma vez definidas as regras, estas serão mantidas para alterações em parâmetros de ajuste (desde que essas não afetem as características intrínsecas do sistema).

Outra grande diferença de fuzzy em relação a outras formas de desenvolvimento matemático é como são trabalhados os parâmetros e ajustes necessários para o desenvolvimento de uma aplicação.

Fuzzy relaciona os elementos a sociedades (tratadas na literatura por *membership* ou funções de pertinência) e atribuiu a esses elementos graus de pertinência (de 0 a 1) em relação às respectivas *membership*. Sendo que um elemento pode estar associado a mais de uma *membership*.

O exemplo clássico tratado no início da seção ilustra bem o processo, onde;

"Muito quente" e "Muito Frio" são memberships atribuídos à temperatura e o ajuste do ar condicionado. Para cada valor de temperatura lido, é atribuído um valor de pertinência às devidas *memberships*.

Para que isso ocorra, cada grandeza relacionada no processo deve ser escalonada em memberships, inclusive as grandezas que são utilizadas como resultado da lógica. Isso pode ser melhor visualizado na forma gráfica, como mostra a figura 1.1.

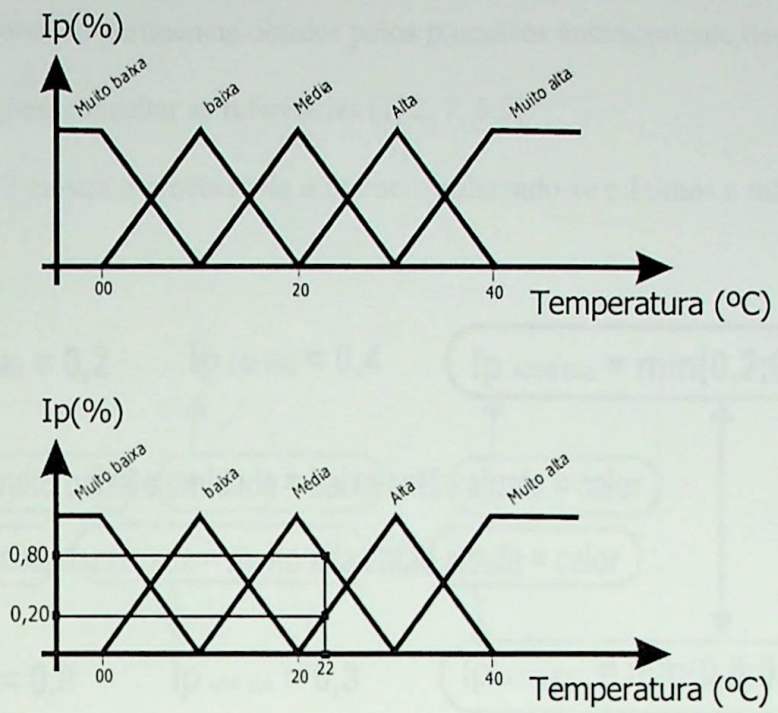



*figura 1.1 – a) divisão da grandeza em memberships;*
*b) Obtenção dos índices de pertinência*

A figura 1.1a mostra uma grandeza (no caso, temperatura) dividida em cinco memberships; "muito alta", "alta", "média", "baixa" e "muito baixa". A figura 1.1b mostra, para um valor de temperatura, a obtenção dos índices de pertinência em relação aos memberships relacionados. Onde, dado um valor, é feita a projeção vertical deste para a obtenção dos pontos de interseção com os memberships. Uma nova projeção horizontal, feita a partir dos pontos de interseção obtidos, é feita para a obtenção dos índices de pertinência do valor lido com esses memberships.

A próxima etapa de um processo de lógica fuzzy é atribuir os índices de pertinência calculados a um processo de inferência, ou seja, as regras previamente determinadas que expressam como o sistema se comportará.

Existem vários métodos para desenvolver o processo de inferência já mencionado. Porém, para esse ensaio, será demonstrado o método de mínimos e máximos. Esse método

trabalha com os valores de pertinência obtidos pelos processos anteriormente descritos. Para maiores informações, consultar as referências [1, 2, 7, 8,9].

A figura 1.2 mostra o processo de inferência utilizando-se mínimos e máximos.

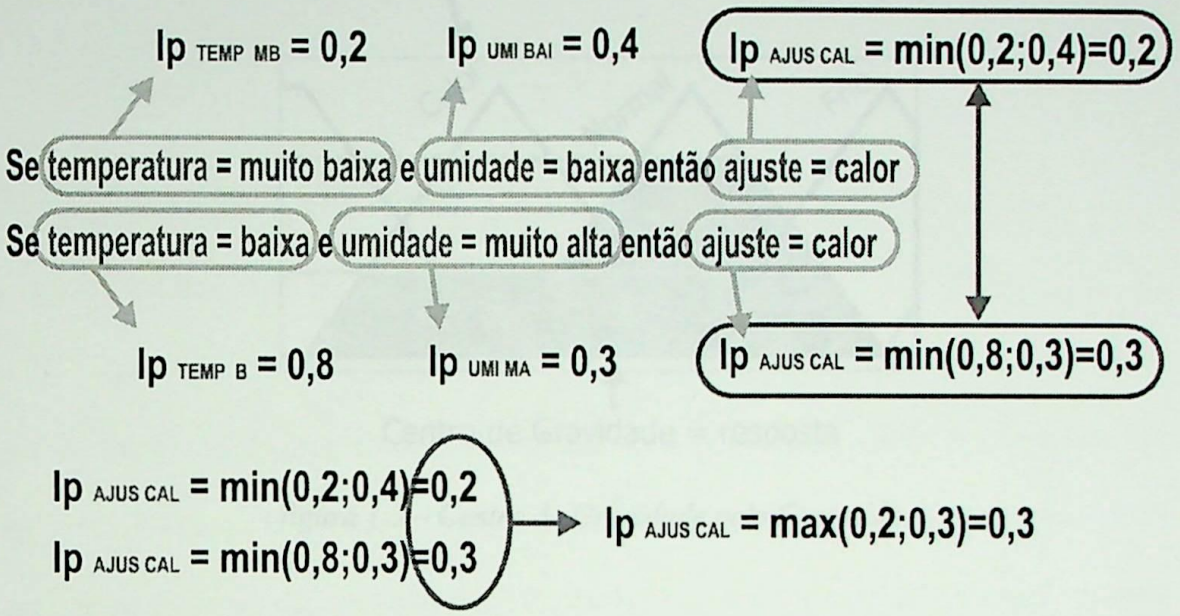



*figura 1.2 – processo de inferência por máximos e mínimos*

Observa-se na figura 1.2 que nesse processo de inferência, atribui-se ao índice da preposição um índice de pertinência igual ao mínimo dos valores de pertinência dos índices contidos na proposição. Podem existir casos em que um membership esteja em mais de uma proposição, obtendo-se assim mais de um valor para o mesmo. Nesse caso o índice final para esse membership é o máximo dos índices atribuídos ao mesmo.

O último passo desse processo é o cálculo do valor de saída, esse processo é tratado na literatura com defuzzificação. Como o processo de inferência, existem diversas formar de defuzzificação [1, 2, 7, 9]. O que será mostrado aqui é o de centro de gravidade pelo centróide como pode ser visto na figura 1.3.

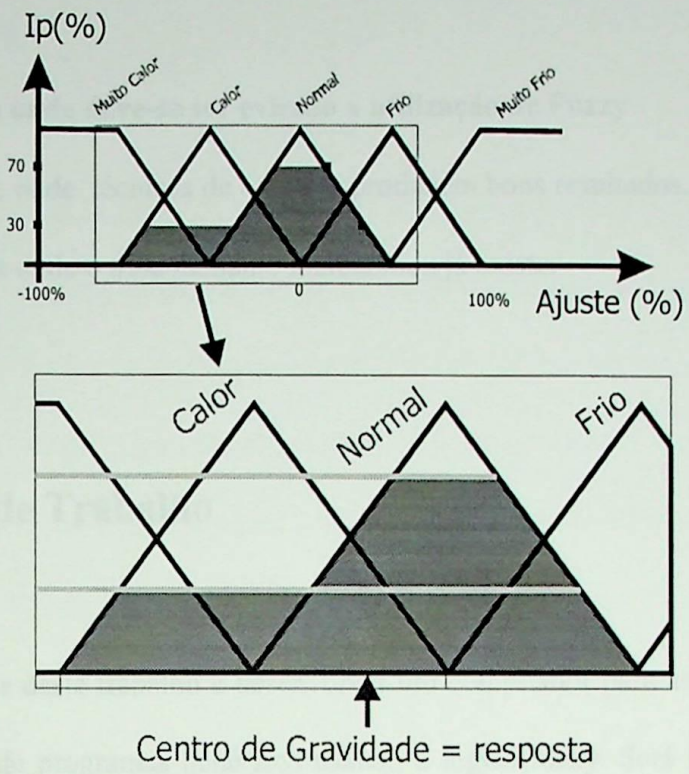



*figura 1.3 – Centro de Gravidade pelo Centroide*

No exemplo mostrado acima, tem-se dois índices de pertinência associados dois diferentes memberships. Acha-se a área sob cada membership tendo como máximo o valor do índice de pertinência associado. A resposta da lógica, nesse caso, é o centro de gravidade da área formada pela união dos dois triângulos encontrados, vide figura 1.3.

Também é interessante fazer-se claro algumas peculiaridades que são motivo de dúvidas para muitos;

**Condições onde Fuzzy é bem aplicável**

- Em sistemas muito complexos, onde é difícil desenvolver o modelo matemático.
- Para sistemas extremamente não lineares que podem ser bem explicados heuristicamente.
- Se o sistema pode ser descrito através de termos lingüísticos.

**Condições onde deve-se ser evitado a utilização de Fuzzy**

- Em sistemas onde técnicas de controle produzem bons resultados.
- Em sistemas onde o modelamento matemático já existe.

## 1.3 Proposta de Trabalho

A proposta deste trabalho é desenvolver um compilador para auxiliar os projetistas na construção de programas genéricos usando a lógica fuzzy. Será implementado em ambiente Windows utilizando a linguagem de programação Visual Basic que proporciona grandes recursos em manipulação de banco de dados e geração de telas gráficas. O programa terá uma rica interface gráfica e sistema de armazenamento de dados no formato .mdb (podendo ser acessado por programas como Excel®) e terá capacidade de gerar código em linguagem C [24,25]ou linguagem de programação para Matlab® [5]. Para uma maior flexibilidade será acoplado ao pacote um módulo externo que atuará como uma biblioteca de funções editáveis pelo usuário. Isso permitirá a geração de código de programas para os mais diversos fins, como acesso a placa de aquisições de dados, controladores lógico programáveis ou para gerar novas funções para os processos de inferência das regras. Serão implementados diferentes métodos de defuzzificação permitindo ao usuário ter um maior controle do tipo de sistema se deseja gerar.

## 1.4 Organização

Este trabalho está organizado na seguinte forma; o capítulo 2 mostra o desenvolvimento do processo de compilação. Abrange o dimensionamento e o armazenamento do banco de dados, a análise do processo e os algoritmos de implementação. O capítulo 3 descreve o processo de operação do compilador, como se processa a entrada de dados e a geração do código. O capítulo 4 faz uma comparação prática das técnicas de defuzzificação, introduz uma técnica fuzzy com ajuste de ganho automático baseado em um controlador adaptativo e mostra a implementação de um problema utilizando a técnica de controle fuzzy adaptativo. Por fim no capítulo 5 será discutido a avaliação dos resultados.

# 2

# Análise, Projeto e Implementação

*O objetivo da análise e projeto é mostrar o desenvolvimento do programa desde o fluxo de informações, realizações de tarefas, dimensionamento e forma de armazenamento dos dados utilizados até seu algoritmo de execução. Desta forma será mostrado, neste capítulo, primeiro uma idéia global de todos os eventos realizáveis que serão detalhados até o nível de implementação passando por cada etapa de desenvolvimento.*

## 2.1. Dimensionamento dos dados utilizados

Temos, como regra geral para um sistema fuzzy, quatro etapas básicas de implementação como é mostrado na figura 2.1:

1 - Definição das grandezas de entrada e saída que fazem parte do processo.

2 - Ajuste dos campos de pertinência de cada grandeza.

3 - Definição das regras associadas ao processo de defuzzificação. Sendo este último pré-definido, cabendo ao usuário apenas na escolha dentre um dos possíveis métodos.

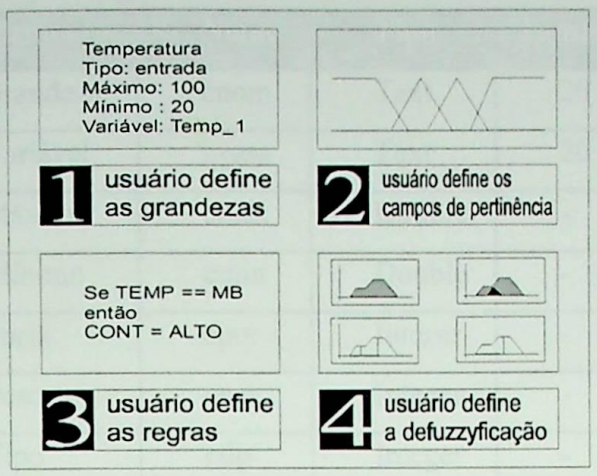





*Figura 2.1 - Processo de definição da lógica difusa pelo operador do compilador fuzzy*

Logo, para ser possível a criação de um sistema de controle com lógica fuzzy, é necessário que o compilador seja capaz de armazenar os dados das etapas 1,2 e 3. Foram então criadas três tabelas de armazenamento de dados interligadas através de campos chaves. Desta forma cada nova entrada de dados feita pelo usuário em uma dada etapa será armazenada sob a forma de um registro (uma linha da tabela) em uma das três tabelas correspondente à mesma. Os campos (colunas) de cada tabela foram dimensionados tendo em vista não só a redundância de informações como também a facilidade de acesso dos dados.

### 2.1.1. Banco de Dados das Entradas / Saídas

Nome: Campo, representação na tabela 2.1.

Função: Armazenar dados sobre as entradas e saídas. É necessário definir a grandeza e seu respectivo tipo de atuação(entrada analógica, saída analógica ou se é uma grandeza de relacionamento fuzzy); a variável associada a essa grandeza; para efeito de uma escala gráfica mais precisa, o mínimo e o máximo assumidos; o peso; e a porta de entrada ou saída (caso seja utilizado uma placa de aquisição de dados).

| Elemento | Nome | Tipo | Tamanho |
|---|---|---|---|
| Grandeza | cnom | Text | 20 |
| Variável | cvass | Text | 20 |
| Máximo | cmax | Double | - |
| Mínimo | cmin | Double | - |
| Porta | cpor | Integer | - |
| Peso | cpes | Integer | - |
| Tipo | ctip | Integer | - |

*Tabela 2.1 - Descrição dos campos da tabela de entrada / saída*

- **Grandeza**

Armazena o nome da grandeza lida ou controlada.

- **Variável**

Nome da variável atribuída à respectiva grandeza.

- **Máximo**

Valor máximo estipulado que a variável pode assumir.

- **Mínimo**

Valor mínimo estipulado que a variável pode assumir.

- **Peso**

Atribui um peso a grandeza. Opcionalmente utilizado durante o processo de cálculo dos índices de pertinência.

- **Porta**

Define o endereço de direcionamento dos elementos externos (placas de aquisição de dados, PLC's ou simplesmente a saída paralela do micro) de execução da leitura ou entrada das grandezas. Valor de entrada deve ser em hexadecimal.

- **Tipo**

Define se a variável em questão é uma entrada analógica, uma saída analógica ou um relacionamento fuzzy.

### 2.1.2. Banco de Dados dos Membership's

Nome: Mship, representação na tabela 2.2.

Função: Armazenar dados sobre os campos de pertinência. Os membership's serão armazenados em forma de segmentos de reta, onde teremos a coordenada inicial e a final dos mesmos. Para efeito de filtragem e seleção também é necessário acoplar a cada registro um número indexador, o nome do membership, sua grandeza de atuação e sua respectiva variável de acesso. Sendo esta última desnecessária em termos de redundância de informação haja visto que o banco de dados já possui um campo chave (nome da grandeza) que o liga com o banco de dados das entradas/saídas, onde esta informação está disponível, porém, neste caso, é preferível a redundância do que o esforço computacional de uma busca.

| Elemento | Nome | Tipo | Tamanho |
|---|---|---|---|
| Indexador | msnum | Integer | - |
| Grandeza | msnom | Text | 20 |
| Variável | msvas | Text | 20 |
| Nome do Campo | mscam | Text | 20 |
| Posição X1 | posx1 | Double | - |
| Posição Y1 | posy1 | Double | - |
| Posição X2 | posx2 | Double | - |
| Posição Y2 | posy2 | Double | - |

*Tabela 2.2 - Descrição dos campos da tabela dos campos de pertinência*

- **Indexador**

Função de indexar os campos de pertinência por uma ordem específica.

- **Grandeza**

Possui a função de chave de ligação entre o banco de dados membership e o de entrada/saída.

- **Variável**

Possui função repetida. Este campo poderia não existir, sua função é de economia de tempo de busca. Armazena a variável de acesso da grandeza.

- **Nome do Campo**

Armazena o nome do membership editado.

- **Posição X1**

Armazena a posição x inferior (em relação ao eixo y) do segmento de reta. Excetuando o caso quando o segmento de reta for paralelo ao eixo x.

- **Posição Y1**

Armazena a posição y inferior (em relação ao eixo y) do segmento de reta. Excetuando o caso quando o segmento de reta for paralelo ao eixo x.

- **Posição X2**

Armazena a posição x superior (em relação ao eixo y) do segmento de reta. Excetuando o caso quando o segmento de reta for paralelo ao eixo x.

- **Posição Y2**

Armazena a posição y superior (em relação ao eixo y) do segmento de reta. Excetuando o caso quando o segmento de reta for paralelo ao eixo x.

### 2.1.3. Banco de Dados das Regras

Nome: Regras, representação na tabela 2.3.

Função: Armazenar dados sobre as regras. Estas serão armazenadas tanto na forma de linha de comando para futuramente facilitar a compilação como na de termos lingüísticos, evitando assim, a implementação de um trabalhoso algoritmo de tradução necessário para a edição das regras pelo usuário.

| Elemento | Nome | Tipo | Tamanho |
|---|---|---|---|
| Indexador | r_num | Integer | - |
| Ação | r_acao | Text | 50 |
| Condição | r_cond | Text | 50 |
| Tradução 1 | r_trad1 | Text | 50 |
| Tradução 2 | r_trad2 | Text | 50 |

*Tabela 2.3 - Descrição dos campos da tabela das regras.*

- **Indexador**

Função de indexar as regras por uma ordem específica.

- **Ação**

Campo que armazena a ação assumida dado uma determinada condição. É armazenada na forma de linha de comando (igual tanto na geração em C quanto em Matlab®).

- **Condição**

Campo que armazena a condição que determinará uma ação específica. É também armazenada na forma de linha de comando.

- **Tradução 1**

Armazena em forma de termos lingüísticos a ação que está armazenada em código de comando.

- **Tradução 2**

Armazena em forma de termos lingüísticos a ação que está armazenada em código de comando.

## 2.2. Análise

A apresentação da análise será feita através de DFD's (diagrama de fluxo de dados) . Uma nota importante na análise é que esta foi desenvolvida sob o paradigma de orientação a eventos, ou seja não existe necessariamente um fluxo definido de dados, deixando o usuário livre para seguir vários caminhos simultaneamente ou não.

### 2.2.1 Análise Inicial

Em uma primeira análise o usuário terá três caminhos a seguir, como mostra a figura 2.2:

1 - Definição dos dados da lógica difusa, onde os dados definidos serão armazenados nas tabelas de dados já mencionadas, pertencentes a um arquivo de trabalho já existente, o fuzzy.mdb.

2 - Manipulação dos dados definidos, onde o usuário poderá gravar o arquivo de trabalho em um arquivo específico ou vice-versa.

3 - Escolha do processo de defuzzificação e do tipo de arquivo gerado.

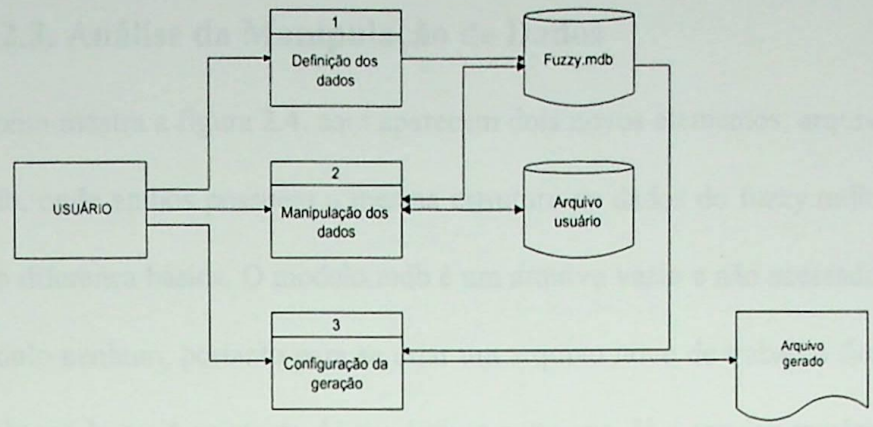



*figura 2.2 - Diagrama de fluxo de dados da análise inicial*

### 2.2.2 Análise da Definição dos Dados

Aprofundando a análise da definição de dados temos o usuário com três opções:

1- Definir as grandezas de entrada/saída

2- Definir os membership's (campos de pertinência) de cada grandeza

3- Definir as regras de controle do sistema

Cada etapa de definição está ligada com sua respectiva tabela de armazenamento de dados, localizada no arquivo de trabalho fuzzy.mdb, como mostra a figura 2.3.

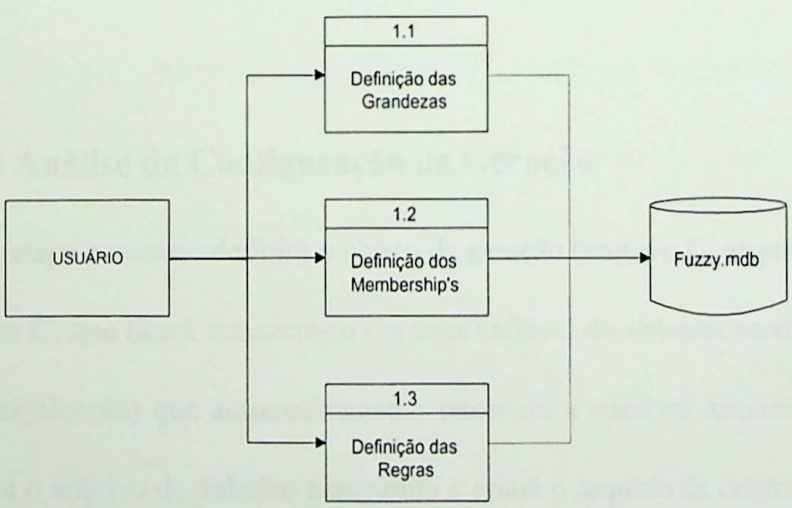



*figura 2.3 - Diagrama de fluxo de dados da definição de dados*

### 2.2.3. Análise da Manipulação de Dados

Como mostra a figura 2.4. aqui aparecem dois novos elementos; arquivo usuário e modelo.mdb, onde ambos possuem a mesma estrutura de dados do fuzzy.mdb, possuindo apenas uma diferença básica. O modelo.mdb é um arquivo vazio e não acessado pelo usuário em módulo nenhum, portanto para se criar um arquivo novo de trabalho fica mais fácil e rápido sobrepô-lo ao já existente do que limpar o mesmo. Já o arquivo usuário é o arquivo de trabalho renomeado, sendo necessário, então, apenas sobrepô-lo ao existente para restaurar seus dados.

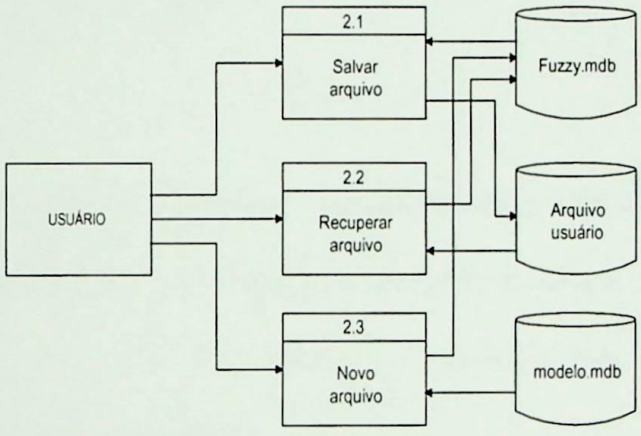



*figura 2.4 - Diagrama de fluxo de dados da manipulação de arquivos*

### 2.2.4 Análise da Configuração da Geração

Nesta etapa o usuário definirá o objeto da geração (arquivo C, arquivo Matlab® ou supervisório em C) que ficará armazenado em uma variável do sistema, escolherá o tipo de geração (defuzzyficação) que automaticamente retornará a variável armazenada anteriormente, acessará o arquivo de trabalho fuzzy.mdb e criará o arquivo de controle fuzzy sob a forma de texto. O usuário poderá gravar o mesmo para disco, como mostra a figura 2.5.

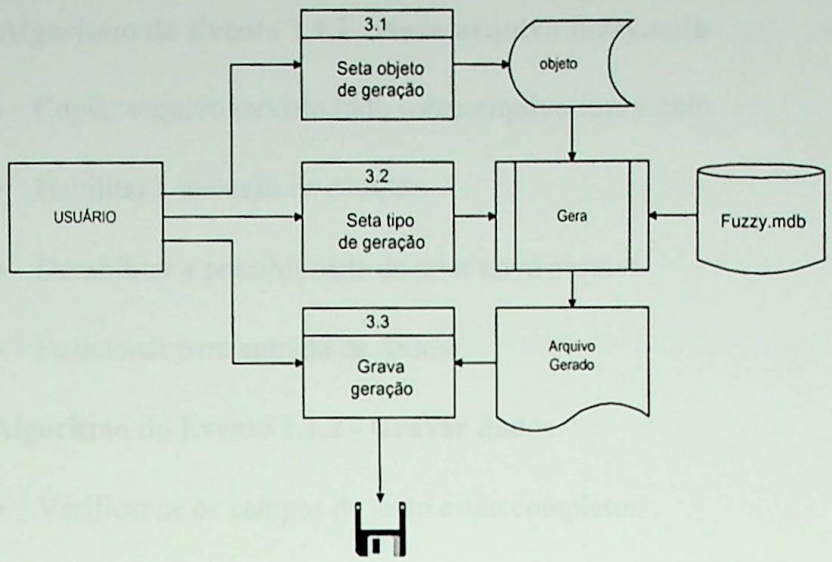



*figura 2.5 - Diagrama de fluxo de dados da análise da configuração da geração*

## 2.3 Projeto

Por definição, projeto de *software* é basicamente definir de uma forma mais precisa a análise inicial, ou seja, mostrar detalhadamente como os eventos com suas respectivas rotinas serão implementados. Em outras palavras, a análise define as ações e o projeto define como serão realizadas essas ações.

### 2.3.1. Projeto da Definição das Grandezas de Entrada/Saída

Nessa seção o usuário faz a entrada e a edição dos dados das grandezas de entrada e saída, como mostra a figura 2.6.

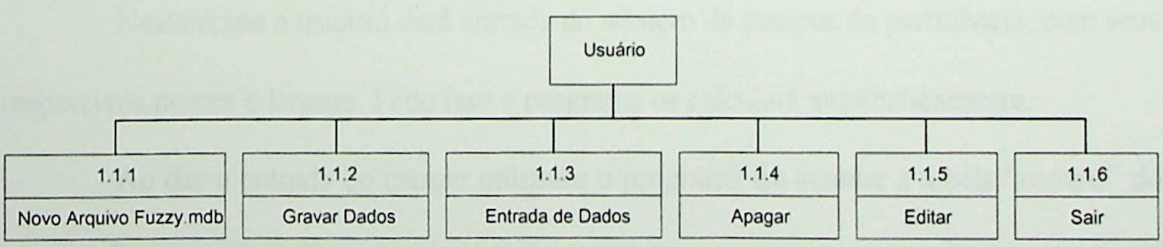



*figura 2.6 - eventos realizáveis na definição de dados*

- **Algoritmo do Evento 1.1.1 - Novo arquivo fuzzy.mdb**
  - Copiar arquivo modelo.mdb sobre arquivo fuzzy.mdb
  - Habilitar a inclusão de campos
  - Desabilitar a possibilidade de criar novo arquivo
  - Posicionar para entrada de dados
- **Algoritmo do Evento 1.1.2 - Gravar dados**
  - Verificar se os campos de texto estão completos
  - Enviar dados dos campos textos aos seus respectivos campos na tabela *campo* no arquivo fuzzy.mdb
  - Limpar campos de texto
- **Algoritmos dos Evento 1.1.3 ao Evento 1.1.6**

São eventos padrão ou encapsulados, ou seja foram utilizados objetos prontos nas quais não se tem acesso ao código, somente ao evento.

### 2.3.2 Projeto da Definição os Campos de Pertinência

Nessa seção é dada a entrada dos campos de pertinência referentes as grandezas definidas na seção anterior.

- **Projeto da Definição Automática de Campos de Pertinência**

Nesta etapa o usuário dará entrada do número de campos de pertinência, com seus respectivos nomes e largura. Feito isso o programa os calculará automaticamente.

Ao dar a entrada do campo utilizado o programa irá acessar a tabela "campo" do arquivo fuzzy.mdb e irá retornar os valores de máximo e mínimo pertinentes ao mesmo. O valor da precisão (largura) do membership é dado em percentagem em relação à faixa de

atuação de seu respectivo campo. O número de memberships setados pelo usuário não representa o número total, mas sim o número de memberships à esquerda ou à direita, pois este módulo assistente gera somente conjuntos simétricos a um membership que estará no centro de gravidade (*set point)* do conjunto que também é definido pelo usuário. Sendo assim o conjunto dos campos de pertinência gerados ficará conforme a figura 2.7. O fluxograma de eventos pode ser visualizado na figura 2.8.

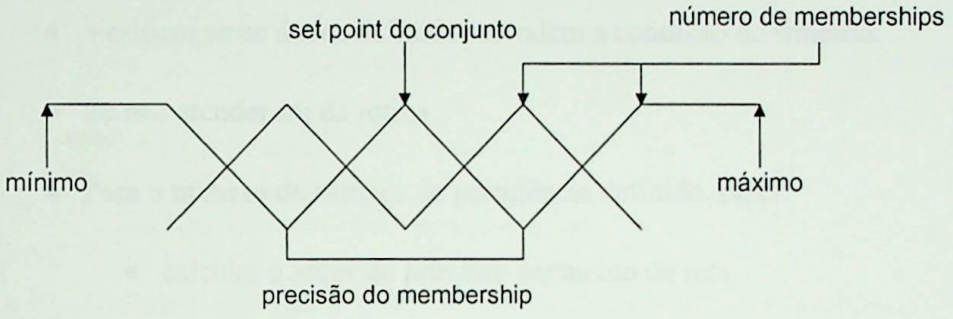



*figura. 2.7 - visualização da forma de geração dos memberships no modo de assistente*

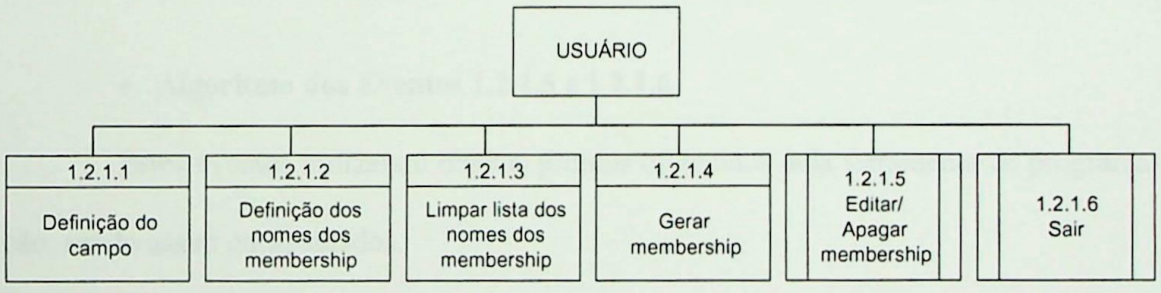



*figura 2.8 - eventos realizáveis na definição automática de memberships*

- **Algoritmo do Evento 1.2.1.1**
  - Selecionar campo
  - Localizar na tabela campo os dados do campo selecionado
- **Algoritmo do Evento 1.2.1.2**
  - Entrar com o nome do membership corrente

- Repetir enquanto não extrapolar o número de campos de pertinência definidos pelo usuário.

- **Algoritmo do Evento 1.2.1.3**
  - Limpar a lista dos campos de pertinência.
- **Algoritmo do Evento 1.2.1.4**
  - Calcular o valor da precisão em unidade
  - Calcular os limites mínimo e máximo para a precisão e centro de gravidade
  - Verificar se os dados definidos atendem a condição de simetria.
  - Se não atender sai da rotina
  - Para o número de campos de pertinência definido, fazer:
    - calcular o valor do primeiro segmento de reta
    - calcular o valor do segundo segmento de reta
    - gravar na tabela "mship" no arquivo fuzzy.mdb

- **Algoritmo dos Eventos 1.2.1.5 e 1.2.1.6**

Estes eventos utilizaram objetos prontos oferecidos pela ferramenta de programação, sendo assim encapsulados.

- **Projeto da Definição Manual de Membership**

Diferente do módulo de definição automática, neste módulo a responsabilidade de dar entrada do número e dos nomes de memberships é toda do usuário que deverá também definir os pontos (x1,y1,x2,y2) de cada segmento de reta de cada membership, figura 2.9.

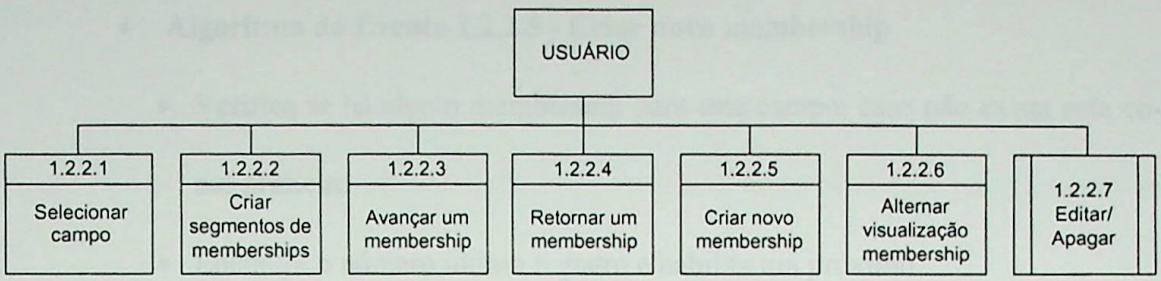



*figura. 2.9 - eventos realizáveis na criação manual de memberships*

- **Algoritmo do Evento 1.2.2.1 - Selecionar campo de atuação**
  - Localiza na tabela "campo" os valores máximo e mínimo
  - Gera uma tabela baseada em "mship" contento somente os memberships do campo selecionado
  - Se houver memberships monta o gráfico
- **Algoritmo do Evento 1.2.2.2 - Criar segmentos de memberships**
  - Verifica a existência do nome do membership; caso não exista aborta
  - Verifica a duplicidade do nome do membership; caso exista aborta
  - Lê os valores de x1 e x2 dados pelo usuário
  - Cria uma rampa ou uma paralela dependendo da escolha do usuário
  - Desenha objeto na tela
  - Grava o objeto na tabela "mship"
- **Algoritmo do Evento 1.2.2.3 - Avançar um membership**
  - Avança um registro na tabela criada no evento 1.2.2.1
- **Algoritmo do Evento 1.2.2.4 - Retornar um membership**
  - Retorna um registro na tabela criada no evento 1.2.2.1

- **Algoritmo do Evento 1.2.2.5 - Criar novo membership**
  - Verifica se há algum membership para este campo; caso não exista seta como primeiro
  - Encontra o número último registro e habilita um próximo
- **Algoritmo do Evento 1.2.2.6 - Alternar visualização de memberships**

  Para visualização geral:
  - Seleciona todos os registros da tabela criada no evento 1.2.2.1

  Para visusalização individual
  - Seleciona os registros com o mesmo nome do registro atual, lembrando que cada registro é um segmento de reta de um membership.

- **Algoritmo do Evento 1.2.2.7 - Editar / Apagar**

Estes eventos utilizaram objetos prontos oferecidos pela ferramenta de programação, sendo assim encapsulados.

### 2.3.3 Projeto da Definição das Regras

Nessa seção o usuário dá a entrada de regra que determina o comportamento da atuação do sistema, figura 2.10.


*Figura 2.10 - Eventos realizáveis na geração de regras*

- **Algoritmo do Evento 1.3.1 - Escolha do Campo**
  - Seleciona da tabela "mship" registros com nome do campo igual ao selecionado
  - Se não houver registros abandona
  - Seleciona da tabela "mship" os memberships dos registros selecionados
- **Algoritmo do Evento 1.3.2 - Escolha do Membership**
  - Selecionar um membership da tabela montada pelo evento 1.3.1
- **Algoritmo do Evento 1.3.3 - Escolha da Função (ou operação)**
  - Selecionar a função
  - Verificar o flag de localização para determinar se é uma ação ou condição
  - Montar a função em linha programação em termos lingüísticos
- **Algoritmo do Evento 1.3.4 - Retorna regra**
  - Retorna um registro da tabela selecionada no evento 1.3.1
- **Algoritmo do Evento 1.3.5 - Avança regra**
  - Avança um registro da tabela selecionada no evento 1.3.1
- **Algoritmo do Evento 1.3.6 - Alteração de regras**
  - Grava alterações por cima do registro atual
- **Algoritmo do Evento 1.3.7 - Deletar regras**
  - Deleta registro atual
  - Verifica a existência de registros
  - Retorna um registro

### 2.3.4 Projeto da Configuração da Geração

Neste módulo o usuário apenas seta o objeto e o tipo de geração, não necessitando mais entrada de dados. O usuário pode gerar até 16 tipos diferentes de arquivos com a mesma lógica difusa porém se assemelham por possuir a mesma estrutura lógica diferindo apenas na linguagem de implementação e na forma como é processada a entrada e saída de dados. Estes arquivos são gerados a partir da combinação de funções primárias. Sendo assim neste módulo será mostrado além destas funções primárias e suas combinações que dão origem aos arquivos fuzzy, o algoritmo de defuzzyficação que é o processo mais divergente neste módulo de geração, figura 2.11.

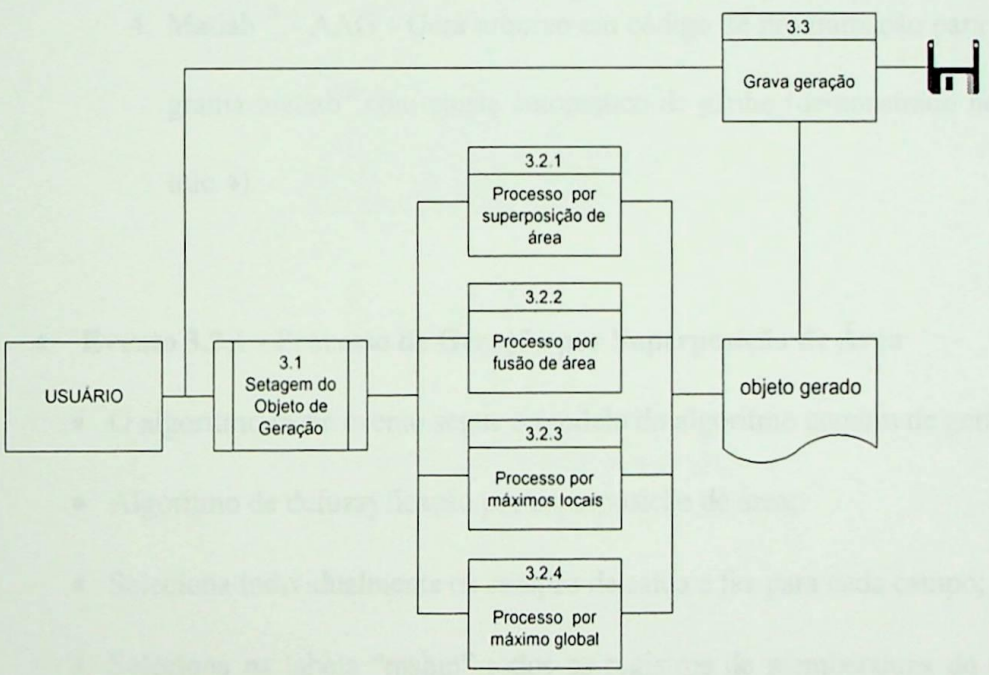



*Figura 2.11 - Projeto do módulo de geração*

- **Algoritmo Comum de Geração**
  - Inicializar variáveis
  - Ler entrada de dados (seja via teclado ou placa de aquisição de dados)
  - Transformar entrada de dados em índices de pertinência

- Aplicar índices de pertinência no processo de inferência (aplicação das regras setadas pelo usuário)
- Utilizar os índices de pertinência obtidos pelo processo de inferência para calcular a resposta (processo de defuzzyficação)

- **Evento 3.1 - Setar Objeto de Geração**

  São quatro objetos de geração:

  1. C - PAD - Gera arquivo C com I/O de dados via placa de aquisição
  2. C - EMD - Gera arquivo C com I/O de dados via teclado
  3. Matlab ® - Gera arquivo em código de programação para o programa matlab®
  4. Matlab ® - AAG - Gera arquivo em código de programação para o programa matlab® com ajuste automático de ganho (demonstrado no capítulo 4).

- **Evento 3.2.1 - Processo de Geração por Superposição de Área**
  - O algoritmo deste evento segue o modelo do algoritmo comum de geração.
  - Algoritmo de defuzzyficação por superposição de área;
  - Seleciona individualmente os campos de saída e faz para cada campo;
  - Seleciona na tabela "mship" todos os registros de memberships do campo selecionado
  - Se o ínidice de pertinência do campo não for nulo, calcula a resposta
  - Para mais de um membership, calcular a média ponderada com o anterior
  - método de cálculo da resposta

**1º caso ; $x_1 < x_2$**

Decompõe-se o trapézio em dois polígonos, um triângulo e um retângulo, calcula-se o centro de gravidade de cada um e logo após tira-se a média ponderada para se achar o centro de gravidade, figura 2.12.

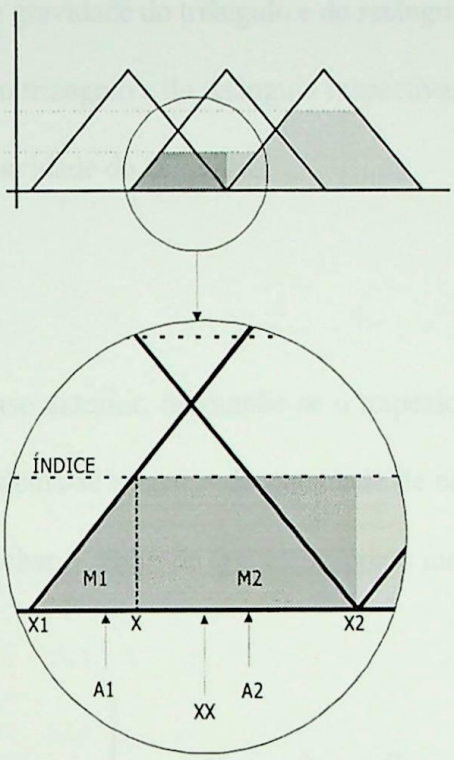



*figura 2.12 – Decomposição do trapézio em polígonos para cálculo do centro de gravidade para o caso em que $x_1 < x_2$*

Onde, dados;

$$X = indice \cdot (X_2 - X_1) + X_1$$

$$A_1 = X_1 + \frac{2}{3} \cdot (X - X_1)$$

$$A_2 = X + \frac{(X_2 - X)}{2}$$

$$M_1 = \frac{(X - X_1)}{2} \cdot indice$$

$$M_2 = (X_2 - X) \cdot indice$$

Tem-se o centro de gravidade desta área como sendo;

$$XX = \frac{A_1 \cdot M_1 + A_2 \cdot M_2}{M_1 + M_2}$$

onde tem-se;

$A_1$, $A_2$ - centro de gravidade do triângulo e do retângulo respectivamente.

$M_1$, $M_2$ - massa do triângulo e do retângulo respectivamente.

XX - centro de gravidade do conjunto.

**2º caso (x1 > x2)**

Semelhante ao caso anterior, decompõe-se o trapézio em dois polígonos, um triângulo e um retângulo, calcula-se o centro de gravidade de cada um e logo após tira-se a média ponderada para se achar o centro de gravidade, como mostra a figura 213.

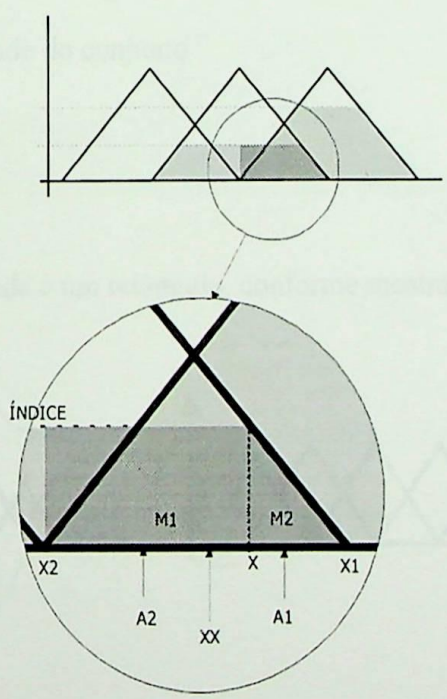



*Figura2.13 – Decomposição do trapézio em polígonos para cálculo do centro de gravidade para o caso em que $x_1 > x_2$*

Onde, dados;

$$X = X_1 - indice \cdot (X_1 - X_2)$$

$$A_1 = X_1 + \frac{1}{3} \cdot (X_1 - X)$$

$$A_2 = X_2 + \frac{(X + X_2)}{2}$$

$$M_1 = \frac{(X_1 - X)}{2} \cdot indice$$

$$M_2 = (X - X_2) \cdot indice$$

Tem-se o centro de gravidade desta área como sendo;

$$XX = \frac{A_1 \cdot M_1 + A_2 \cdot M_2}{M_1 + M_2}$$

onde tem-se;

$A_1$, $A_2$ - centro de gravidade do retângulo e do triângulo respectivamente

$M_1$, $M_2$ - massa do retângulo e do triângulo respectivamente

XX - centro de gravidade do conjunto

**3º caso $X_1$=$X_2$**

Neste caso, a área gerada é um retângulo, conforme mostra a figura 2.14

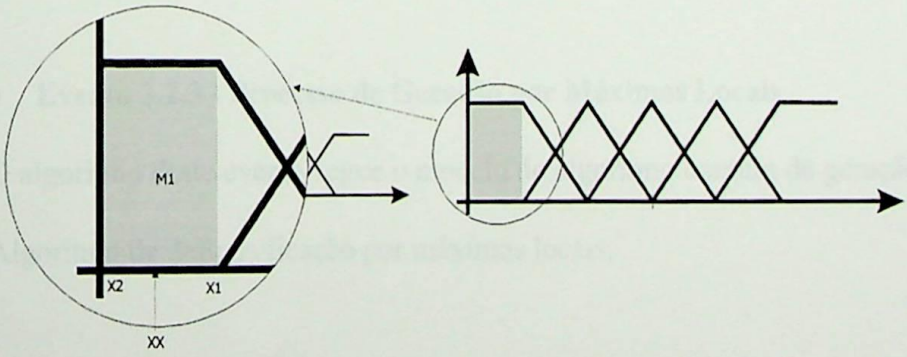



*Figura 2.14 – Decomposição do trapézio em polígonos para cálculo do centro de gravidade para o caso em que $x_1$=$x_2$*

- **Evento 3.2.2 - Processo de Geração por Fusão de Área**
  - algoritmo deste evento segue o modelo do algoritmo comum de geração.
  - Algoritmo de defuzzyficação por fusão de área;
  - O processo utilizado é o de integração por partes
  - Encontra-se os pontos de mínimo e máximo dos memberships de índice não nulos
  - Varia do mínimo para o máximo calculando o maior valor de pertinência de todos os memberships envolvidos no processo, conforme figura 2.15.
  - Soma-se o valor encontrado ao acumulando.
  - Armazena o ponto e o valor acumulado em uma matriz
  - Procura o valor médio final na matriz
  - Retorna o ponto correspondente armazenado também na matriz

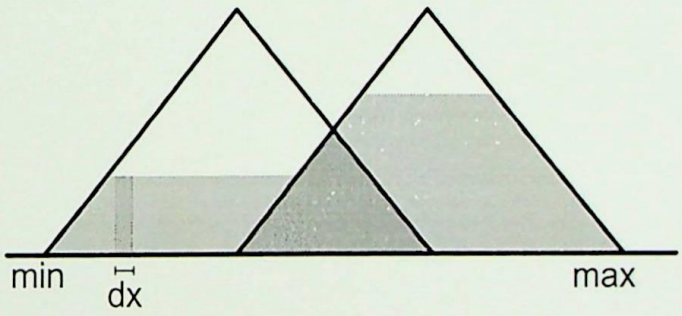



*Figura 2.15 – Geração por fusão de área*

- **Evento 3.2.3 - Processo de Geração por Máximos Locais**

O algoritmo deste evento segue o modelo do algoritmo comum de geração.

Algoritmo de defuzzyficação por máximos locais;

Similar ao evento 3.2.1. divide-se o polígono retornando o centro de gravidade dos retângulos encontrados, conforme a figura 2.16.

Acumula-se este valor tirando a média ponderada com os demais.

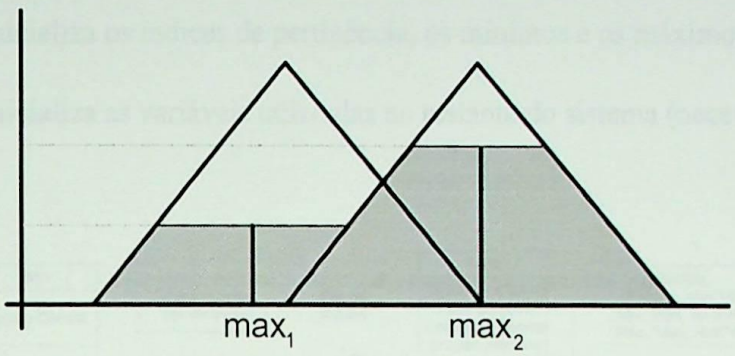



*figura 2.16 – Processo de geração por máximo local*

- **Evento 3.2.4 - Processo de Geração por Máximo Global**

O algoritmo deste evento segue o modelo do algoritmo comum de geração.

Algoritmo de defuzzyficação por máximos global;

Também similar ao evento 3.2.1. divide-se o polígono retornando o centro de gravidade do retângulo com maior índice de pertinência, como mostra a figura 2.17.

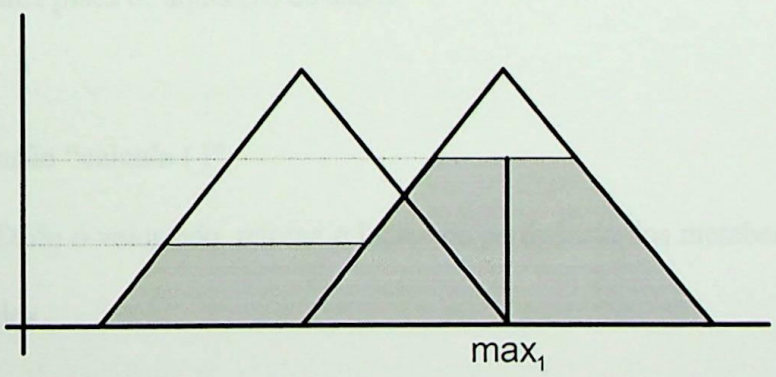



*figura 2.17 – Processo de geração por máximo global*

- **Função "inicializa ( )"**
  - O algoritmo segue o DFD da figura 2.18.
  - Inicializa os índices de pertinência, os mínimos e os máximos com zero
  - Inicializa as variáveis utilizadas no restante do sistema (necessário em C)

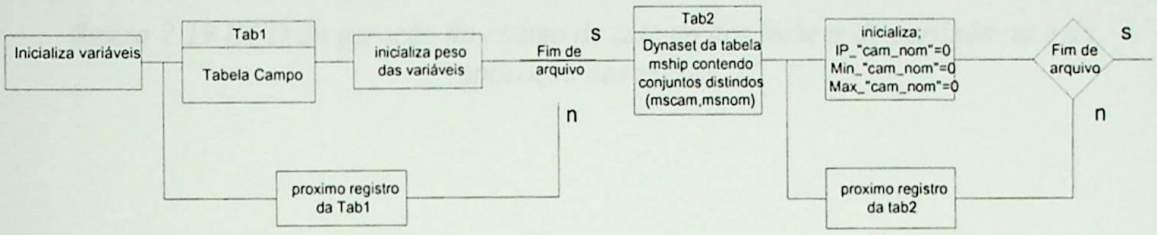



*figura 2.18 – DFD do processo de inicialização*

- **Função "levariaveis ( )"**
  - Faz a entrada de dados seja via teclado ou placa de aquisição de dados

*obs.: Neste módulo existe uma função usuário, ou seja uma função em um arquivo anexo que é acessada pelo programa. Esta função, a "leportaana()" retorna um valor lido em uma placa de aquisição de dados.

- **Função "calcula ( )"**
  - Dado o valor lido, retorna o índice de pertinência dos memberships associados.

- **Função "aplica ( )"**
  - Aplica as regras definidas pelo usuário, algoritmo mostrado na figura 2.19.

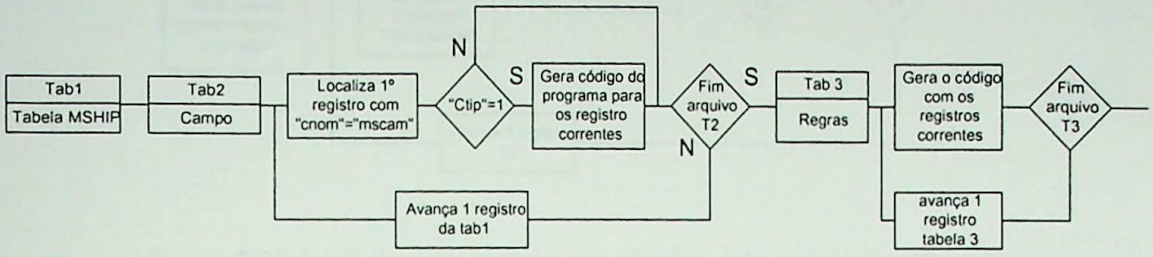



*figura 2.19 DFD da geração do código de cálculo dos índices de pertinência e da aplicação das regras*

- **Função "defuzzy1 ( )"**

Método de defuzzyficação por superposição de áreas

- **Função "defuzzy2 ( )"**

Método de defuzzyficação por fusão de áreas

- **Função "defuzzy3 ( )"**

Método de defuzzyficação por máximos locais.

- **Função "defuzzy4 ( )"**

Método de defuzzyficação por máximos globais.

A figura 2.20 ilustra o processo geral da defuzzificação.

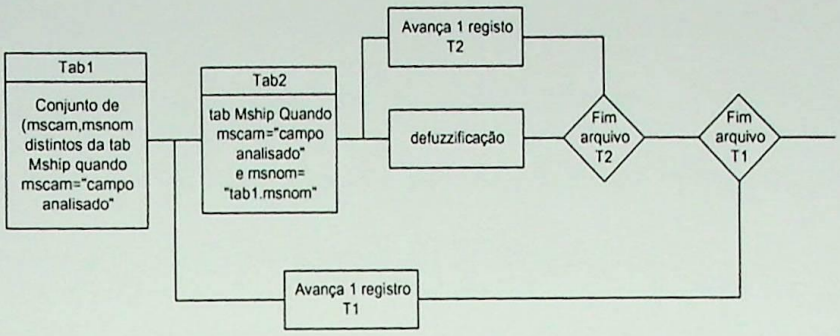



*figura 2.20 DFD do processo de geração do código de defuzzificação*

# 3

# Utilização do Compilador Fuzzy

*Após a implementação do software surge uma nova fase; sua utilização, ou seja, como se processa a entrada dos dados, dos campos e das regras, a escolha do método de defuzzificação e da forma de geração, a definição e utilização de funções externas criadas pelo usuário, a manipulação de arquivos , formas de edição e os métodos de utilização do código gerado.*

## 3.1. Introdução Sobre Compilador Fuzzy

Confirmando a análise do capítulo anterior, apesar de ser possível não seguir uma ordem fixa de desenvolvimento e entrada de dados é aconselhável para o usuário seguir um roteiro de desenvolvimento para evitar inconsistência dos dados o que levaria a problemas na geração do código. Lembramos que é necessário ao usuário além de um conhecimento básico em fuzzy, e para um aproveitamento completo do sistema, um conhecimento aprofundado em linguagem C ou MatLab®, já que é possível para o usuário implementar funções externas ou utilizar as funções já existentes nestes ambientes de desenvolvimento.

É fundamental ter em mente que para um bom aproveitamento da lógica difusa é necessário um bom dimensionamento dos campos de pertinência e das regras, pois é justamente nesta etapa que se introduz o conhecimento que o operador tem do problema para

o compilador, logo qualquer erro de dimensionamento ou falha na lógica de resolução do problema pode levar o compilador a respostas insatisfatórias para o problema em questão.

Para facilitar a exibição desta fase, será implementado um exemplo prático de utilização do compilador, mostrando desde a tela de abertura (figura 3.1) até a utilização em um ambiente de desenvolvimento, no caso o Matlab®. O exemplo proposto, um controle de temperatura de um ambiente fechado, possuiu como entradas a leitura de temperatura e umidade feitas por sensores e como elemento atuador terá um conjunto ar-condicionado/aquecedor para manter a temperatura estável no ponto de referência.

**Dados da simulação:**

**Grandeza:** Temperatura, umidade, ajuste

**Nome atribuído:** temp_1, umi_1,aj_1

**Limites:** 0 < temp_1 < 45 ; 10 < umi_1 < 90 ; -100 < aj_1 < 100

**Pesos:** ambos unitários

**Porta:** Simulação não utilizando placa de aquisição de dados.

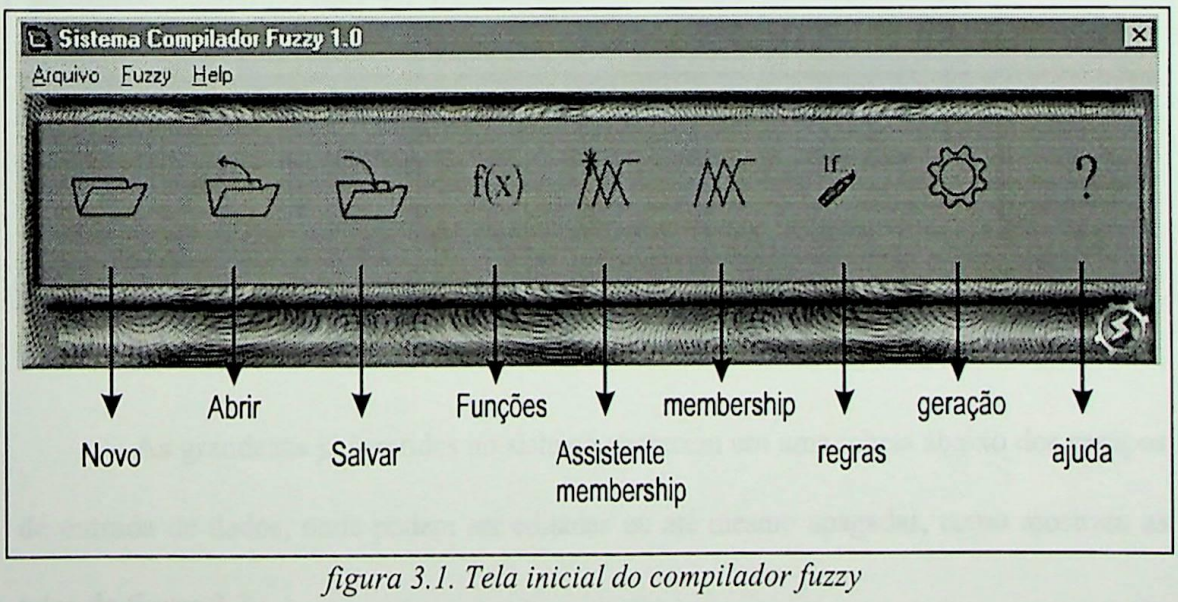



*figura 3.1. Tela inicial do compilador fuzzy*

## 3.2. Entrada de Dados

Como visto anteriormente, são três os módulos de entrada de dados. Entrada das grandezas de atuação, dos campos de pertinência e das regras de atuação. Todos podem ser acessados diretamente pelos ícones situados na tela inicial do compilador.

### 3.2.1 Entrada das grandezas de atuação

Antes de dar a entrada dos dados, deve-se primeiro gerar um novo arquivo de dados (figura 3.2) . Feito isto dá-se a entrada dos dados necessários às grandezas que são armazenadas no banco de dados mencionado no capítulo 2.

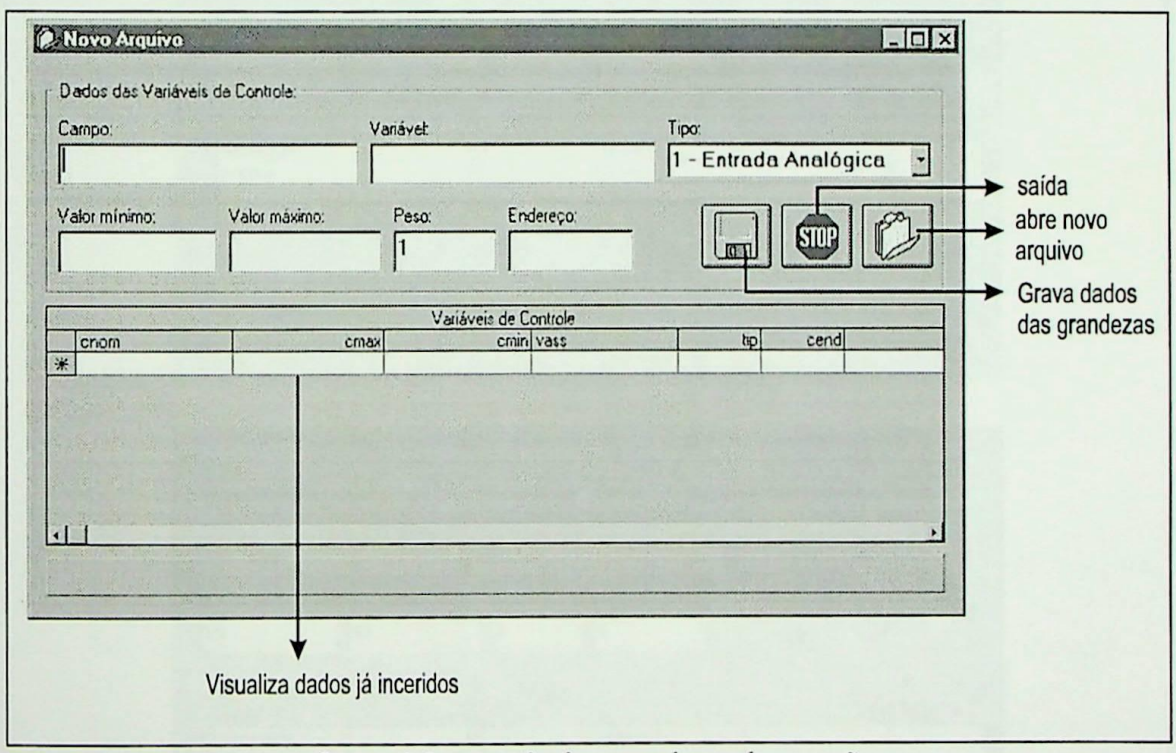



*figura 3.2. entrada das grandezas de atuação*

As grandezas já inseridas no sistema aparecem em uma tabela abaixo dos campos de entrada de dados, onde podem ser editadas ou até mesmo apagadas, como mostram as telas da figura 3.3.

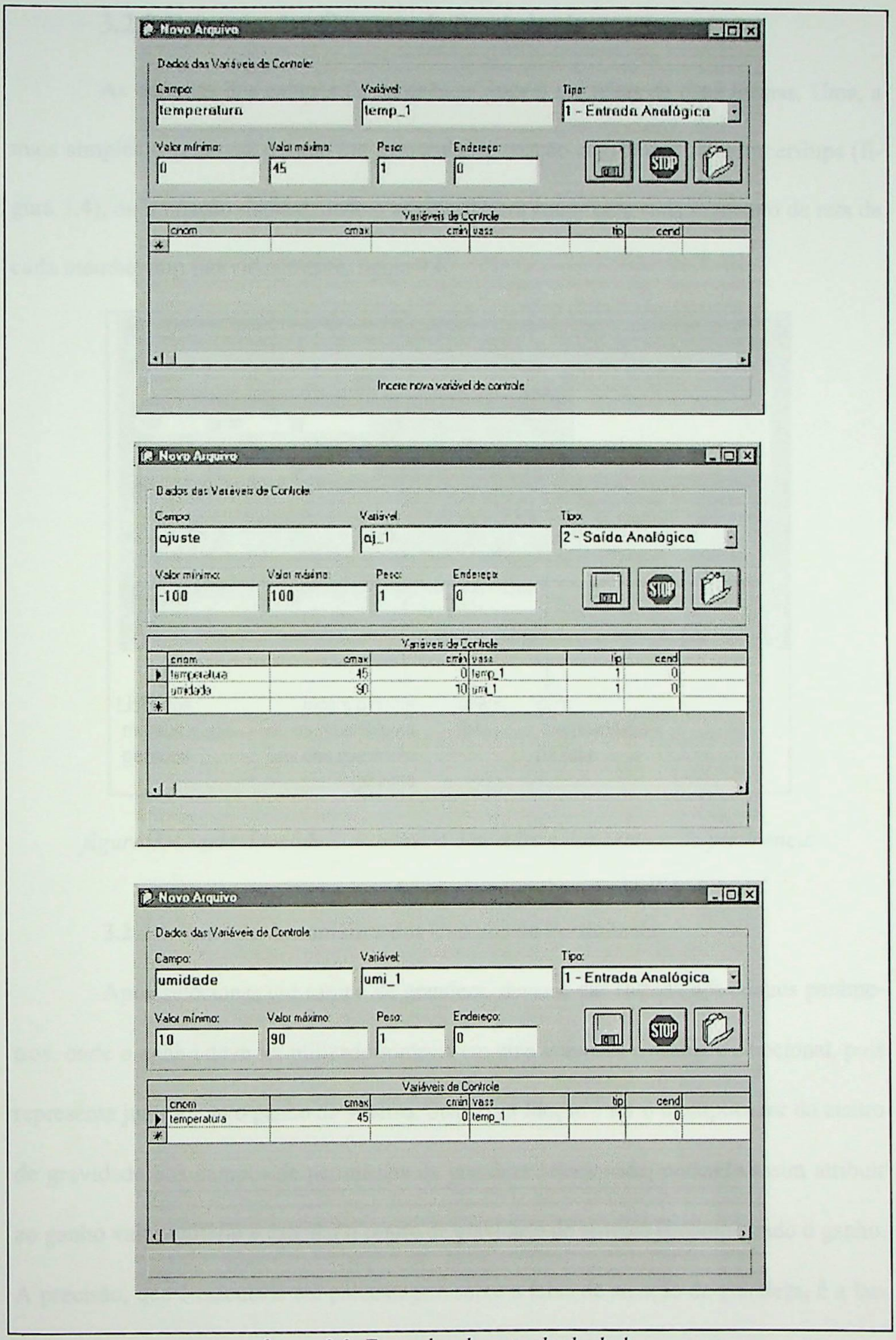



*figura 3.3. Exemplos de entrada de dados*

### 3.2.2 Entrada dos Campos de Pertinência

As entradas dos campos de pertinência podem ser feitas de duas formas. Uma, a mais simples e genérica, usando um assistente de criação automática de memberships (figura 3.4), ou a criação manual, onde o usuário deverá entrar com cada segmento de reta de cada membership individualmente, figura 3.6.

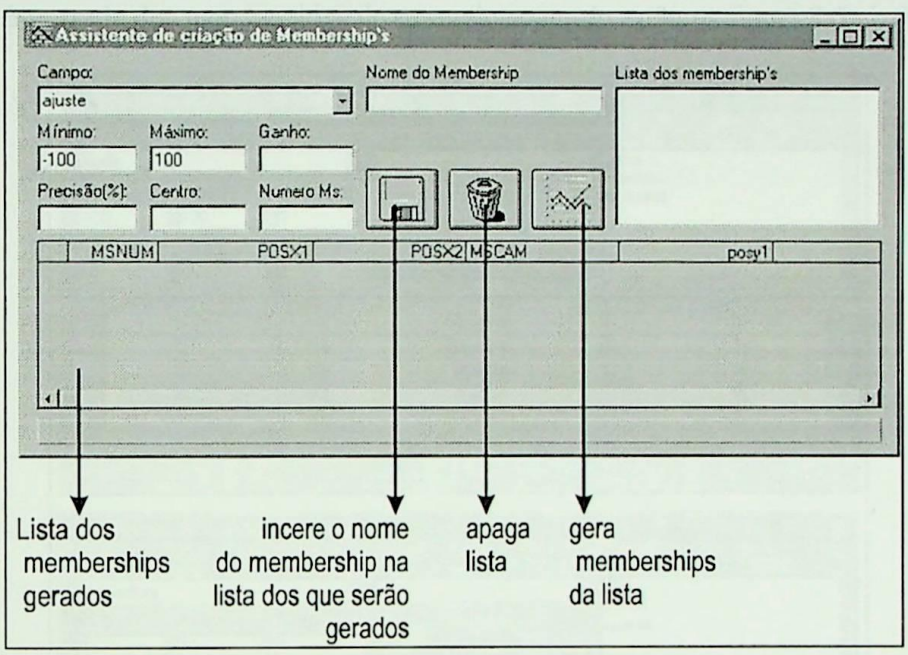



*figura 3.4 - tela do módulo de entrada automática dos campos de pertinência*

#### 3.2.2.1 Entrada Automática dos Campos de Pertinência

Após selecionar um campo de grandeza, deve-se dar entrada nos demais parâmetros, onde o ganho deve ser utilizado somente em uma grandeza de saída e é opcional, pois representa justamente o ganho do sistema. Sua única função é ser o multiplicador do centro de gravidade dos campos de pertinência da grandeza selecionada, podendo assim atribuir ao ganho valor unitário e calcular o centro de gravidade do sistema já computando o ganho. A precisão, que é calculada em porcentagem sobre a faixa de atuação da grandeza, é a largura inferior do campo de pertinência. O centro é o centro de gravidade do sistema e o nú-

mero de memberships é a quantidade de memberships a esquerda ou a direita do centro, sendo assim para um total de sete memberships, deve-se dar uma entrada de três.

Após os ajustes dos parâmetros, é necessário dar entrada nos nomes dos campos de pertinência que serão utilizados, para isso é preciso entrar individualmente com os nomes e carregá-los na respectiva lista.

A entrada dos memberships do exemplo sugerido estão na figura 3.5 abaixo.

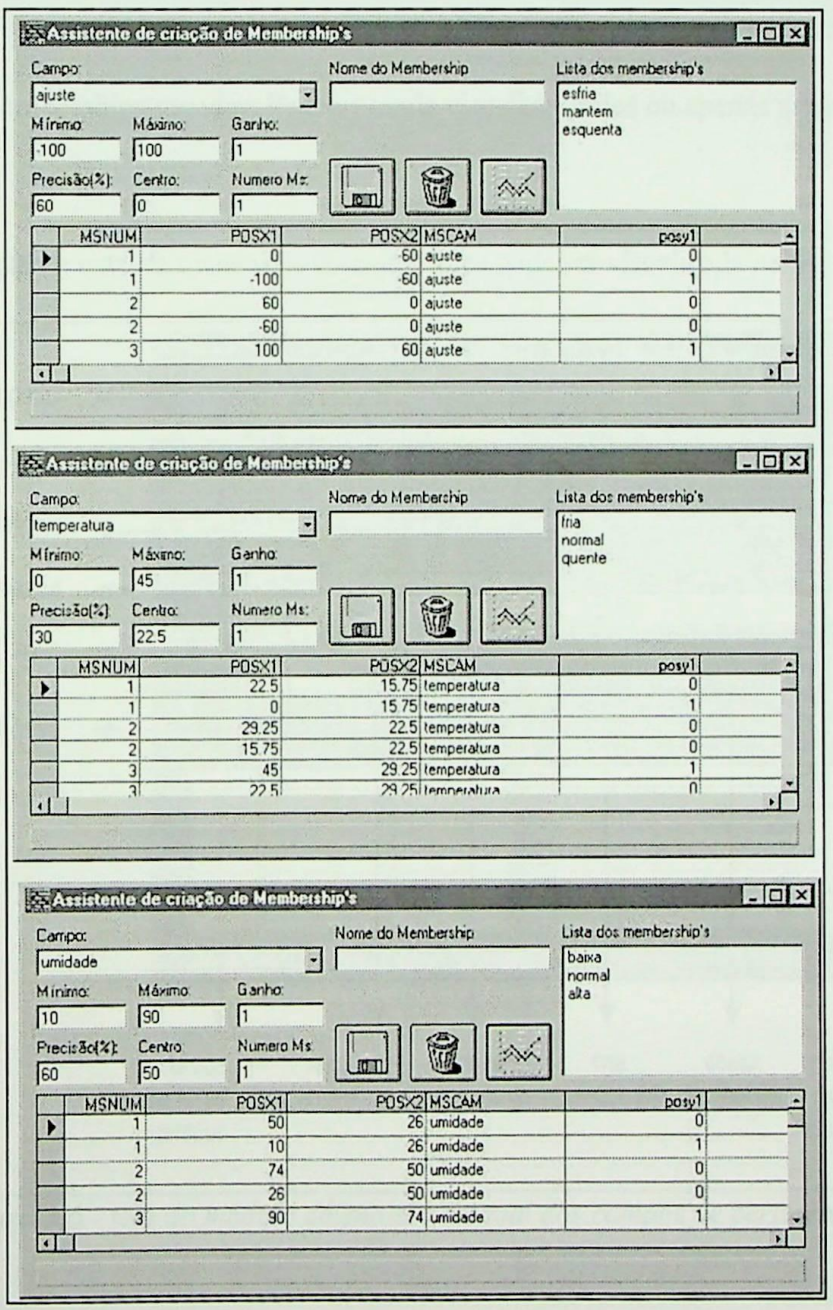



*figura 3.5. Exemplos de entrada automática de memberships*

### 3.2.2.2 Entrada Manual dos Campos de Pertinência

A entrada manual de dados é feita sob a forma de segmentos de reta, porém deve-se primeiro, abrir um novo membership e designar um nome para então dar a entrada nos respectivos segmentos.

Existem duas formas de dar entrada nos segmentos, a primeira é ajustar as reguas nos valores desejados e criar o segmento, ou dar entrada manualmente nos valores das reguas e novamente criar o segmento. E pode-se alterar qualquer segmento diretamente pela tabela dos memberships gerados. Pode-se ainda visualizar todos ou apenas um membership gerado.

A tela de entrada manual dos memberships pode ser visualizada na figura 3.6.

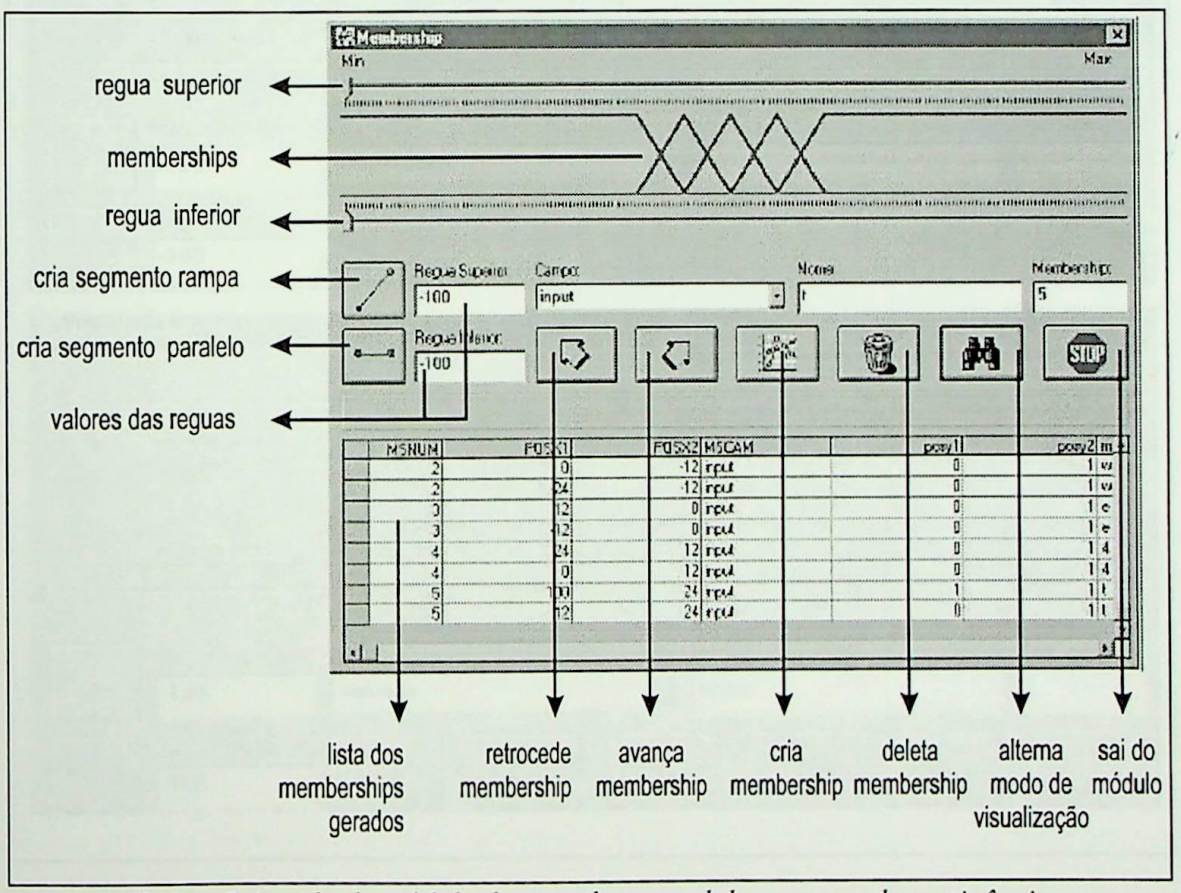



*figura 3.6 - tela do módulo de entrada manual dos campos de pertinência*

Para o exemplo proposto no início do capítulo, pode-se visualizar os memberships gerados no tópico 3.2.2.1 na figura 3.7.

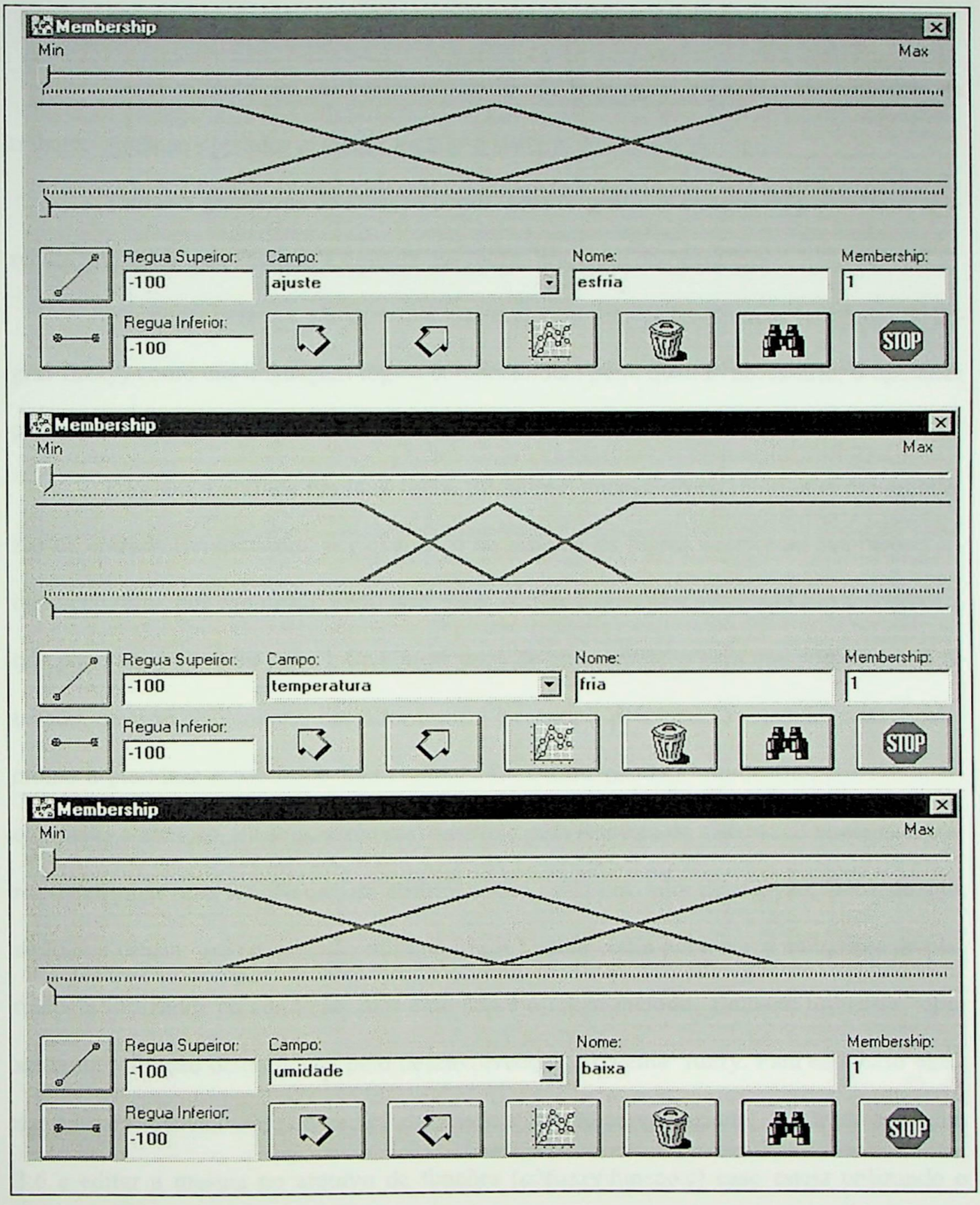



*figura 3.7 - Campos de pertinência gerados pelo módulo de criação automático*

### 3.2.3 Entrada das Regras

Nesse módulo um conhecimento maior da linguagem específica do ambiente de trabalho ajuda ao operador entender melhor o sistema de entrada de regras.

Tanto as grandezas de atuação como seus respectivos campos de pertinência aparecem presentes para facilitar a entrada das regras por parte do operador.

As regras geradas, são apresentadas tanto em termos lingüisticos (conjunto de regras fuzzy) como em linguagem lógica computacional para, quando necessário, o operador poder fazer entrada de regras utilizando alguma função específica.

Para dar a entrada em uma regra, primeiro é necessário escolher uma das grandezas de entrada (temperatura, por exemplo) no sistema da lógica fuzzy com sua respectiva condição (alta, por exemplo). Feito isso, seleciona-se o operador associado a esta condição, que pode ser [ = = ] ou [ not ]. Se houver mais de uma condição para que possa haver a atuação, deve-se selecionar o operador [ and ] e repetir o processo. O segundo passo é selecionar a grandeza e sua respectiva condição (ar condicionado e muito frio, por exemplo) que farão a atuação, ou seja, serão responsáveis pela resposta do sistema. É ainda necessário determinar uma função que irá atribuir a essa saída um valor de atuação. Normalmente na lógica difusa usa-se a função mínimo [ min ] tendo como parâmetros os índices de pertinência utilizados na condição, mas esse não é o único método. Também utiliza-se como saída uma função determinada pelo desenvolvedor do sistema fuzzy. Para esse caso basta dar entrada manualmente no campo destinado a função computacional, mostrado na figura 3.6 e editar a mesma no arquivo de funções (c:\fuzzy\funcao.c) caso esteja utilizando o ambiente C ou editar no próprio Matlab®.

Ex.: Regra: Se temperatura for quente e a umidade for alta

então ajuste o controle para esfria

Para o exemplo da seção 1.1 foram criadas um conjunto de cinco regras totalmente heurísticas baseadas somente em um sentimento de como um controle de temperatura baseado sob influência da umidade. Sendo assim é de fundamental importância que para o desenvolvimento das regras o operador tenha um conhecimento razoável de como se funciona o processo. Outro fator importante para o bom funcionamento do projeto é que todo o universo de acontecimentos esteja presente nas regras, caso contrário irá existir uma situação na qual não existirá sequer uma regra de atuação, ou seja, o sistema irá gerar uma resposta nula.

Regras editadas para o exemplo corrente:

*Se temperatura = = quente, então ajuste = esfria*
*Se temperatura == fria, então ajuste = esquenta*
*Se temperatura = = normal e umidade = = alta então ajuste = esfria*
*Se temperatura = = normal e umidade = = normal então ajuste = mantêm*
*Se temperatura = = normal e umidade = = baixa então ajuste = esquenta*

Neste exemplo as grandezas caracterizadoras de condição são a temperatura e a umidade, e a de atuação é o ajuste do controle.

A tela de entrada de regras pode ser visualizada na figura 3.8.

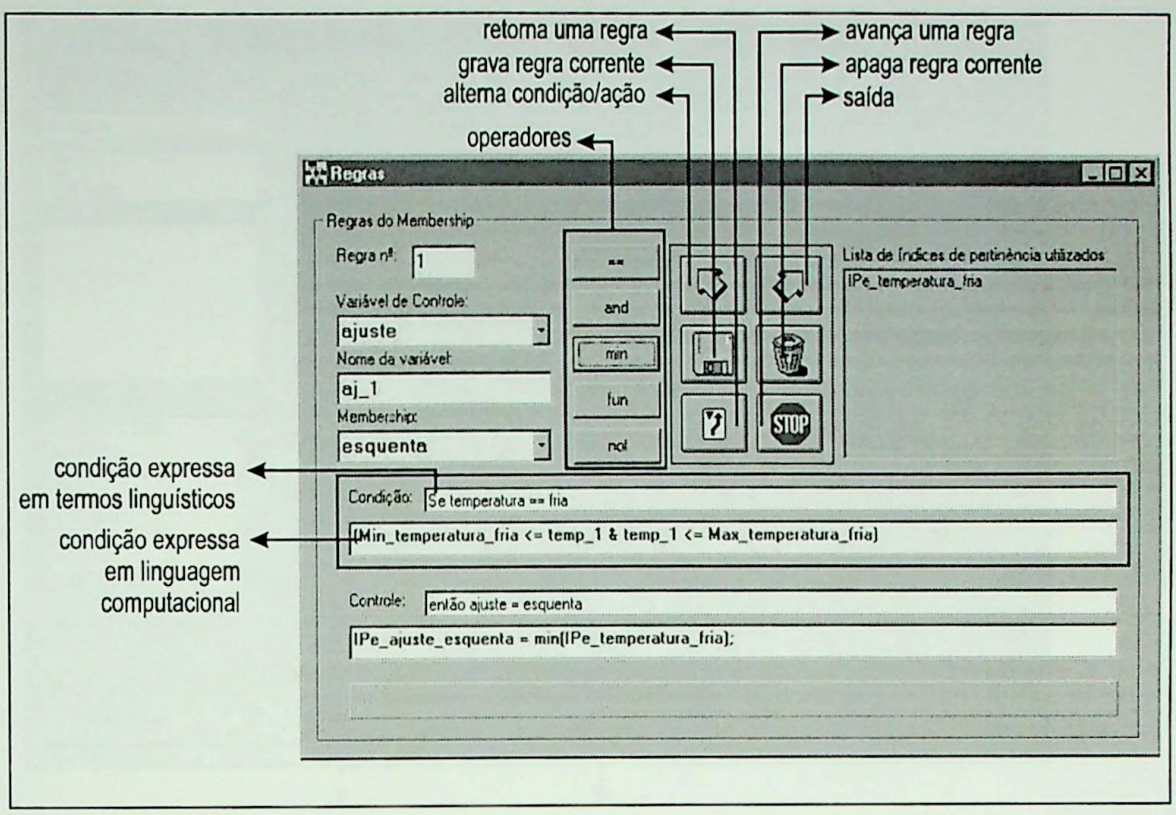



*figura 3.8 - tela de entrada de regras*

## 3.3 Geração do Código

É o final do processo, primeiro escolhe-se o objeto de geração (linguagem C com entrada manual de dados, linguagem C com placa de aquisição de dados, arquivo Matlab® e arquivo Matlab® com ajuste automático de ganho) e, em seguida, ao definir o processo de defuzzificação inicia-se, automaticamente, a geração do código. Pode-se visualizar o código gerado pela janela lateral (figura 3.9). Por fim, define-se um nome e o diretório em que o arquivo gerado será armazenado.

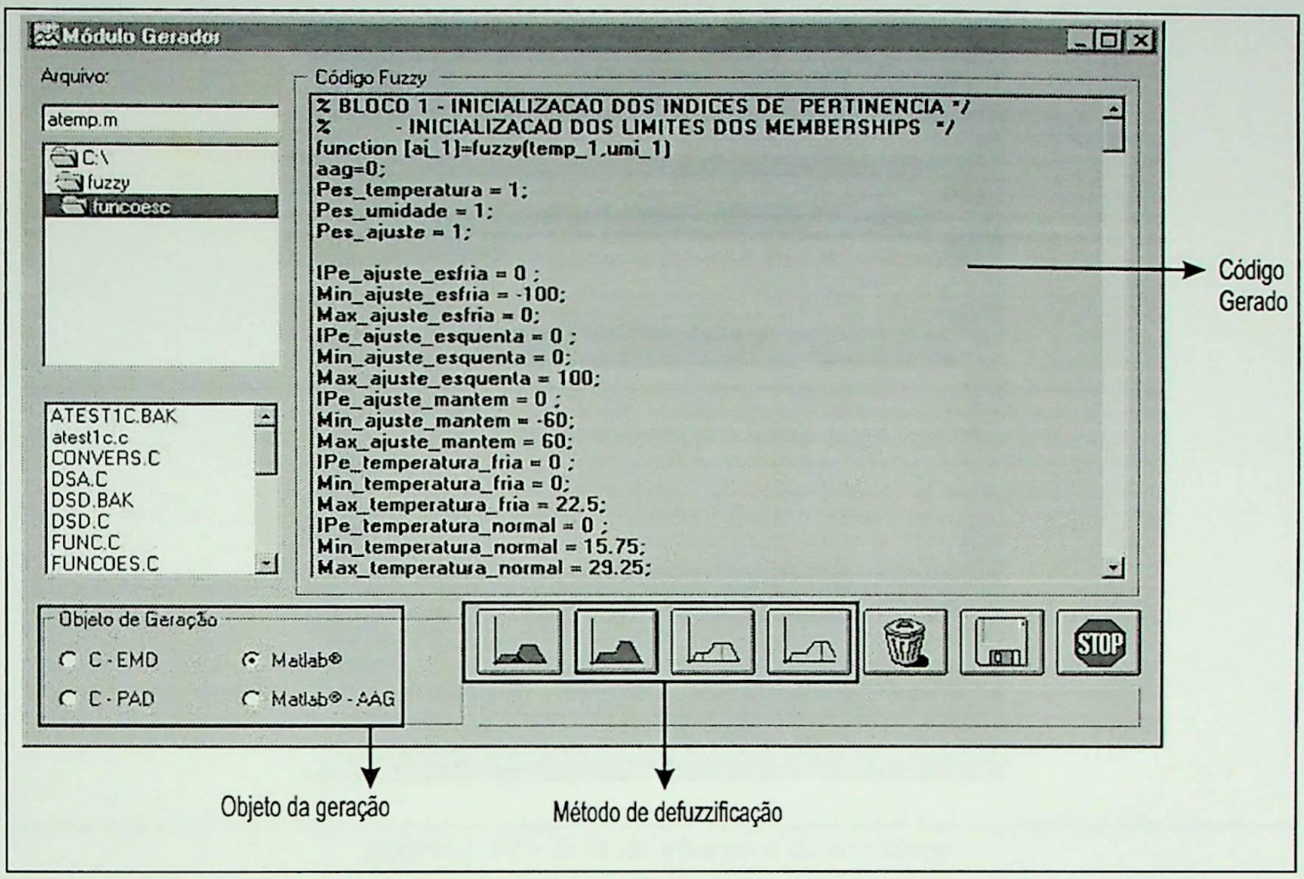



*figura 3.9 - Tela da geração do código*

## 3.4. Manipulação de Arquivos

Consiste em salvar ou recuperar um arquivo de trabalho. Como visto no capítulo anterior, o compilador não gera um arquivo de banco de dados, mas sim sobrepõe um arquivo em branco já existente para o arquivo de trabalho (fuzzy.mdb).

As telas de abertura ou armazenamento de arquivos encontram-se ilustradas nas figuras 3.10 e 3.11 respectivamente.

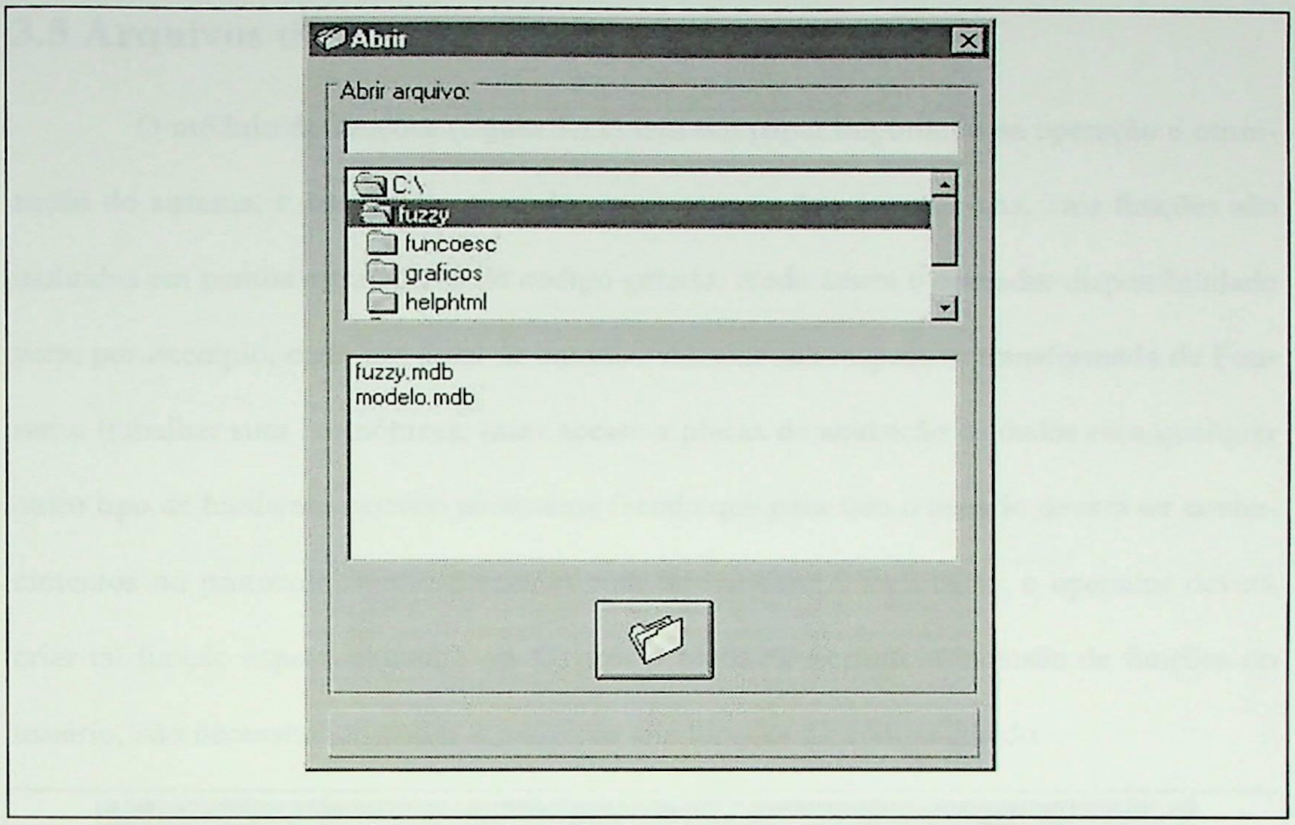



*figura 3.10 - Tela de abertura de arquivos*

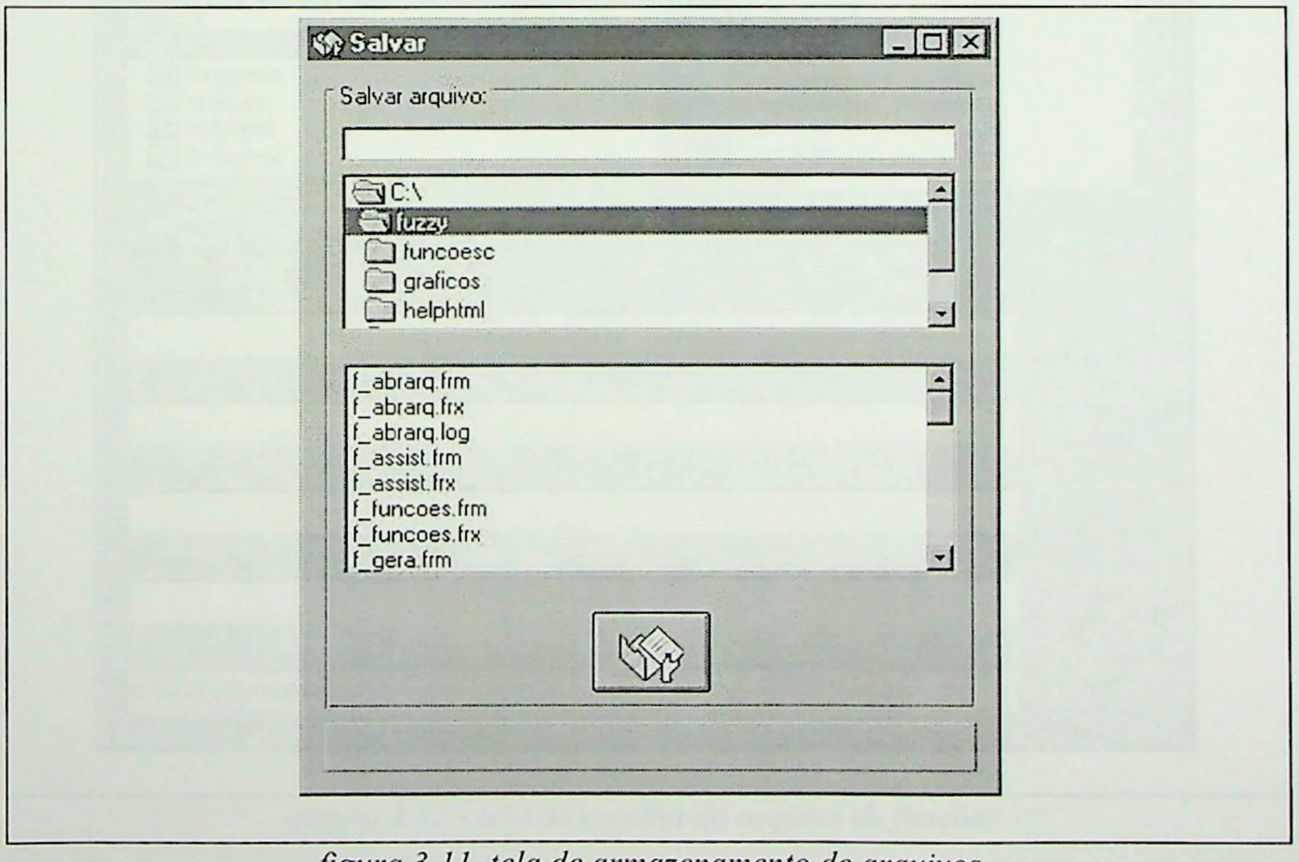



*figura 3.11. tela de armazenamento de arquivos*

## 3.5 Arquivos de funções

O módulo de funções (figura 3.12) tem um papel importante na operação e otimização do sistema, é aqui que o operador entra com as funções externas. Tais funções são incluídas em pontos estratégicos do código gerado, tendo assim o operador disponibilidade para, por exemplo, dado um sinal de entrada, calcular sua respectiva transformada de Fourier e trabalhar suas harmônicas, fazer acesso a placas de aquisição de dados ou a qualquer outro tipo de hardware externo ao sistema (sendo que para isso o usuário deverá ter conhecimentos no protocolo de comunicação com tal harware) . Para tanto, o operador deverá criar tal função especificamente em C, pois o Matlab® permite a inclusão de funções do usuário, não necessitando assim acrescentar tais funções no código gerado.

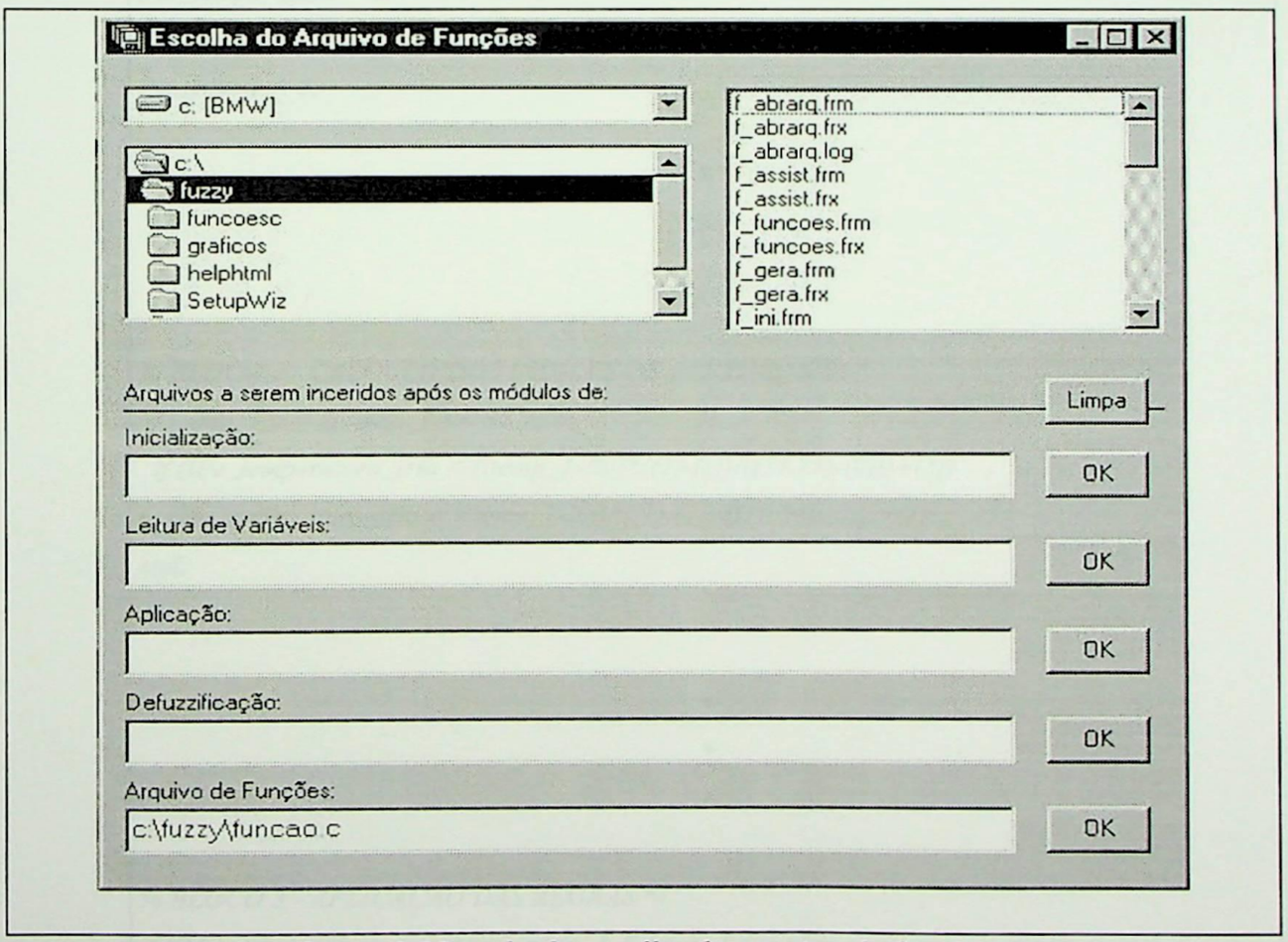



*figura 3.12 - tela de escolha do arquivo de funções*

## 3.6. Utilizando o código gerado

A utilização do código depende unicamente do conhecimento do operador na utilização do ambiente de desenvolvimento utilizado. São muitas as possibilidades de utilização, ficando assim inviável o detalhamento de cada uma. No capítulo 4 será mostrado os testes realizados com os códigos do exemplo deste capítulo e alguns exemplos extras que serão ainda descritos. A seguir tabela 3.1 é mostrado seções do código do exemplo entrado nesse capítulo. O código inteiro é mostrado no apêndice B.

```
% BLOCO 1 - INICIALIZACAO DOS INDICES DE  PERTINENCIA */
%        - INICIALIZACAO DOS LIMITES DOS MEMBERSHIPS  */
function [aj_1]=fuzzy(temp_1,umi_1)
aag=0;
Pes_temperatura = 1;
Pes_umidade = 1;
Pes_ajuste = 1;

                                        *
                                        *
                                        *

% BLOCO 2 - CALCULO DOS INDICES DE PERTINENCIA */
 if (temp_1 > 0 & temp_1 <= 15.75 )
  if (IPe_temperatura_fria < ((temp_1-(0))*((1)-(1))/((15.75)-(0)))+(1))
    IPe_temperatura_fria = ((temp_1-(0))*((1)-(1))/((15.75)-(0)))+(1);
end;
end;

                                        *
                                        *
                                        *

% BLOCO 3 - APLICACAO DAS REGRAS */
if ((Min_temperatura_quente <= temp_1 & temp_1 <= Max_temperatura_quente))
matriz=[IPe_temperatura_quente];
```

```
retorno= min(matriz);
IPe_ajuste_esfria = max(IPe_ajuste_esfria ,retorno);
end;

                                        *
                                        *
                                        *

% INICIO DO BLOCO 4 - DEFUZZYFICACAO*/
xx = 0;
xm = 0;
xmin =99999;
xmax =-99999;
x1 =-100;
x2 =-60;
indice= IPe_ajuste_esfria;
if (indice > 0)
xmin=min(xmin, Min_ajuste_esfria);
xmax=max(xmax,Max_ajuste_esfria);
end

                                        *
                                        *
                                        *

flag=0;
for b=1:a-1
if (matriz(b,2)>xm/2 & flag==0)
xx=matriz(b,1);
flag=1;
end;
end;
aj_1=xx+aag;
```

*tabela 3.1 - listagem parcial do código do exemplo corrente*

# 4

# Exemplos da utilização do compilador fuzzy

*Depois da geração do código pelo compilador, vem a fase final do projeto; sua utilização. Tal utilização depende principalmente do conhecimento do ambiente de trabalho devido a diversidade de opções possíveis para a uso do código. A apresentação deste capítulo será dada através de exemplos exibidos da seguinte forma: diferenciação dos métodos de defuzzificação e um controlador de temperatura. Também será apresentado uma nova metodologia de fuzzy adaptativo e, por fim, a conclusão do capítulo.*

## 4.1. Diferença prática entre os processos de defuzzificação

Como visto anteriormente (capítulo 2 evento 3.2.1 a 3.2.4) são quatro os processos de defuzzyficação disponíveis pelo compilador: centróide com superposição de área, centróide sem superposição de área, máximos locais e máximo global. Cada um possui uma forma de atuação que será demonstrada no exemplo abaixo.

**Proposta:**

Uma entrada, conforme ilustrada na figura 4.2, será aplicada a quatro sistemas de lógica fuzzy (figura 4.1) referentes aos quatro tipos diferentes de defuzzificação: defuzzificação por áreas sobrepostas (figura 4.3 ) por centróide sem superposição (figura 4.4) , por

máximos locais (figura 4.5) e por máximo global (figura 4.6 ). Ainda será gerado um gráfico com a superposição das saídas (figura 4.7) para serem analisadas posteriormente.

Definição dos dados de entrada:

Variável de entrada: erro

Limite: -100 a 100%

Memberships: 7, com centro em 0 e precisão de 20%.

Variável de saída: atuação.

Limite: -100 a 100%

Memberships: 7, com centro em 0 e precisão de 30%.

**Processo de defuzzyficação:**

Será gerado um código para cada forma de defuzzificação.

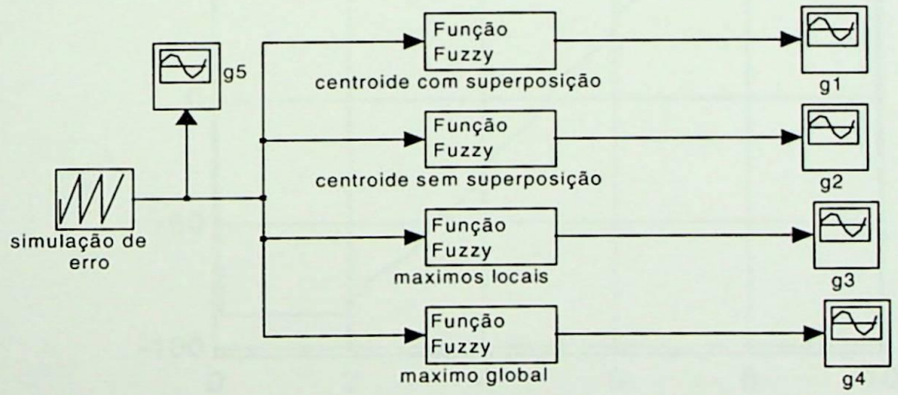



*Figura 4.1 - Esquema de simulação*

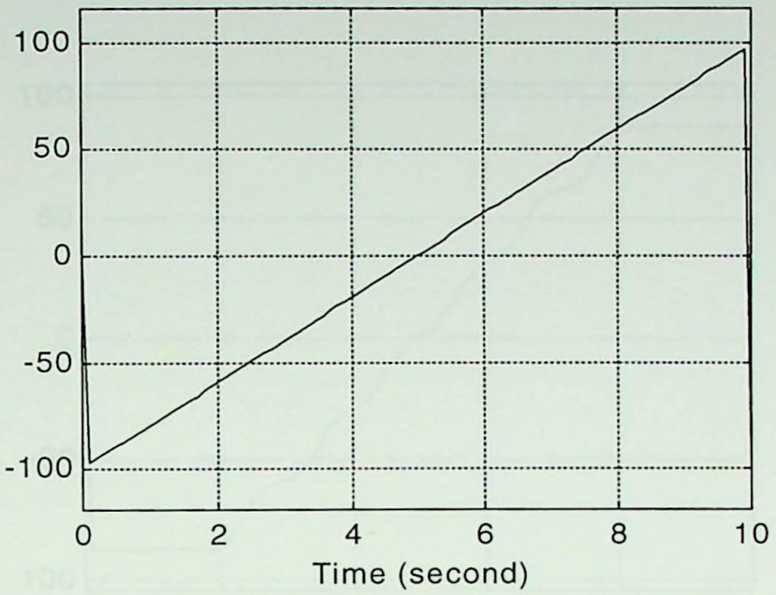



*Figura 4.2 - Sinal de entrada*

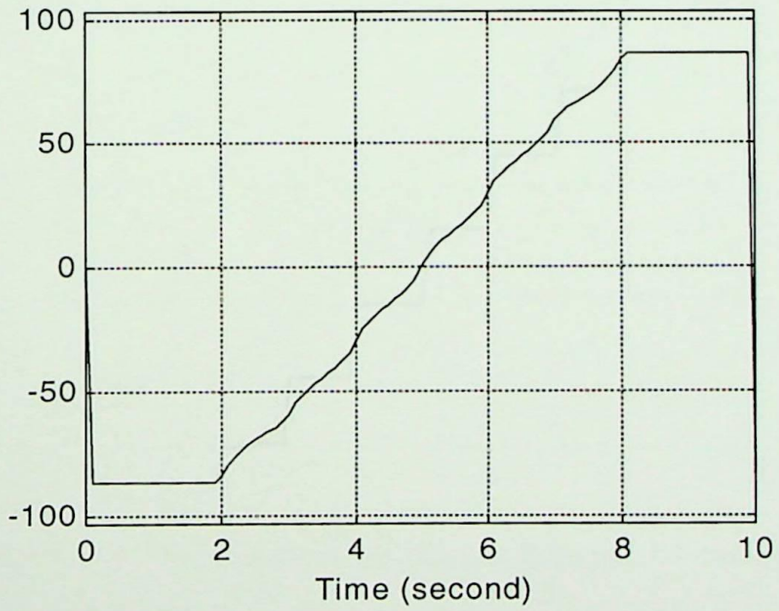



*Figura 4.3 - Resposta da defuzzyficação por áreas sobrepostas*

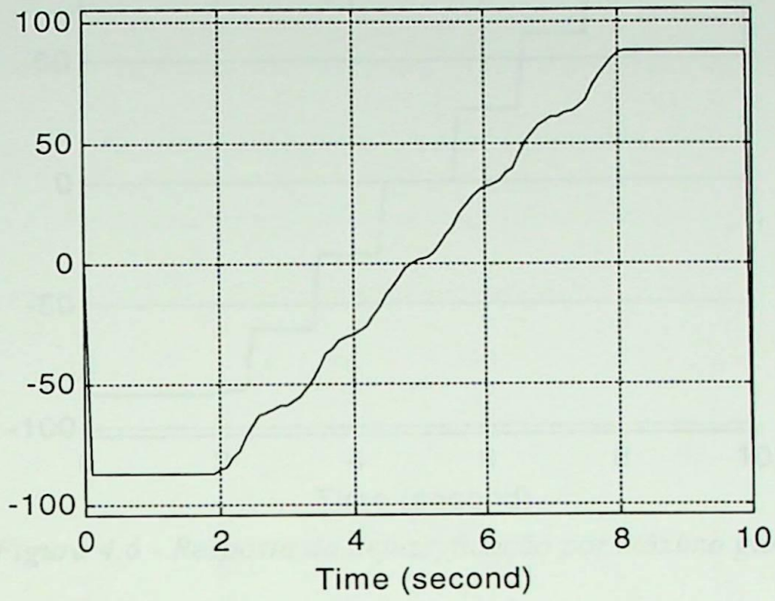



*Figura 4.4 - Resposta da defuzzyficação por centróide sem superposição*

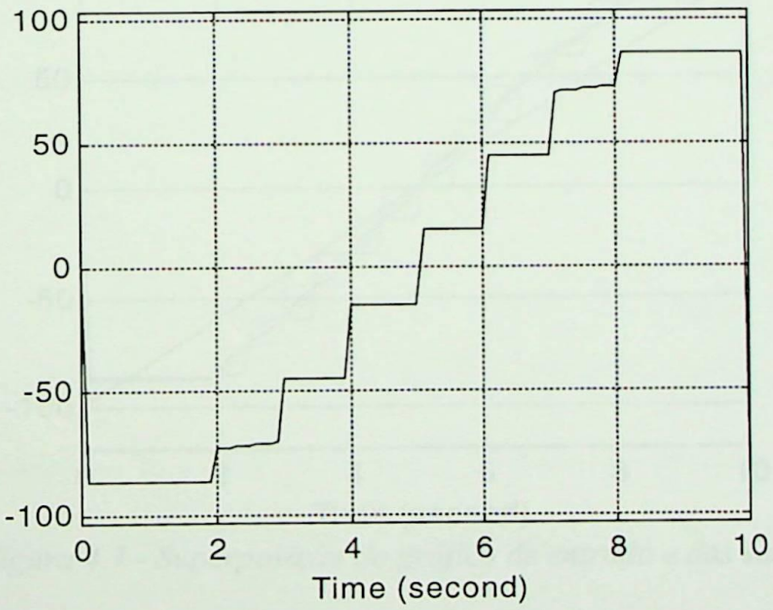



*Figura 4.5 - Resposta da defuzzyficação por máximos locais*

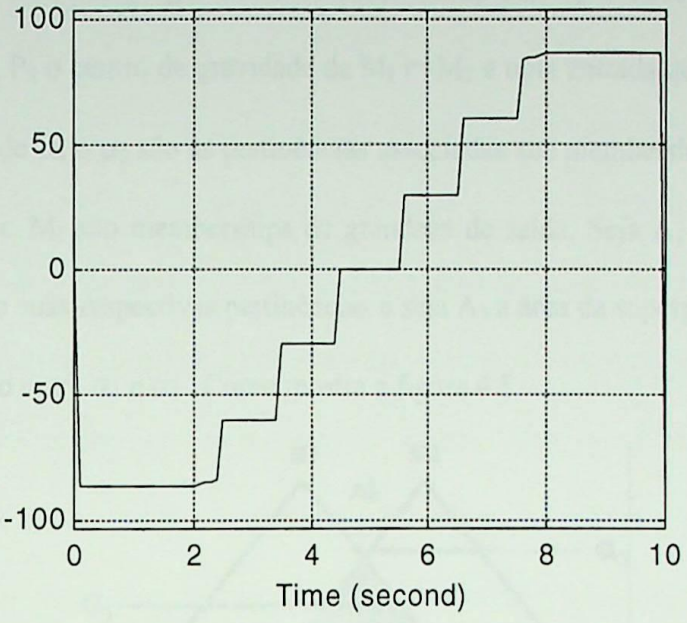



*Figura 4.6 - Resposta da defuzzyficação por máximo global*

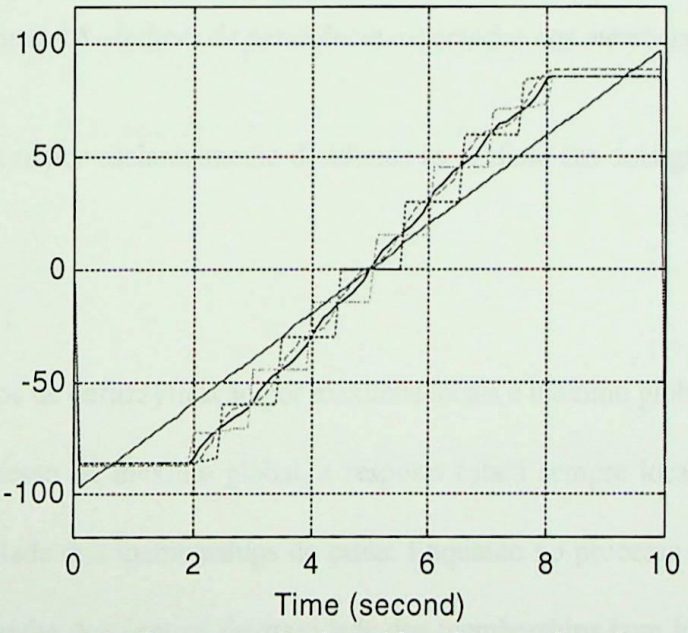



*Figura 4.7 - Superposição do gráfico de entrada e das saídas*

Analisando as respostas obtidas pelos diferentes métodos de defuzzyficação te-

mos;

Seja $P_1$ o centro de gravidade do membership $M_1$, $P_2$ o centro de gravidade do membership $M_2$, $P_3$ o centro de gravidade de $M_1 \cap M_2$ e uma entrada que gere como saída $[\alpha_1.P_1, \alpha_2.P_2]$ onde $\alpha_1$ e $\alpha_2$ são as pertinências associadas aos memberships $P_1$ e $P_2$ respectivamente e $M_1$ e $M_2$ são memberships da grandeza de saída. Seja $A_1$ e $A_2$ as áreas dos memberships sob suas respectivas pertinências e seja $A_3$ a área da superposição de $M_1$ com $M_2$ sob o máximo entre $\alpha_1$ e $\alpha_2$ . Como mostra a figura 4.8.

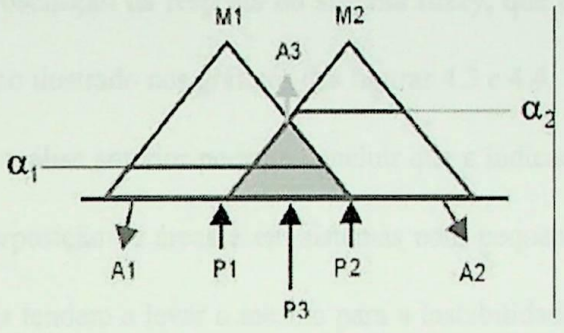





*figura 4.8 –índices de pertinência associados aos memberships*

Para um maior esclarecimento dividimos as análises em dois grupos com características distintas.

Grupo 1;

Processos de defuzzyficação por máximos locais e máximo global.

No processo de máximo global, a resposta estará sempre localizada em um dos centros de gravidade dos memberships de saída. Enquanto no processo máximos locais, a resposta será a média dos centros de gravidade dos memberships com índices de pertinência não nulos.

Grupo 2;

Processos de defuzzyficação pelo centróide com e sem superposição de áreas que tem como função de saída:

$$S = \frac{A_1 \cdot P_1 + A_2 \cdot P_2 + A_3 \cdot P_3}{A_1 + A_2 + A_3} \quad (1)$$

Analisando a figura 4.8 e a fórmula (1) deduz-se que o processo com superposição de áreas, a resposta é maior quando $\alpha_1 < \alpha_2$ e menor para $\alpha_2 < \alpha_1$. O que é natural pois com a superposição de áreas há, uma área extra $A_3$ com centro de gravidade em $P_3$ (que sem a superposição assume um valor nulo) que funciona como um atrator para o próprio $P_3$. Isso também influencia na oscilação da resposta do sistema fuzzy, que é menor para processo com superposição, como ilustrado nos gráficos das figuras 4.3 e 4.4.

Com base na análise anterior pode-se concluir que a indicação de se utilizar a defuzzificação com superposição de áreas é em sistemas com pequena constante de tempo, onde oscilações rápidas tendem a levar o mesmo para a instabilidade. Estendendo o raciocínio, a indicação para utilização do método sem superposição de áreas seria em sistemas com constantes de tempo elevadas onde é necessário uma rápida atuação para a otimização do desempenho.

## 4.2. Controle genérico utilizando fuzzy adaptativo.

Para sistemas de controle, uma das dificuldades da lógica difusa é saber definir com precisão os valores para os campos de pertinência das variáveis envolvidas, principalmente as da saída, pois é justamente nessas em que estão definidos os pontos de referência do sistema a ser controlado. Tal dificuldade poderia ser contornada sabendo-se a função de transferência do sistema [13] ou levantando os parâmetros do controlador por técnicas como a de Ziegler-Nichols [12], gerando ao operador uma necessidade de possuir um conhecimento aprofundado do sistema. Para facilitar a operação de sistemas cujas fun-

ções de transferências são desconhecidas ou que simplesmente sofram alterações e interferências durante o processo, foi desenvolvido uma metodologia de controle utilizando a já tradicional lógica fuzzy junto com um sistema adaptativo. Isso permite que, segundo um parâmetro de ajuste (item 4.2.1) o sistema tenha a liberdade de alterar o centro de gravidade dos campos de pertinência de saída, deslocando-os para direita ou para esquerda (figura. 4.9) seguindo a necessidade de manter estável o ponto de referência do mesmo, ficando assim confiável e de fácil modelagem.

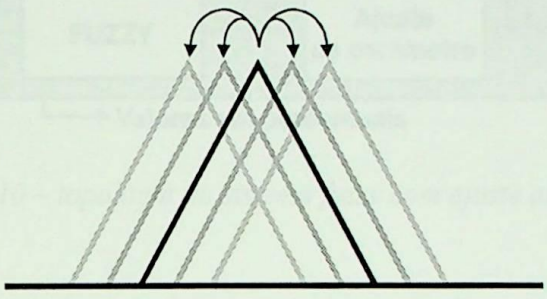


*figura 4.9 – deslocamento do campo de pertinênica.*

Tal metodologia trabalha com parâmetros ATUAÇÃO, ERRO, TENDÊNCIA e AJUSTE, sendo que os três últimos são utilizados tanto no sistema de lógica fuzzy como no sistema adaptativo.

Para uma maior abrangência, ambos os sistemas foram desenvolvidos para trabalharem com valores percentuais. Então para um dado erro, é necessário transformá-lo em porcentagem de erro, variando de -100% a 100%. Para isso é necessário estabelecer um limite de tolerância para o mesmo, ou seja, determinar para quais valores de erro se estipulariam os limites em porcentagem. Para qualquer valor fora do assim chamado limite de tolerância, tanto o inferior quanto o superior é atribuído o respectivo valor de limite, gerando assim erros percentuais na faixa de -100% a 100%. Como os parâmetros TENDÊNCIA e AJUSTE são obtidos a partir do erro, nenhuma conversão é necessária. Para o parâmetro A-

TUAÇÃO faz-se necessário o inverso do processo já que ele é obtido através do erro normalizado.

A topologia do sistema é ilustrada na figura 4.10, abaixo.

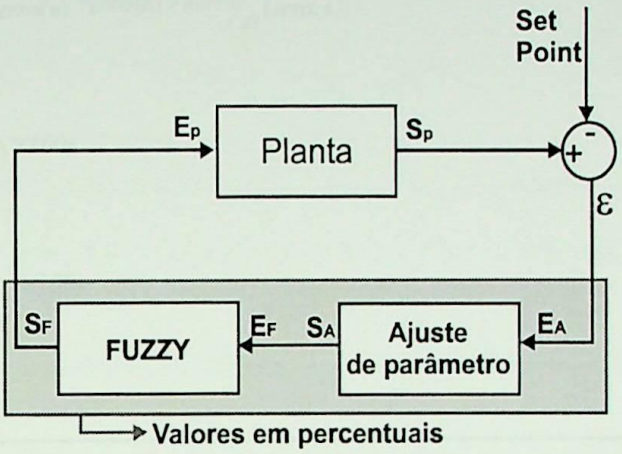



*figura 4.10 – topologia do sistema fuzzy com ajuste adaptativo*

onde;

$S_P$ = Saída da planta

$S_A$ = ajuste

ε = erro

$E_F = [\varepsilon, \text{arctg}\ (d\varepsilon/dt), S_A]$

$E_A = [\varepsilon$ , acumulado, $d\varepsilon/dt]$

$S_F = \text{fuzzy}[\varepsilon, \text{arctg}(d\varepsilon/dt)]+S_A$

### 4.2.1 Detalhamento do Sistema Adaptativo

Obtido o ERRO, este é submetido a uma função que irá determinar se será gerada uma atuação menor, igual ou maior do que o valor atual de erro. Esse valor de atuação é armazenado porque se, na próxima interação o erro for nulo, caracteriza ser este o valor de atuação como o que melhor se adapta ao sistema naquele instante. Tal índice é utilizado para o deslocamento do centro de gravidade dos memberships relacionados do parâmetro ERRO. E é calculado seguindo o seguinte procedimento, tabela 4.1;

```
function[ajuste(n)]= ajustar(erro(n), ∑_{t=0}^{n} erro(n), derro/dt(erro(n)))
if (erro==0)
  ajuste= ∑_{t=0}^{n} erro(n);
else
ajuste= ∑_{t=0}^{n} erro(n) +erro(n)+ derro/dt(erro(n));
end;

if(∑_{t=0}^{n} erro(n) >100)
 ajuste=100;
end;
if(∑_{t=0}^{n} erro(n) <-100)
 ajuste=-100;
end;
```

*tabela 4.1 – função de ajuste adaptativo*

Na tabela 4.1 é visto uma função de ajuste cujos parâmetros são o ERRO, a TENDÊNCIA e o ACUMULADO. Para uma melhor otimização do sistema pode-se utilizar parâmetros multiplicadores no ERRO e na TENDÊNCIA (por exemplo; erro = erro*β). Tanto os parâmetros ERRO como TENDÊNCIA serão também utilizados no sistema de lógica fuzzy, que é a próxima etapa do sistema de controle.

### 4.2.2. Detalhamento do Sistema Fuzzy

O sistema de controle fuzzy foi desenvolvido baseado em um conhecimento empírico de controle para ser genérico [14], isso é, poder ser utilizado, a princípio, em qualquer planta.

Como em qualquer sistema fuzzy a primeira etapa na execução é a escolha das grandezas utilizadas durante o processo e seus respectivos campos de pertinência. Nesse

caso será utilizado como grandezas de entrada o ERRO e a TENDÊNCIA (que é calculada por; arctg(dε/dt)) e, como grandeza de saída a ATUAÇÃO , figuras 4.11,4.12 e 4.13.

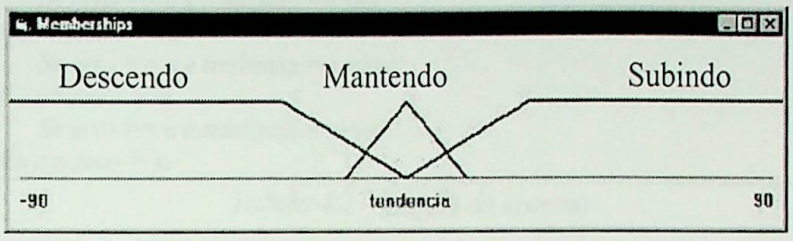



*figura 4.11 – Memberships de tendência(em graus)*

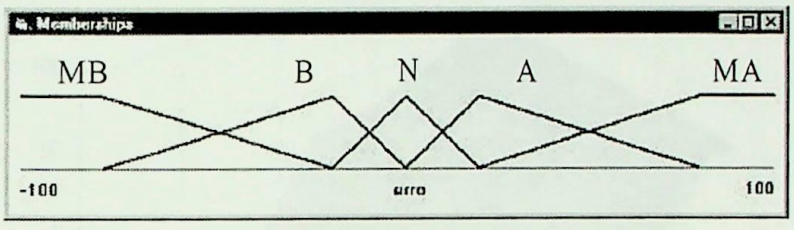



*figura 4.12 – Memberships de erro(em %)*

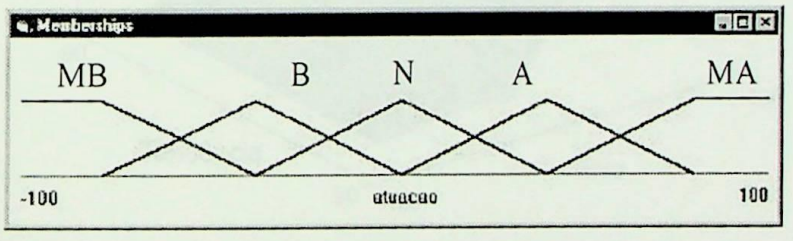



*figura 4.13 – Memberships de atuação (em %)*

A próxima etapa é a definição das regras que devem ser suficientes para abranger todo o universo de atuações possíveis (tabela 4.2).

```
1.      Se erro == mb
entÆo atuacao = mb
2.      Se erro == ma
entÆo atuacao = ma
3.      Se erro == b e tendencia == des
entÆo atuacao = mb
4.      Se erro == b e tendencia == mant
entÆo atuacao = b
5.      Se erro == b e tendencia == sub
entÆo atuacao = m
6.      Se erro == m e tendencia == des
```

| |
|---|
| *entÆo atuacao = b*<br>*7. Se erro == m e tendencia == mant*<br>*entÆo atuacao = m*<br>*8. Se erro == m e tendencia == sub*<br>*entÆo atuacao = a*<br>*9. Se erro == a e tendencia == sub*<br>*entÆo atuacao = ma*<br>*10. Se erro == a e tendencia == mant*<br>*entÆo atuacao = m*<br>*11. Se erro == a e tendencia == des*<br>*entÆo atuacao = m* |

*Tabela 4.2 – regras do sistema*

O sistema acima pode ser visualizado sob a forma gráfica na figura 4.14.

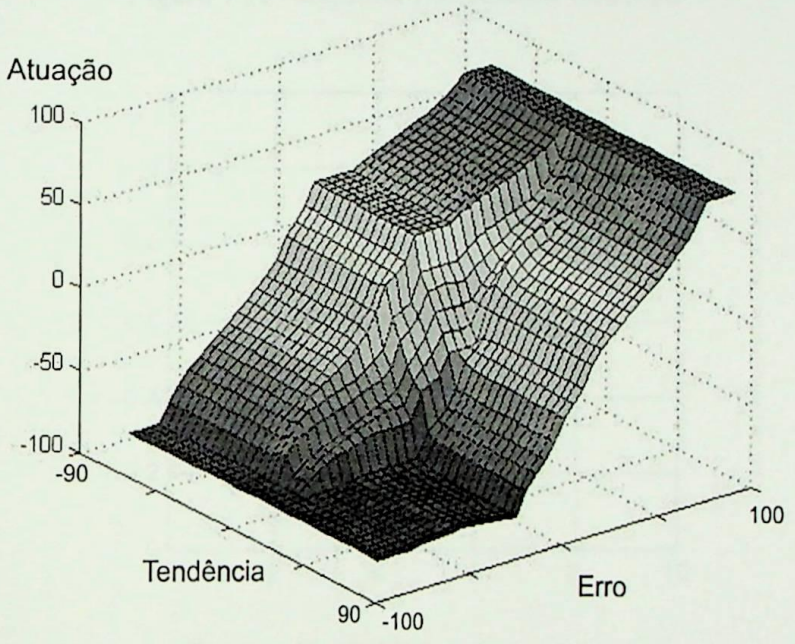



*figura 4.14 – representação gráfica do sistema fuzzy*

### 4.2.3. Exemplos

Os exemplos a seguir seguirão a forma de controle propostas nos itens 4.2.1 e 4.2.2 sendo que o sistema fuzzy gerado possui como forma de defuzzyficação a do centróide com superposição de áreas.

Apesar de se alterar a função de transferência do sistema, a topologia será mantida como a da figura 4.15, que tem como ponto de referência (set point) um valor fixo de 2. Também será jogado no sistema um ruído, como mostra o gráfico da figura 4.16.

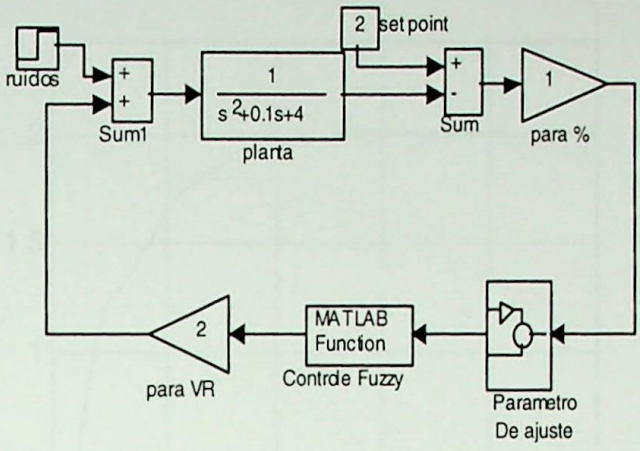



*Figura 4.15 - Esquema 1 do controle com AAG*

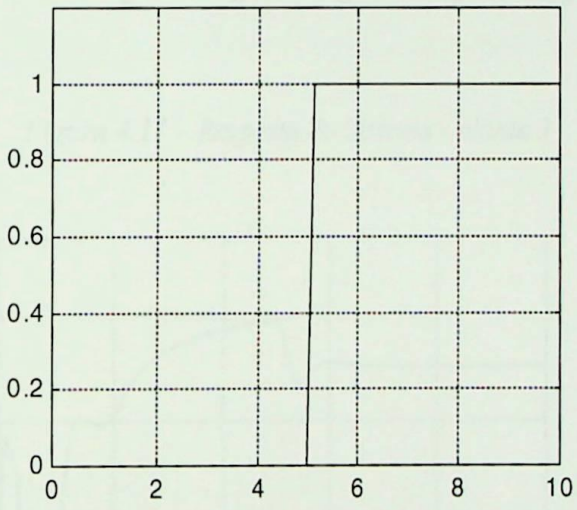



*Figura 4.16 - Ruído jogado no sistema*

### 4.2.3.1 Planta 1

Esta planta esta representada na figura 4.14, com Wn=2; Ke=0,25 e $\zeta$=0,05. Obtendo como respostas as figuras 4.17, 4.18 e 4.19.

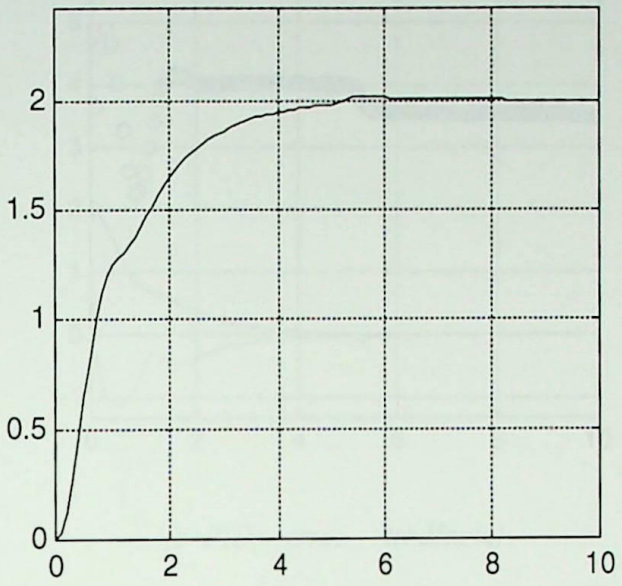



*Figura 4.17 – Resposta do Sistema - planta 1*

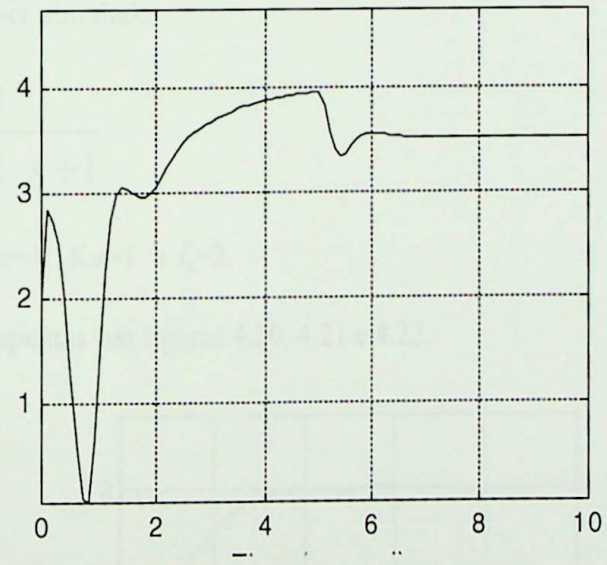
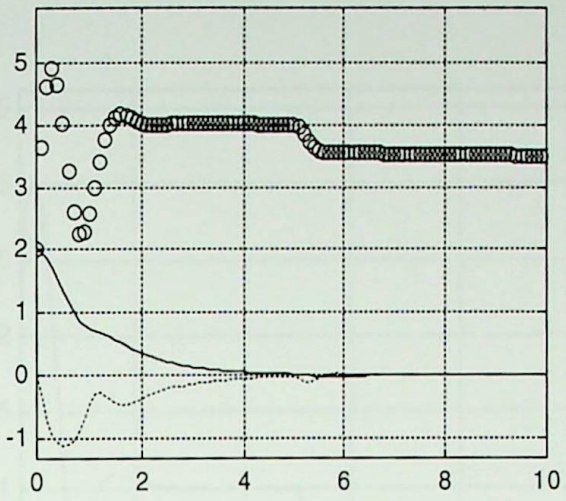



*Figura 4.18 - Saída do Controle Fuzzy – planta 1*




*(o PID; - erro; - - tendência)*

*Figura 4.19 - Saída do Controle Fuzzy – planta 1*

#### 4.2.3.2 Planta 2

Planta a ser simulada:

$$\frac{1}{s^2 + 2 \cdot s + 1}$$

com Wn=1; Ke=1 e $\zeta$=2.

Como respostas nas figuras 4.20, 4.21 e 4.22.

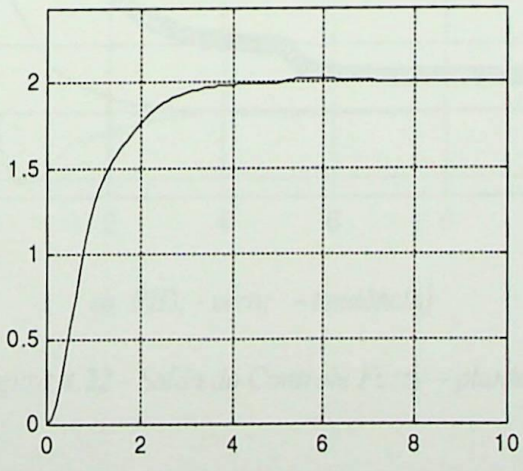



*Figura 4.20 – Resposta do Sistema – planta 2*

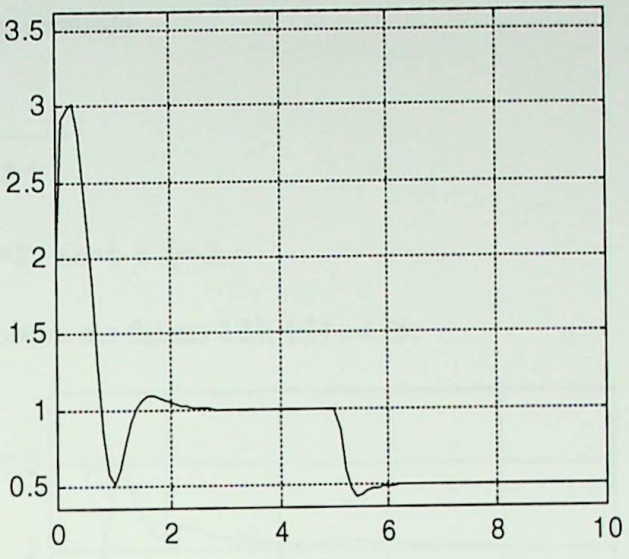
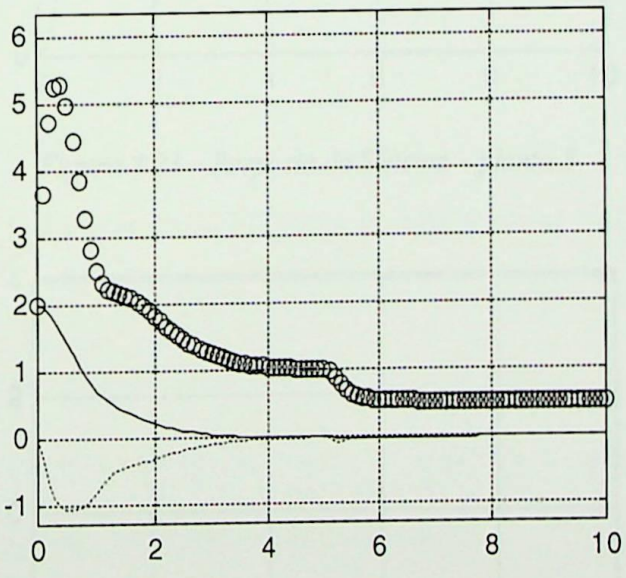



*Figura 4.21 - Saída do Controle Fuzzy – planta 2*




*(o PID; - erro; - - tendência)*

*Figura 4.22 - Saída do Controle Fuzzy – planta 2*

#### 4.2.3.3 Planta 3

Planta a ser simulada:

$$\frac{1}{s^2 - 2 \cdot s + 1}$$

com Wn=1; Ke=1 e $\zeta$=-2.

Como respostas nas figuras 4.23, 4.24 e 4.25.

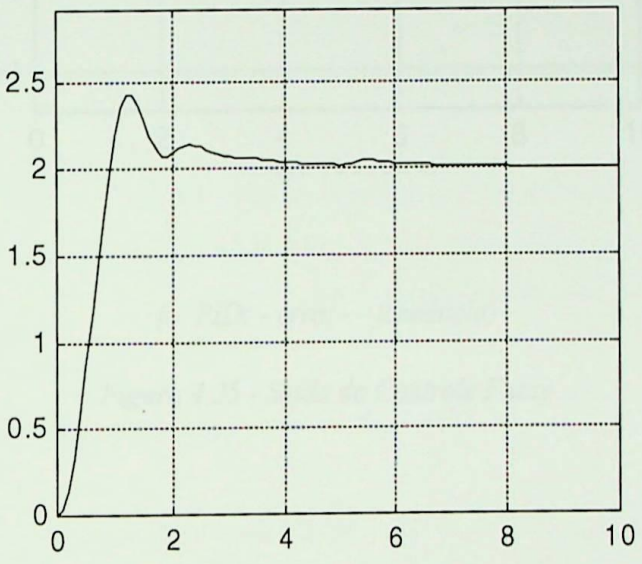



*Figura 4.23 – Resposta do Sistema – planta 3*

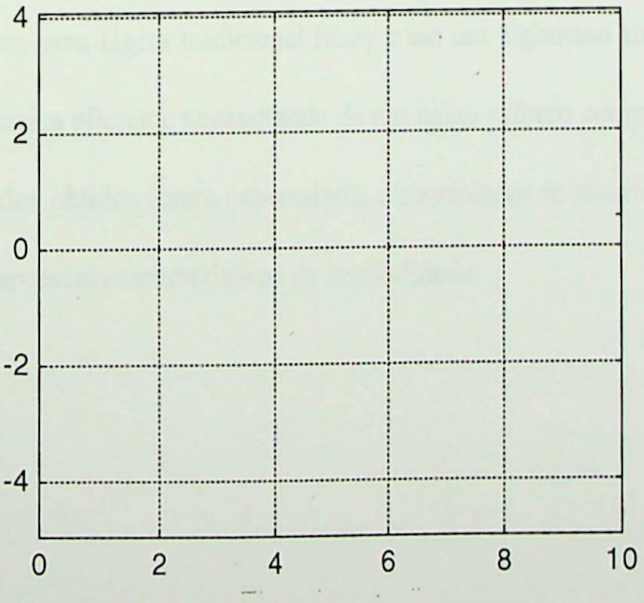
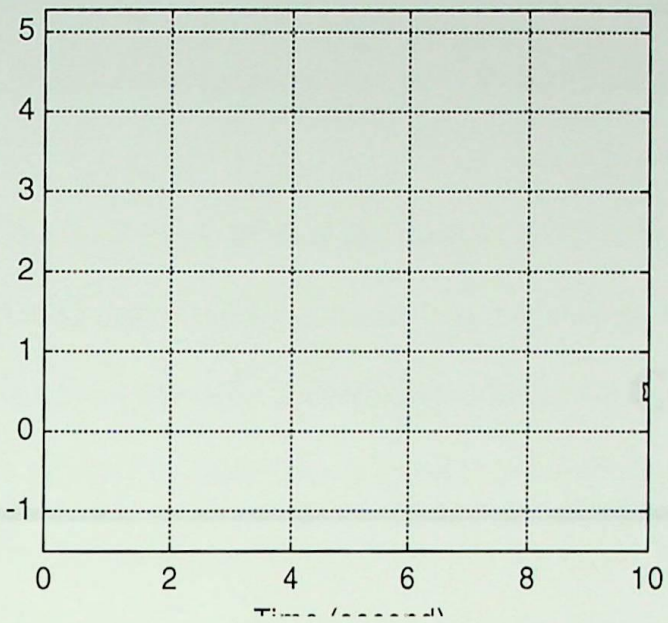



*Figura 4.24 – Saída do controle fuzzy – planta 3*




*(o PID; - erro; - - tendência)*

*Figura 4.25 - Saída do Controle Fuzzy*

### 4.2.4. Conclusões

O sistema de controle Fuzzy adaptativo se mostrou eficaz para plantas semelhantes. Por se basear em uma lógica tradicional fuzzy e em um algoritmo simples de controle resultou em uma técnica eficiente necessitando de um baixo esforço computacional.

Os resultados obtidos foram expressivos, o controlador se mostrou eficaz inclusive na planta 3 que apresenta características de instabilidade.

# 5

# Conclusão

*Neste capítulo será discutido o processo de desenvolvimento e teste, apresentadas as conclusões chegadas durante a execução das fases anteriores e sugestões de novos trabalhos a ser desenvolvidos tomando como base o compilador fuzzy .*

## 5.1. Considerações finais

O objetivo principal deste trabalho foi o desenvolvimento de uma ferramenta capaz de gerar código de linguagens de programação para aplicações de lógica fuzzy.

### 5.1.1. O Software

O Compilador Fuzzy foi desenvolvido e testado, sendo utilizado o Visual Basic 5.0 como plataforma de desenvolvimento. Isso levou a um ambiente de desenvolvimento gráfico totalmente amigável e orientado a eventos, aumentando assim sua flexibilidade.

O lay-out das telas da interface do software foram diagramadas buscando uma ergonomia que obedecesse a ordem natural de desenvolvimento da lógica fuzzy.

Os dados necessários para o desenvolvimento da lógica fuzzy, como demonstrado no capítulo 2, são armazenados utilizando o paradigma de banco de dados estruturado. Para tanto foram criadas três tabelas otimizadas utilizando-se o formato MDB do access.

A geração do código se processa a partir do relacionamento das tabelas criadas utilizando-se uma poderosa linguagem relacional: SQL [20, 21] (structured query languange).

O código pode ser gerado em duas linguagens (C ou Matlab®) ganhando assim recursos para implementação em diversos ambientes (o caso da linguagem C) ou poderosos recursos gráficos de testes e simulações (utilizando o simulink do MatLab®).

Durante o processo de implementação o usuário pode interagir com o programa anexando suas próprias funções, podendo desenvolver drivers de acesso a um hardware externo como CLPs, placas de aquisição de dados ou até mesmo câmeras. Isso torna capaz o desenvolvimento de sistemas de controle real, levando o software além do nível acadêmico.

O armazenamento dos dados no formato MDB permite uma ligação direta com softwares com o Word, Excel e Access, facilitando a documentação e a criação de relatórios de desenvolvimento.

### 5.1.2 Os Resultados

O código gerado segue rigorosamente as etapas de desenvolvimento de um sistema de lógica fuzzy demonstrados na bibliografia referente [1, 2, 7, 9, 11], sendo assim uma ferramenta confiável para seu propósito.

Extrapolando o projeto inicial, foi proposta uma nova técnica de controle adaptativa utilizando-se fuzzy, como apresentada no item 4.2.

O exemplos no item 4.2.3 mostram três diferentes funções de transferência, todas de segunda ordem, mas todas com o mesmo controle adaptativo fuzzy. Para todas, é aplicada uma entrada e um distúrbio ao longo do tempo e é obtido o resultado. Observa-se pe-

las figuras 4.17, 4.20 e 4.23 que em ambos os casos os sistemas tiveram uma rápida estabilização e um baixo overshoot. Constando-se assim, bons resultados.

Tão bons resultados são obtidos em [26, 27] onde define-se os memberships automaticamente utilizando-se a entropia de Komongorov-Sinai [28] . Nessa metodologia, basicamente faz-se o cálculo da entropia para cada elemento criando-se assim um vetor onde cada elemento é associado com sua respectiva entropia. Para a criação dos memberships percorre-se o já referido vetor em busca do menor valor de entropia. Encontrado tal elemento, divide-se os elementos em dois conjuntos que definem os memberships. Para novos memberships deve-se repetir o processo redividindo cada conjunto já obtido. Com isso tem-se um sistema robusto, com bons resultados, porém com um alto esforço computacional. O que já não acontece utilizando-se o controle fuzzy adaptativo proposto por este trabalho, onde um algoritmo de rápida resposta desloca os campos de pertinência da grandeza de saída do sistema fuzzy adequando-o assim às necessidades da planta. Tal algoritmo demonstrou-se eficaz, simples e exige baixo esforço computacional.

## 5.2 Sugestões para Trabalhos Futuros

Com o Compilador Fuzzy desenvolvido e testado, tem-se dois caminhos a serem seguidos;

**Implementação de novos módulos para o compilador**

Pode-se também incrementar o compilador, criando módulos para criação de campos de pertinência a partir de uma base de dados. Pode-se também ser implementado um módulo de otimização de regras e campos de pertinência utilizando-se técnicas de redes-neurais. Ainda pode ser desenvolvido um módulo dedicado de testes e simulações, deixando o compilador independente de outro software de simulação.

**Implementação de Aplicações para o código**

Utilizar o compilador para desenvolver uma aplicação de controle real, utilizando ou o PC ou um hardware específico (ex.:PLC – trabalho esse já inicializado para um PLC MPC4004 em parceria com a fabricante Atos). Nessa opção existem diversos outras variedades como integrar fuzzy com outro tipo de sistema de controle, utilizar em sistema supervisórios, etc.

Dentro desse contexto, o trabalho desenvolvido abriu caminho para novas pesquisas, podendo auxiliar alunos de graduação em iniciação científica, gerar novas dissertações de mestrado, publicações a nível de doutorado.

# Referências Bibliográficas


[1] Kaufmann, A. (1975), "Theory of Fuzzy Subsets", Academic Press, Inc.

[2] J.F. Baldwin, "Fuzzy logic and fuzzy reasoning," in Fuzzy Reasoning and Its Applications, E.H. Mamdani and B.R. Gaines, (eds.), London: Academic Press, 1981.

[3] Lygeros, John (1997), "A Formal Approach to Fuzzy Modeling", IEEE Transactions on Fuzzy Systems, Vol. 5, Nº. 3., August 1997, pg. 317

[4] Park, Mignon (1997), "A New Approach to Fuzzy Modeling", IEEE Transactions on Fuzzy Systems, Vol. 5, Nº. 3., August 1997, pg. 328

[5] MatLab® 5 - Manual On-line de Referência

[6] FuzzyTech Tutorial – Online Book – www.fuzzytech.com

[7] L.A. Zadeh, "Fuzzy sets," Info. & Ctl., Vol. 8, 1965, pp. 338-353.

[8] L.A. Zadeh, "Fuzzy algorithms," Info. & Ctl., Vol. 12, 1968, pp. 94-102.

[9] L.A. Zadeh, "Making computers think like people," I.E.E.E. Spectrum, 8/1984, pp. 26-32.

[10] S. Haack, "Do we need fuzzy logic?" Int. Jrnl. of Man-Mach. Stud., Vol. 11, 1979, pp.437

[11] Brule, James F. (1985), "Fuzzy Systens – A Tutorial" , Pacific Northwest National

[12] Ziegler, J. G.; N.B. Nichols and N.Y. Rochester (1942), Optimum settings for Automatic Controllers, Trans ASME, 64, 759.

[13] Ogata, Katsuhiko, (1993) "Engenharia de Controle Moderno", EditoraPrentice-Hall do Brasil Ltda.

[14] I.G. Umbers and P.J. King, "An analysis of human decision-making in cement kiln control and the implications for automation," Int. Jrnl. of Man-Mach. Stud., Vol. 12, 1980, pp. 11-23.

[15] Jain, R. Fuzzyism and real world problems. In P.P. Wang & S.K. Chang (Eds.), Fuzzy Sets, New York: Plenum Press.

[16] Zadeh, L.A. (1965) Fuzzy sets. Information and Control, Vol. 8, pp. 338-353.

[17] Zadeh, L.A. (1978) PRUF - A meaning representation language for natural languages. International Journal of Man-Machine Studies, Vol. 10, pp. 395-460.

[18] MicroftPress, Visual Basic 5 – On line Reference, 1998

[19] Zadeh, L.A. (1983) The role of fuzzy logic in the management of uncertainty in expert systems. Memorandum No. UCB/ERL M83/41, University of California, Berkeley.

[20] CRAG, John Clark, *Microsoft Visual Basic*, Makron Books, 1994.

[21] Microsoft Press, *Access for windows*, Makron Books, 1994.

[22] KOSKO, Bart, *Neural Networks and Fuzzy Systems – A Dynamical Systems Approach to Machine Intelligent*, Prentice-Hall International Editions, 1992.

[23] YEN, John et al.: *Industrial Applications of Fuzzy Logic ans Inteligence Systems*, IEEE Press, 1994.

[24] KRIS, Jamsa, Microsoft C; São Paulo, Makron Books, 1992

[25] SCHILD, Herbert; Borland C++: completo e total, São Paulo: Makron Books, 1997

[26] FERRARA, Nelson; PRADO Carmen P. Cintra do, (1994) "Caos – Uma introdução", Editora Edgard Blücher Ltda, 1994

[27] WANG, L.X., "Stable Adptive fuzzy Control of nonlinear Systems", IEEE Trans, Vol 1, pp 146-155, 1993

[28] FISCHLE, K, SHRÖDER, D., "Na Improved Adaptive Fuzzy Control Method", IEEE Fuzzy Systens, Vol 7, nº 1, pg 27-40, fev 1999

# A

# Apêndice

*Neste apêndice será mostrado as rotinas de programação mostradas no capítulo 2 e utilizadas para a implementação do software desenvolvido. As rotinas que não estão aqui listadas são rotinas próprias do ambiente de trabalho, estando assim encapsuladas.*

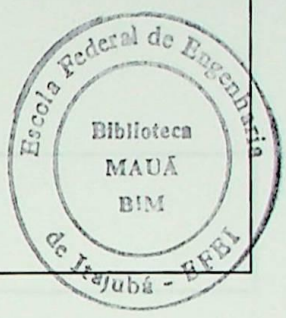
## A.1 Rotina do Evento 1.1.1 - Novo arquivo fuzzy.mdb

```
FileCopy "c:\fuzzy\modelo.mdb", "c:\fuzzy\fuzzy.mdb"
Data1.DatabaseName = "c:\fuzzy\fuzzy.mdb"
Data1.Refresh
sgrava.Enabled = True
sgrava.Visible = True
snovo.Enabled = False
tnomc.SetFocus
```

## A.2 Rotina do Evento 1.1.2 - Gravar dados

```
If tnomc.Text = "" Or ttmax.Text = "" Or ttmin.Text = "" Or tvass.Text = "" Then
MsgBox "Existem campos em branco", 48, "A T E N Ç Ã O"
Exit Sub
End If
Data1.Recordset.AddNew
Data1.Recordset.Fields("cnom").Value = tnomc.Text
Data1.Recordset.Fields("cmax").Value = CDbl(ttmax.Text)
```

```
Data1.Recordset.Fields("cmin").Value = CDbl(ttmin.Text)
Data1.Recordset.Fields("vass").Value = tvass.Text
Data1.Recordset.Fields("tip").Value = CInt(Mid$(Combo1.Text, 1, 1))
Data1.Recordset.Fields("cend").Value = CInt(Val("&H" + tend.Text))
Data1.Recordset.Fields("cpes").Value = CDbl(tpes.Text)
Data1.Recordset.Update
Data1.Refresh
tnomc.Text = ""
tvass.Text = ""
ttmax = ""
ttmin = ""
tend.Text = ""
tpes.Text = "1"
```




## A.3 Rotina do Evento 1.2.1.1

```
Data2.RecordSource = "select * from mship where mscam=" & """ & Combo1.Text & """ & "order by msnum"
Data2.Refresh
escmax = 0
escmin = 0
Data1.Recordset.MoveFirst
Data1.Recordset.FindFirst "cnom=" & """ & Trim$(Combo1.Text) & """
escmax = Data1.Recordset.Fields("cmax")
escmin = Data1.Recordset.Fields("cmin")
flagvass = Data1.Recordset.Fields("vass")
tmin.Text = escmin
tmax.Text = escmax
```

## A.4 Rotina do Evento 1.2.1.2

```
If flag_geral = (2 * CInt(tnum.Text) + 1) Then
MsgBox "Excedido o número máximo de memberships", 65, "Atenção"
SSCommand3.SetFocus
Exit Sub
End If

List1.AddItem tnme.Text
```

```
tnme.Text = ""
flag_geral = flag_geral + 1
If flag_geral = (2 * CInt(tnum.Text) + 1) Then
SSCommand3.SetFocus
Exit Sub
End If
tnme.SetFocus
```

## A.5 Rotina do Evento 1.2.1.3

```
List1.Clear
flag_geral = 0
tnme.Text = ""
```

## A.6 Rotina do Evento 1.2.1.4

```
Dim flag As Integer
Dim pres As Double
Dim p1 As Double, pu As Double
flag = 1
pres = (CDbl(tpre.Text) / 100) * (CDbl(tmax.Text) - CDbl(tmin.Text)) / 2
p1 = ((CDbl(tcen.Text) / CDbl(tgan.Text)) - (CDbl(tnum.Text) * pres))
pu = ((CDbl(tcen.Text) / CDbl(tgan.Text)) + (CDbl(tnum.Text) * pres))
If escmin > p1 Then
MsgBox "Numero de MS ou a precisao setada" + Chr$(13) + "fazem extrapolar o limite inferior",
65, "Atenção"
Exit Sub
End If
If escmax < pu Then
MsgBox "Numero de MS ou a precisao setada" + Chr$(13) + "fazem extrapolar o limite superi-
or", 65, "Atenção"
Exit Sub
End If
If Data2.Recordset.RecordCount > 0 Then
Data2.Recordset.MoveLast
flag = Data2.Recordset.Fields("msnum") + 1
End If
For f = 0 To flag_geral - 1
```

```
'primeira perna
Data2.Recordset.AddNew
Data2.Recordset.Fields("msnum") = flag + f
Data2.Recordset.Fields("posx1") = p1 - pres
Data2.Recordset.Fields("posx2") = p1
Data2.Recordset.Fields("posy1") = 0
If f = 0 Then
Data2.Recordset.Fields("posx1") = escmin
End If
If f = 0 Then
Data2.Recordset.Fields("posy1") = 1
End If
Data2.Recordset.Fields("posy2") = 1
Data2.Recordset.Fields("mscam") = Combo1.Text
Data2.Recordset.Fields("msnom") = List1.List(f)
Data2.Recordset.Fields("msvas") = flagvass
Data2.Recordset.Update
Data2.Refresh
'segunda perna
Data2.Recordset.AddNew
Data2.Recordset.Fields("msnum") = flag + f
Data2.Recordset.Fields("posx1") = p1 + pres
Data2.Recordset.Fields("posx2") = p1
Data2.Recordset.Fields("posy1") = 0
If f = flag_geral - 1 Then
Data2.Recordset.Fields("posx1") = escmax
End If
If f = flag_geral - 1 Then
Data2.Recordset.Fields("posy1") = 1
End If
Data2.Recordset.Fields("posy2") = 1
Data2.Recordset.Fields("mscam") = Combo1.Text
Data2.Recordset.Fields("msnom") = List1.List(f)
Data2.Recordset.Fields("msvas") = flagvass
Data2.Recordset.Update
Data2.Refresh
p1 = p1 + pres
Next f
```

## A.7 Rotina do Evento 1.2.2.1 - Selecionar campo de atuação

```
Data2.RecordSource = "select * from mship where mscam=" & "'" & Combo2.Text & "'" & "order by msnum"
Data2.Refresh
Data1.Recordset.FindFirst "cnom=" & "'" & Combo2.Text & "'"
escmax = Data1.Recordset.Fields("cmax").Value
escmin = Data1.Recordset.Fields("cmin").Value
flagvass = Data1.Recordset.Fields("vass").Value
Cls
plota
```

## A.8 Rotina do Evento 1.2.2.2 - Criar segmentos de memberships

Caso o usuário escolha a paralela:

```
If tnom.Text = "" Then
MsgBox "Sem nome do Membership", 48, "A T E N Ç Ã O"
Exit Sub
End If
Data3.Refresh
Data3.Recordset.FindFirst "msnom=" & "'" & tnom.Text & "'" & " and msnum<>" & CInt(Text5.Text) & " and mscam=" & "'" & Combo2.Text & "'"
If Not Data3.Recordset.NoMatch Then
MsgBox "Nome repetido", 48, "A T E N Ç Ã O"
Exit Sub
End If
ForeColor = &HFF&
DrawWidth = 2
Line ((CDbl(tinf.Text) - escmin) * 9240 / (escmax - escmin) + 120, 700)-((CDbl(tsup.Text) - escmin) * 9240 / (escmax - escmin) + 120, 700)
If CInt(Text5.Text) = 0 Then
Text5.Text = 1
End If
Data2.Recordset.AddNew
Data2.Recordset.Fields("msnum").Value = CInt(Text5.Text)
Data2.Recordset.Fields("posx1").Value = CDbl(tinf.Text)
Data2.Recordset.Fields("posx2").Value = CDbl(tsup.Text)
```

```
Data2.Recordset.Fields("posy1").Value = 1
Data2.Recordset.Fields("posy2").Value = 1
Data2.Recordset.Fields("mscam").Value = Combo2.Text
Data2.Recordset.Fields("msnom").Value = tnom.Text
Data2.Recordset.Fields("msvas").Value = flagvass
Data2.Recordset.Update
Data2.Refresh
```

Caso o usuário escolha a rampa:

```
If tnom.Text = "" Then
MsgBox "Sem nome do Membership", 48, "A T E N Ç Ã O"
Exit Sub
End If

Data3.Refresh
Data3.Recordset.FindFirst "msnom=" & "'" & tnom.Text & "'" & " and msnum<>" & CInt(Text5.Text) & " and mscam=" & "'" & Combo2.Text & "'"
If Not Data3.Recordset.NoMatch Then
MsgBox "Nome repetido", 48, "A T E N Ç Ã O"
Exit Sub
End If
ForeColor = &HFF&
DrawWidth = 2
Line ((CDbl(tinf.Text) - escmin) * 9240 / (escmax - escmin) + 120, 1550)-((CDbl(tsup.Text) - escmin) * 9240 / (escmax - escmin) + 120, 700)
If CInt(Text5.Text) = 0 Then
Text5.Text = 1
End If
Data2.Recordset.AddNew
Data2.Recordset.Fields("msnum").Value = CInt(Text5.Text)
Data2.Recordset.Fields("posx1").Value = CDbl(tinf.Text)
Data2.Recordset.Fields("posx2").Value = CDbl(tsup.Text)
Data2.Recordset.Fields("posy1").Value = 0
Data2.Recordset.Fields("posy2").Value = 1
Data2.Recordset.Fields("mscam").Value = Combo2.Text
Data2.Recordset.Fields("msnom").Value = tnom.Text
Data2.Recordset.Fields("msvas").Value = flagvass
Data2.Recordset.Update
```

```
Data2.Refresh
```

## A.9 Rotina do Evento 1.2.2.3 - Avançar um membership

```
If Data2.Recordset.RecordCount = 0 Then
MsgBox "Não existem registros", 64, "Comunicado:"
Text5.Text = 1
Cls
Exit Sub
End If

'verifica se esta no ultimo registro
flag = 0
Data2.Recordset.MoveFirst
Do While Not Data2.Recordset.EOF
'procura o maior numero de registro
If Data2.Recordset.Fields("msnum").Value > flag Then
flag = Data2.Recordset.Fields("msnum").Value
End If
Data2.Recordset.MoveNext
Loop
'se o maior numero for menor ou igual esta no ultimo reg
If flag <= CInt(Text5.Text) Then
Exit Sub
End If

Text5.Text = CInt(Text5.Text) + 1

Cls
plota
```

## A.10 Rotina do Evento 1.2.2.4 - Retornar um membership

```
If Data2.Recordset.RecordCount = 0 Then
MsgBox "Não existem registros", 64, "Comunicado:"
Exit Sub
End If
```

```
'caso nao tenha registros anteriores saia
If CInt(Text5.Text) = 0 Or CInt(Text5.Text) = 1 Then
Exit Sub
End If
Cls
Text5.Text = CInt(Text5.Text) - 1

plota
```

## A.11 Rotina do Evento 1.2.2.5 - Criar novo membership

```
If Data2.Recordset.RecordCount = 0 Then
Text5.Text = 1
Exit Sub
End If

Data2.Recordset.MoveLast
flag = Data2.Recordset.Fields("msnum")

If f2 = 0 Then
Cls
End If

Text5.Text = flag + 1
tnom.Text = ""
```

## A.12 Rotina do Evento 1.2.2.6 - Alternar visualização de memberships

```
If Data2.Recordset.RecordCount = 0 Then
MsgBox "Não existem registros", 64, "Comunicado:"
Exit Sub
End If

f1 = CInt(Text5.Text)

'para f2=0 visualiza todos os memberships
```

```
If f2 = 0 Then
Data2.Recordset.MoveFirst
Do While Not Data2.Recordset.EOF
x1 = Data2.Recordset.Fields("posx1")
x2 = Data2.Recordset.Fields("posx2")
x1 = (x1 - escmin) * 9240 / (escmax - escmin) + 120
x2 = (x2 - escmin) * 9240 / (escmax - escmin) + 120
y1 = 1550 - Data2.Recordset.Fields("posy1") * 850
y2 = 1550 - Data2.Recordset.Fields("posy2") * 850
Line (x1, y1)-(x2, y2)
Data2.Recordset.MoveNext
Loop
f2 = 1
Exit Sub
End If

'para f2=1 se nao houver reg so limpa e sai
If f2 = 1 Then
 If Data2.Recordset.RecordCount = 0 Then
  Cls
  Text5.Text = 1
  Exit Sub
 End If

 'caso contrario procura o 1 campo f1
 Data2.Recordset.FindFirst "msnum=" & f1

 'se nao encontra, aponta para o reg 1
 If Data2.Recordset.NoMatch Then
  Data2.Recordset.MoveFirst
 End If

 'retorna text5 como o valor do reg
 Text5.Text = Data2.Recordset.Fields("msnum").Value
 Cls
 plota
 f2 = 0
End If
```

## A.13 Rotina do Evento 1.3.1 - Escolha do Campo

```
'seleciona da tabela mship grupo de valores do nome quando o
'campo for igual ao selecionado por combo1
Data2.RecordSource = "select distinct msnom,msvas from mship where mscam=" & "'" & Combo1.Text & "'" '& "group by msnom"
Data2.Refresh
'se não houver registro sai
If Data2.Recordset.RecordCount = 0 Then
MsgBox "Não há campos de pertinência associados", 48, "A T E N Ç Ã O"
Exit Sub
End If
tnvass.Text = Data2.Recordset.Fields("msvas").Value
'Seleciona da tabela "mship" os memberships dos registros selecionados
Data2.Recordset.MoveFirst
Combo2.Text = Trim$(Data2.Recordset.Fields("msnom").Value)
Do While Not Data2.Recordset.EOF
Combo2.AddItem Trim$(Data2.Recordset.Fields("msnom").Value)
Data2.Recordset.MoveNext
Loop
```

```
'/* Operador ( = = )
argumento_1 = "(Min_" + Trim$(Combo1.Text) + "_" + Trim$(Combo2.Text) + " <= " + Trim$(tnvass.Text)
argumento_2 = Trim$(tnvass.Text) + " <= " + "Max_" + Trim$(Combo1.Text) + "_" + Trim$(Combo2.Text) + ")"
tcond = tcond + argumento_1 + " & " + argumento_2
tcond = tcond + ") "
ttrad1 = ttrad1 + Trim$(Combo1.Text) + " == " + Trim$(Combo2.Text)
List1.AddItem "IPe_" + Trim$(Combo1.Text) + "_" + Trim$(Combo2.Text)

'/* função (min)
tacao = "IPe_" + Trim$(Combo1.Text) + "_" + Trim$(Combo2.Text) + " = min(" + List1.List(0)
ttrad2 = ttrad2 + Trim$(Combo1.Text) + " = " + Trim$(Combo2.Text)
For f = 1 To List1.ListCount - 1
tacao = tacao + "," + List1.List(f)
Next f
```

```
tacao = tacao + ");"

'/* função do usuário (fun)
If arqdefuz = "" Or arqdefuz = " " Then
MsgBox "Arquivo de funções inexistente"
Exit Sub
End If
tacao = "IPe_" + Trim$(Combo1.Text) + "_" + Trim$(Combo2.Text) + " = fun(" + List1.List(0)
ttrad2 = ttrad2 + Trim$(Combo1.Text) + " = " + Trim$(Combo2.Text)
For f = 1 To List1.ListCount - 1
tacao = tacao + "," + List1.List(f)
Next f
tacao = tacao + ");"
End Sub
'/* operador ( AND )
If flagcdac = 1 Then
tcond = tcond + " & "
ttrad1 = ttrad1 + " e "
Else
tacao = tacao + " & "
ttrad2 = ttrad2 + " e "
End If
'/* Operador (OR)
If flagcdac = 1 Then
tcond = tcond + " ! "
Else
tacao = tacao + " ! "
End If

'/* Operadores ( '(' ) , ( ')' ) , ( '<') e ('>')
If flagcdac = 1 Then
tcond = tcond + " ("
Else
tacao = tacao + " ("
End If
If flagcdac = 1 Then
tcond = tcond + ") "
Else
```

```
tacao = tacao + ") "
End If
If flagcdac = 1 Then
tcond = tcond + " > "
ttrad1 = ttrad1 + " > "
Else
tacao = tacao + " > "
ttrad2 = ttrad2 + " > "
End If
If flagcdac = 1 Then
tcond = tcond + " < "
ttrad1 = ttrad1 + " < "
Else
tacao = tacao + " < "
ttrad2 = ttrad2 + " < "
End If

'/* Função (FUZZY RELATION)
If flagcdac = 1 Then
tcond = tcond + "(IPe_" + RTrim$(Combo1.Text) + "_" + RTrim$(Combo2.Text) + ">0)"
ttrad1.Text = trad1.Text + RTrim$(Combo1.Text) + " = " + RTrim$(Combo2.Text)
End If
```

## A.14 Rotina do Evento 1.3.4 - Retorna regra

```
If Data3.Recordset.RecordCount = 0 Then
MsgBox "Não existem registros", 48, "A T E N Ç Ã O"
Exit Sub
End If
List1.Clear
Data3.Recordset.MovePrevious
If Data3.Recordset.BOF Then
Data3.Recordset.MoveNext
End If
tcond.Text = Data3.Recordset.Fields("r_cond").Value
tacao.Text = Data3.Recordset.Fields("r_acao").Value
ttrad1.Text = Data3.Recordset.Fields("r_trad1").Value
ttrad2.Text = Data3.Recordset.Fields("r_trad2").Value
```

```
Text1.Text = Data3.Recordset.Fields("r_num").Value
```

## A.15 Rotina do Evento 1.3.5 - Avança regra

```
If Data3.Recordset.RecordCount = 0 Then
MsgBox "Não existem registros", 48, "A T E N Ç Ã O"
Exit Sub
End If
List1.Clear
Data3.Recordset.MoveNext
If Data3.Recordset.EOF Then
Data3.Recordset.MovePrevious
End If
tcond.Text = Data3.Recordset.Fields("r_cond").Value
tacao.Text = Data3.Recordset.Fields("r_acao").Value
ttrad1.Text = Data3.Recordset.Fields("r_trad1").Value
ttrad2.Text = Data3.Recordset.Fields("r_trad2").Value
Text1.Text = Data3.Recordset.Fields("r_num").Value
```

## A.16 Rotina do Evento 1.3.6 - Alteração de regras

```
Data3.Recordset.Edit
Data3.Recordset.Fields("r_num").Value = CInt(Text1.Text)
Data3.Recordset.Fields("r_cond").Value = tcond.Text
Data3.Recordset.Fields("r_acao").Value = tacao.Text
Data3.Recordset.Fields("r_trad1").Value = ttrad1.Text
Data3.Recordset.Fields("r_trad2").Value = ttrad2.Text
Data3.Recordset.Update
Data3.Refresh
tcond.Text = Data3.Recordset.Fields("r_cond").Value
tacao.Text = Data3.Recordset.Fields("r_acao").Value
ttrad1.Text = Data3.Recordset.Fields("r_trad1").Value
ttrad2.Text = Data3.Recordset.Fields("r_trad2").Value
Text1.Text = Data3.Recordset.Fields("r_num").Value
```

## A.17 Rotina do Evento 1.3.7 - Deletar regras

```
If Data3.Recordset.RecordCount = 0 Then
MsgBox "Não existem registros", 48, "A T E N Ç Ã O"
Exit Sub
End If
List1.Clear
Data3.Recordset.Delete
Data3.Refresh
If Data3.Recordset.RecordCount = 0 Then
MsgBox "Não existem registros", 48, "A T E N Ç Ã O"
Exit Sub
End If
Data3.Recordset.MoveNext
If Data3.Recordset.EOF Then
Data3.Recordset.MovePrevious
End If
tcond.Text = Data3.Recordset.Fields("r_cond").Value
tacao.Text = Data3.Recordset.Fields("r_acao").Value
ttrad1.Text = Data3.Recordset.Fields("r_trad1").Value
ttrad2.Text = Data3.Recordset.Fields("r_trad2").Value
Text1.Text = Data3.Recordset.Fields("r_num").Value
```

## A.18 Rotina Comum de Geração

```
If arq_ml = True Or mataag = True Then
mat_inicializa
mat_calcula
mat_aplica
List1.AddItem "% INICIO DO BLOCO 4 - DEFUZZYFICACAO*/"
List1.AddItem " "
Data7.Recordset.MoveFirst
Do While Not Data7.Recordset.EOF
If Data7.Recordset.Fields("tip") = 2 Or Data7.Recordset.Fields("tip") = 4 Then
If tipo = 1 Then
mat_defuzzy1 (Data7.Recordset.Fields("cnom"))
ElseIf tipo = 2 Then
```

```
mat_defuzzy2 (Data7.Recordset.Fields("cnom"))
ElseIf tipo = 3 Then
mat_defuzzy3 (Data7.Recordset.Fields("cnom"))
ElseIf tipo = 4 Then
mat_defuzzy4 (Data7.Recordset.Fields("cnom"))
End If
End If
Data7.Recordset.MoveNext
Loop
ElseIf teste = True Or superc = True Then
inicializa
If arqinici <> "" And arqinici <> " " Then
learquivo (arqinici)   'arquivo usuario
End If
levariaveis
If arqlevar <> "" And arqlevar <> " " Then
learquivo (arqlevar)   'arquivo usuario
End If
calcula
aplica
If arqaplic <> "" And arqaplic <> " " Then
learquivo (arqaplic)   'arquivo usuario
End If
List1.AddItem "/* INICIO DO BLOCO 4 - DEFUZZYFICACAO*/"
List1.AddItem " "
Data7.Recordset.MoveFirst
Do While Not Data7.Recordset.EOF
canal = 0
If Data7.Recordset.Fields("tip") = 2 Or Data7.Recordset.Fields("tip") = 4 Then
If tipo = 1 Then
defuzzy1 (Data7.Recordset.Fields("cnom"))
ElseIf tipo = 2 Then
defuzzy2 (Data7.Recordset.Fields("cnom"))
ElseIf tipo = 3 Then
defuzzy2 (Data7.Recordset.Fields("cnom"))
ElseIf tipo = 4 Then
defuzzy4 (Data7.Recordset.Fields("cnom"))
End If
```

```
End If
Data7.Recordset.MoveNext
Loop

If arqdefuz <> "" Or arqdefuz <> " " Then
learquivo (arqdefuz)
End If
If teste.Value = True Then
List1.AddItem "}"
Else
List1.AddItem "}while(!kbhit());"
List1.AddItem "}"
End If
List1.AddItem "/* FIM DO SISTEMA */"
If arqdefuz <> "" Or arqdefuz <> " " Then
learquivo (arqfun)
End If
End If
```

## A.19 Evento 3.2.2 - Processo de Geração por Fusão de Área

Mesma que no evento 3.2.1

## A.20 Evento 3.2.3 - Processo de Geração por Máximos Locais

Mesma que no evento 3.2.1

## A.21 Função "inicializa()"

```
List1.AddItem "#define max(a,b) ((a)<(b)?(b):(a))"
List1.AddItem "#define men(a,b) ((a)<(b)?(a):(b))"
List1.AddItem "#define abs(a) ((a)>(0)?(a): (-a))"
List1.AddItem "/* BLOCO 1 - INICIALIZACAO DOS INDICES DE  PERTINENCIA */"
List1.AddItem "/* - INICIALIZACAO DOS LIMITES DOS MEMBERSHIPS  */"
List1.AddItem " "
List1.AddItem "main() "
List1.AddItem "{ "
```

```
List1.AddItem " "

If teste.Value = False Then
List1.AddItem "do "
List1.AddItem "{ "
End If
Data7.Recordset.MoveFirst
Do While Not Data7.Recordset.EOF
X = Data7.Recordset.Fields("cnom")
List1.AddItem "float Pes_" + X + " = " + CStr(Data7.Recordset.Fields("cpes")) + ";"
Data7.Recordset.MoveNext
Loop
List1.AddItem ""
Data2.Recordset.MoveFirst
Do While Not Data2.Recordset.EOF
X = Trim$(Data2.Recordset.Fields("mscam") + "_" + Data2.Recordset.Fields("msnom"))
List1.AddItem "float IPe_" + X + " = 0 ;"
List1.AddItem "float Min_" + X + " = " + CStr(fmin(Data2.Recordset.Fields("mscam"), Da-
ta2.Recordset.Fields("msnom"))) + ";"
List1.AddItem "float Max_" + X + " = " + CStr(fmax(Data2.Recordset.Fields("mscam"), Da-
ta2.Recordset.Fields("msnom"))) + ";"
Data2.Recordset.MoveNext
Loop
List1.AddItem "float soma;"
List1.AddItem "int flag,flag1;"
List1.AddItem "float x1,x2,y1,y2,xmin,xmax,ymax,ymax1,yant;"
List1.AddItem "float a1,a2;"
List1.AddItem "float m1,m2;"
List1.AddItem "float xx1,xm1,xx2,xm2,xx,xm,xm2;"
List1.AddItem "float indice,ipert;"
'List1.AddItem "float menor;"
List1.AddItem "float x;"
List1.AddItem "char pausa;"
List1.AddItem "unsigned int base,resp;"
List1.AddItem "int porta;"
List1.AddItem "int canal= 0;"
List1.AddItem "float retorno,matriz[50];"
List1.AddItem "int elementos,count;"
List1.AddItem "int cont,temp1,temp2,temp3,temp4;"
```

```
List1.AddItem "float saidaana1,saidaana2;"
List1.AddItem " "
List1.AddItem " "
List1.AddItem " "
'faz varredura de entradas/saidas
Data7.Recordset.MoveFirst
Do While Not Data7.Recordset.EOF
List1.AddItem "float " + Trim$(Data7.Recordset.Fields("vass")) + ";"
Data7.Recordset.MoveNext
Loop
List1.AddItem " "
List1.AddItem "/* FIM DO BLOCO 1 */"
```

## A.22 Função "levariaveis ()"

```
List1.AddItem " "
If teste.Value = True Then
Data7.Recordset.MoveFirst
Do While Not Data7.Recordset.EOF
If Data7.Recordset.Fields("tip") = 1 Or Data7.Recordset.Fields("tip") = 3 Then
List1.AddItem "printf(" + Chr$(34) + "Entrada: " + Trim$(Data7.Recordset.Fields("vass")) + " =
" + Chr$(34) + ");"
List1.AddItem "gets(pausa);"
List1.AddItem "sscanf(pausa," + Chr$(34) + "%f" + Chr$(34) + ",&" +
Trim$(Data7.Recordset.Fields("vass")) + ");"
End If
Data7.Recordset.MoveNext
Loop
End If
If teste.Value = False Then
'ordem das portas dada pela ordem de entrada no arquivo fuzzy.mdb
'inicializa com 0 e incrementa a cada interacao
List1.AddItem "porta=0;"
Data7.Recordset.MoveFirst
Do While Not Data7.Recordset.EOF
If Data7.Recordset.Fields("tip") = 1 Or Data7.Recordset.Fields("tip") = 3 Then
List1.AddItem "base=" + Hex(Data7.Recordset.Fields("cend")) + ";"
List1.AddItem "leportaana(porta,base,&" + Trim$(Data7.Recordset.Fields("vass")) + ");"
List1.AddItem "porta++;"
```

```
List1.AddItem " "
End If
Data7.Recordset.MoveNext
Loop
End If
```

## A.23 Função "calcula()"

```
List1.AddItem "/* BLOCO 2 - CALCULO DOS INDICES DE PERTINENCIA */"
List1.AddItem " "
Dim X As String
Dim x1 As String
Dim xx1 As String
Dim xx2 As String
Dim xy1 As String
Dim xy2 As String
Data3.Recordset.MoveFirst
Do While Not Data3.Recordset.EOF
Data7.Recordset.MoveFirst
Data7.Recordset.FindFirst "cnom=" + "'" + Data3.Recordset.Fields("mscam") + "'"
'gera somente os MS da entrada
If Data7.Recordset.Fields("tip") = 1 Then
X = ""
x1 = ""
xx1 = ""
xx2 = ""
xy1 = ""
xy2 = ""
X = Trim$(Data3.Recordset.Fields("msvas"))
x1 = Trim$("IPe_" + Data3.Recordset.Fields("mscam") + "_" + Da-
ta3.Recordset.Fields("msnom"))
xx1 = Trim$(CStr(Data3.Recordset.Fields("posx1")))
xx2 = Trim$(CStr(Data3.Recordset.Fields("posx2")))
xy1 = Trim$(CStr(Data3.Recordset.Fields("posy1")))
xy2 = Trim$(CStr(Data3.Recordset.Fields("posy2")))
If Data3.Recordset.Fields("posx1") < Data3.Recordset.Fields("posx2") Then
List1.AddItem " "
List1.AddItem "if (" + X + " > " + xx1 + " & " + X + " <= " + xx2 + ")"
```

```
List1.AddItem "  if (" + x1 + " < ((" + X + "-(" + xx1 + "))*((" + xy2 + ")-(" + xy1 + "))/((" + xx2 + ")-(" + xx1 + ")))+(" + xy1 + "))"
List1.AddItem "    " + x1 + " = ((" + X + "-(" + xx1 + "))*((" + xy2 + ")-(" + xy1 + "))/((" + xx2 + ")-(" + xx1 + ")))+(" + xy1 + ");"
End If
If Data3.Recordset.Fields("posx1") > Data3.Recordset.Fields("posx2") Then
List1.AddItem " "
List1.AddItem "if (" + X + " > " + xx2 + " & " + X + " <= " + xx1 + " )"
List1.AddItem "  if (" + x1 + " < ((" + xx1 + "-" + X + ")*((" + xy2 + ")-(" + xy1 + "))/((" + xx1 + ")-(" + xx2 + ")))+(" + xy1 + "))"
List1.AddItem "    " + x1 + " = ((" + xx1 + "-" + X + ")*((" + xy2 + ")-(" + xy1 + "))/((" + xx1 + ")-(" + xx2 + ")))+(" + xy1 + ");"
End If
End If
Data3.Recordset.MoveNext

Loop
If teste.Value = 1 Then
Data2.Recordset.MoveFirst
Do While Not Data2.Recordset.EOF
X = Trim$(Data2.Recordset.Fields("mscam") + "_" + Data2.Recordset.Fields("msnom"))
List1.AddItem "if (IPe_" + X + " > 0)"
List1.AddItem "  printf(" + Chr$(34) + X + " = %f \n" + Chr$(34) + ", IPe_" + X + ");"
Data2.Recordset.MoveNext
Loop
List1.AddItem "gets (pausa);"
End If
List1.AddItem " "
List1.AddItem "/* FIM DO BLOCO 2 */"
List1.AddItem " "
```



## A.24 Função "aplica()"

```
List1.AddItem " "
List1.AddItem "/* BLOCO 3 - APLICACAO DAS REGRAS */"
List1.AddItem " "
Data1.Recordset.MoveFirst
Do While Not Data1.Recordset.EOF
List1.AddItem ""
```

```
aa1 = Len(Data1.Recordset.Fields("r_acao"))
nom_ind = ""
nom_ind2 = ""
menor = ""
nom_ind3 = ""
funcao = ""
elementos = ""
flagind = 0
f = 1
'joga a condicao para o  programa
List1.AddItem "if (" + Trim$(Data1.Recordset.Fields("r_cond"))
List1.AddItem "{ "
'le o nome do indice a ser controlado / rotina de busca
Do While Not Mid$(Data1.Recordset.Fields("r_acao"), f, 1) = "="
nom_ind = nom_ind + Mid$(Data1.Recordset.Fields("r_acao"), f, 1)
f = f + 1
Loop
'encontra a funcao a ser aplicada
Do While Mid$(Data1.Recordset.Fields("r_acao"), f, 1) = " " Or
Mid$(Data1.Recordset.Fields("r_acao"), f, 1) = "="
f = f + 1
Loop
Do While Mid$(Data1.Recordset.Fields("r_acao"), f, 1) <> "("
funcao = funcao + Mid$(Data1.Recordset.Fields("r_acao"), f, 1)
f = f + 1
Loop
f = f + 1
'encontra o vetor de elementos
flag_n_elementos = 0
char_flag = "0"
Do While Mid$(Data1.Recordset.Fields("r_acao"), f, 1) <> ")"
elementos = elementos + Mid$(Data1.Recordset.Fields("r_acao"), f, 1)
'a cada , aumenta um elemento
If Mid$(Data1.Recordset.Fields("r_acao"), f, 1) = "," Then
flag_n_elementos = flag_n_elementos + 1
char_flag = char_flag + "," + CStr(flag_n_elementos)
End If
f = f + 1
Loop
```

```
'manda funcao para o programa
List1.AddItem "matriz[" + char_flag + "]=" + elementos + ";"
List1.AddItem "elementos=" + CStr(flag_n_elementos + 1) + ";"
List1.AddItem funcao + " (elementos,matriz,&retorno);"
List1.AddItem nom_ind + "= max(" + nom_ind + ",retorno);"
List1.AddItem "}"
List1.AddItem ""
Data1.Recordset.MoveNext
Loop
'rotina de teste
If teste.Value = 1 Then
Data2.Recordset.MoveFirst
Do While Not Data2.Recordset.EOF
X = Trim$(Data2.Recordset.Fields("mscam") + "_" + Data2.Recordset.Fields("msnom"))
List1.AddItem ""
List1.AddItem "if (IPe_" + X + " > 0)"
List1.AddItem "  printf(" + Chr$(34) + X + " = %f \n" + Chr$(34) + ", IPe_" + X + ");"
Data2.Recordset.MoveNext
Loop
List1.AddItem "gets (pausa);"
End If
List1.AddItem " "
List1.AddItem "/* FIM DO BLOCO 3 */"
List1.AddItem " "
```

## A.25 Função "defuzzy1()"

```
'variaveis que assumem posx1,...,posy2 de mship
Dim x1 As Double, x2 As Double
Dim y1 As Double, y2 As Double
Dim soma As Double, ipert As Double
List1.AddItem "xx = 0;"
List1.AddItem "xm = 0;"
Data5.RecordSource = "select distinct mscam,msnom from mship where mscam=" + "'" + campo
+ "'"
Data5.Refresh
Data5.Recordset.MoveFirst
Do While Not Data5.Recordset.EOF
```

```
Data6.RecordSource = "select * from mship where mscam=" + "'" + campo + "'" + " and msnom= " + "'" + Trim$(Data5.Recordset.Fields("msnom")) + "'" + " order by posx1"
Data6.Refresh
Data6.Recordset.MoveFirst
Do While Not Data6.Recordset.EOF
'encontra a area de um membership
x1 = Data6.Recordset.Fields("posx1")
x2 = Data6.Recordset.Fields("posx2")
y1 = Data6.Recordset.Fields("posy1")
y2 = Data6.Recordset.Fields("posy2")
List1.AddItem "x1 =" + CStr(Data6.Recordset.Fields("posx1")) + ";"
List1.AddItem "x2 =" + CStr(Data6.Recordset.Fields("posx2")) + ";"
List1.AddItem "y1 =" + CStr(Data6.Recordset.Fields("posy1")) + ";"
List1.AddItem "y2 =" + CStr(Data6.Recordset.Fields("posy2")) + ";"
List1.AddItem "indice= IPe_" + Trim$(Data6.Recordset.Fields("mscam")) + "_" + Trim$(Data6.Recordset.Fields("msnom")) + ";"
List1.AddItem ""
List1.AddItem ""

List1.AddItem "if (indice > 0)"
List1.AddItem "{"
'primeiro caso
If x1 < x2 And y1 < y2 Then
List1.AddItem "x = indice*(x2-x1)+ x1;"
List1.AddItem "m1 = abs((x-x1)*indice/2);"
List1.AddItem "m2=abs((x2-x)*indice);"
List1.AddItem "a1=x1+(2*(x-x1)/3);"
List1.AddItem "a2=x+(x2-x)/2;"
List1.AddItem "xm1=m1+m2;"
List1.AddItem "xx1=((a1*m1)+(a2*m2))/xm1;"
List1.AddItem "xx = (xm*xx + xm1*xx1)/(xm+xm1);"
List1.AddItem "xm = xm+xm1;"
End If

'segundo caso
If x1 > x2 And y1 < y2 Then
List1.AddItem "x = x1 - indice*(x1-x2);"
List1.AddItem "xm1 = indice*(-2*x2+x1+x)/2;"
List1.AddItem "m1 = abs((x-x2)*indice);"
```

```
List1.AddItem "m2=abs((x1-x)*indice/2);"
List1.AddItem "a2=x+((x1-x)/3);"
List1.AddItem "a1=x2+(x-x2)/2;"
List1.AddItem "xm1=m1+m2;"
List1.AddItem "xx1=((a1*m1)+(a2*m2))/xm1;"
List1.AddItem "xx = (xm*xx + xm1*xx1)/(xm+xm1);"
List1.AddItem "xm = xm+xm1;"
End If
'terceiro caso
If y1 = y2 Then
List1.AddItem "xm1 = abs(indice*(x2-x1));"
List1.AddItem "xx1 = (x2+x1)/2;"
List1.AddItem "xx = (xm*xx + xm1*xx1)/(xm+xm1);"
List1.AddItem "xm = xm+xm1;"
End If
List1.AddItem "}"
Data6.Recordset.MoveNext
Loop

Data5.Recordset.MoveNext
Loop
If teste.Value = True Then
List1.AddItem "printf(" + Chr$(34) + "resposta " + campo + " = %f \n" + Chr$(34) + ", xx);"
List1.AddItem "gets (pausa);"
End If
If teste.Value = False Then
Data6.Recordset.MoveFirst
List1.AddItem Trim$(Data6.Recordset.Fields("msvas")) + "=xx;"
End If
'fim
List1.AddItem " "
```

## A.26 Função "defuzzy2()"

```
List1.AddItem "xx = 0;"
List1.AddItem "xm = 0;"
Data5.RecordSource = "select distinct mscam,msnom from mship where mscam=" + "'" + campo
+ "'"
```

```
Data5.Refresh
Data5.Recordset.MoveFirst
List1.AddItem "xmin =99999;"
List1.AddItem "xmax =-99999;"
'acha ponto xmin e xmax
Do While Not Data5.Recordset.EOF
Data6.RecordSource = "select * from mship where mscam=" + "'" + campo + "'" + " and msnom= " + "'" + Trim$(Data5.Recordset.Fields("msnom")) + "'" + " order by posx1"
Data6.Refresh
Data6.Recordset.MoveFirst
List1.AddItem "x1 =" + CStr(Data6.Recordset.Fields("posx1")) + ";"
List1.AddItem "x2 =" + CStr(Data6.Recordset.Fields("posx2")) + ";"
List1.AddItem "indice= IPe_" + Trim$(Data6.Recordset.Fields("mscam")) + "_" + Trim$(Data6.Recordset.Fields("msnom")) + ";"
List1.AddItem "if (indice > 0)"
List1.AddItem "{"
List1.AddItem "xmin=men(xmin, Min_" + Trim$(Data6.Recordset.Fields("mscam")) + "_" + Trim$(Data6.Recordset.Fields("msnom")) + ");"
List1.AddItem "xmax=max(xmax,Max_" + Trim$(Data6.Recordset.Fields("mscam")) + "_" + Trim$(Data6.Recordset.Fields("msnom")) + ");"
List1.AddItem "}"
Data5.Recordset.MoveNext
List1.AddItem " "
Loop
List1.AddItem " "
List1.AddItem " "
'integra
List1.AddItem "pre=(xmax-xmin)/50;"
List1.AddItem "xm=0;"
List1.AddItem "a=1;"
List1.AddItem "yant=0;"
List1.AddItem "ymax =0;"
List1.AddItem "for (xx=xmin;xx<=xmax;xx=xx+pre)"
List1.AddItem "{"
Data5.Recordset.MoveFirst
Do While Not Data5.Recordset.EOF
Data6.RecordSource = "select * from mship where mscam=" + "'" + campo + "'" + " and msnom= " + "'" + Trim$(Data5.Recordset.Fields("msnom")) + "'" + " order by posx1"
Data6.Refresh
Data6.Recordset.MoveFirst
```

```
Do While Not Data6.Recordset.EOF
List1.AddItem "indice= IPe_" + Trim$(Data6.Recordset.Fields("mscam")) + "_" + Trim$(Data6.Recordset.Fields("msnom")) + ";"
List1.AddItem "if (indice > 0)"
List1.AddItem "{"
List1.AddItem "x1 =" + CStr(Data6.Recordset.Fields("posx1")) + ";"
List1.AddItem "x2 =" + CStr(Data6.Recordset.Fields("posx2")) + ";"
List1.AddItem "if (men(x1,x2)<=xx) & (xx<=max(x1,x2))"
List1.AddItem "{"
List1.AddItem "y1 =" + CStr(Data6.Recordset.Fields("posy1")) + ";"
List1.AddItem "y2 =" + CStr(Data6.Recordset.Fields("posy2")) + ";"
List1.AddItem "  ymax1=indice;"
List1.AddItem "  if (ymax1 > ((xx -(x1))*((y2)-(y1))/((x2)-(x1)))+(y1))"
List1.AddItem "    ymax1 = ((xx -(x1))*((y2)-(y1))/((x2)-(x1)))+(y1);"
List1.AddItem "ymax=max(ymax,ymax1);"
List1.AddItem "}"
List1.AddItem "}"
Data6.Recordset.MoveNext
Loop
Data5.Recordset.MoveNext
Loop
List1.AddItem "xm=xm+(pre*(ymax+yant)/2);"
List1.AddItem "yant=ymax;"
List1.AddItem "ymax=0;"
List1.AddItem "matriz[a,1]=xx;"
List1.AddItem "matriz[a,2]=xm;"
List1.AddItem "a=a+1;"
List1.AddItem "}"
List1.AddItem " "
List1.AddItem " "
'retorna centro massa
List1.AddItem "flag=0;"
List1.AddItem "for b=1:a-1"
List1.AddItem "if (matriz[b,2]>xm/2 & flag==0)"
List1.AddItem "resp=matriz[b,1];"
List1.AddItem "flag=1;"
List1.AddItem "end;"
List1.AddItem "end;"
Data6.Recordset.MoveFirst
```

```
List1.AddItem Trim$(Data6.Recordset.Fields("msvas")) + "=xx;"
List1.AddItem " "
```

## A.27 Função "defuzzy3()"

```
variaveis que assumem posx1,...,posy2 de mship
Dim x1 As Double, x2 As Double
Dim y1 As Double, y2 As Double
Dim soma As Double, ipert As Double
List1.AddItem "xx = 0;"
List1.AddItem "xm = 0;"
List1.AddItem "xf1 = 0;"
List1.AddItem "mf1 = 0;"
Data5.RecordSource = "select distinct mscam,msnom from mship where mscam=" + "'" + campo + "'"
Data5.Refresh
Data5.Recordset.MoveFirst
Do While Not Data5.Recordset.EOF
Data6.RecordSource = "select * from mship where mscam=" + "'" + campo + "'" + " and msnom= " + "'" + Trim$(Data5.Recordset.Fields("msnom")) + "'" + " order by posx1"
Data6.Refresh
Data6.Recordset.MoveFirst
Do While Not Data6.Recordset.EOF
'encontra a area de um membership
x1 = Data6.Recordset.Fields("posx1")
x2 = Data6.Recordset.Fields("posx2")
y1 = Data6.Recordset.Fields("posy1")
y2 = Data6.Recordset.Fields("posy2")
List1.AddItem "x1 =" + CStr(Data6.Recordset.Fields("posx1")) + ";"
List1.AddItem "x2 =" + CStr(Data6.Recordset.Fields("posx2")) + ";"
List1.AddItem "y1 =" + CStr(Data6.Recordset.Fields("posy1")) + ";"
List1.AddItem "y2 =" + CStr(Data6.Recordset.Fields("posy2")) + ";"
List1.AddItem "indice= IPe_" + Trim$(Data6.Recordset.Fields("mscam")) + "_" + Trim$(Data6.Recordset.Fields("msnom")) + ";"
List1.AddItem ""
List1.AddItem ""
List1.AddItem "if (indice > 0)"
List1.AddItem "{"
'primeiro caso
```

```
If x1 < x2 And y1 < y2 Then
List1.AddItem "x = indice*(x2-x1)+ x1;"
List1.AddItem "m1 = abs((x-x1)*indice/2);"
List1.AddItem "m2=abs((x2-x)*indice);"
List1.AddItem "a1=x1+(2*(x-x1)/3);"
List1.AddItem "a2=x+(x2-x)/2;"
List1.AddItem "xm1=m1+m2;"
List1.AddItem "xx1=((a1*m1)+(a2*m2))/xm1;"
List1.AddItem "xx = (xm*xx + xm1*xx1)/(xm+xm1);"
List1.AddItem "xm = xm+xm1;"
End If
'segundo caso
If x1 > x2 And y1 < y2 Then
List1.AddItem "x = x1 - indice*(x1-x2);"
List1.AddItem "xm1 = indice*(-2*x2+x1+x)/2;"
List1.AddItem "m1 = abs((x-x2)*indice);"
List1.AddItem "m2=abs((x1-x)*indice/2);"
List1.AddItem "a2=x+((x1-x)/3);"
List1.AddItem "a1=x2+(x-x2)/2;"
List1.AddItem "xm1=m1+m2;"
List1.AddItem "xx1=((a1*m1)+(a2*m2))/xm1;"
List1.AddItem "xx = (xm*xx + xm1*xx1)/(xm+xm1);"
List1.AddItem "xm = xm+xm1;"
End If
'terceiro caso
If y1 = y2 Then
List1.AddItem "xm1 = abs(indice*(x2-x1));"
List1.AddItem "xx1 = (x2+x1)/2;"
List1.AddItem "xx = (xm*xx + xm1*xx1)/(xm+xm1);"
List1.AddItem "xm = xm+xm1;"
End If
List1.AddItem "}"
Data6.Recordset.MoveNext
Loop
List1.AddItem "if (indice>0)"
List1.AddItem "{"
List1.AddItem "xf1=(xx+xf1*mf1)/(mf1+1);"
List1.AddItem "mf1=mf1+1;"
```

```
List1.AddItem "xx = 0;"
List1.AddItem "xm = 0;"
List1.AddItem "}"
Data5.Recordset.MoveNext
Loop
If teste.Value = True Then
List1.AddItem "printf(" + Chr$(34) + "resposta " + campo + " = %f \n" + Chr$(34) + ", xf1);"
List1.AddItem "gets (pausa);"
End If
If teste.Value = False Then
Data6.Recordset.MoveFirst
List1.AddItem Trim$(Data6.Recordset.Fields("msvas")) + "=xf1;"
End If
'fim
List1.AddItem " "
```

## A.28 Função "defuzzy4()"

```
'variaveis que assumem posx1,...,posy2 de mship
Dim x1 As Double, x2 As Double
Dim y1 As Double, y2 As Double
Dim soma As Double, ipert As Double
List1.AddItem "xx = 0;"
List1.AddItem "xm = 0;"
List1.AddItem "ymax = 0;"
Data5.RecordSource = "select distinct mscam,msnom from mship where mscam=" + "'" + campo + "'"
Data5.Refresh
Data5.Recordset.MoveFirst
Do While Not Data5.Recordset.EOF
Data6.RecordSource = "select * from mship where mscam=" + "'" + campo + "'" + " and msnom= " + "'" + Trim$(Data5.Recordset.Fields("msnom")) + "'" + " order by posx1"
Data6.Refresh
Data6.Recordset.MoveFirst
Do While Not Data6.Recordset.EOF
'encontra a area de um membership
x1 = Data6.Recordset.Fields("posx1")
x2 = Data6.Recordset.Fields("posx2")
y1 = Data6.Recordset.Fields("posy1")
```

```
y2 = Data6.Recordset.Fields("posy2")
List1.AddItem "x1 =" + CStr(Data6.Recordset.Fields("posx1")) + ";"
List1.AddItem "x2 =" + CStr(Data6.Recordset.Fields("posx2")) + ";"
List1.AddItem "y1 =" + CStr(Data6.Recordset.Fields("posy1")) + ";"
List1.AddItem "y2 =" + CStr(Data6.Recordset.Fields("posy2")) + ";"
List1.AddItem "indice= IPe_" + Trim$(Data6.Recordset.Fields("mscam")) + "_" + Trim$(Data6.Recordset.Fields("msnom")) + ";"
List1.AddItem ""
List1.AddItem ""
List1.AddItem "if (indice > 0)"
List1.AddItem "{"
List1.AddItem "if (indice > ymax)"
List1.AddItem "{"
List1.AddItem "ymax=indice;"
List1.AddItem "xx = 0;"
List1.AddItem "xm = 0;"
List1.AddItem "}"
List1.AddItem "if (indice == ymax)"
List1.AddItem "{"
'primeiro caso
If x1 < x2 And y1 < y2 Then
List1.AddItem "x = indice*(x2-x1)+ x1;"
List1.AddItem "m1 = abs((x-x1)*indice/2);"
List1.AddItem "m2=abs((x2-x)*indice);"
List1.AddItem "a1=x1+(2*(x-x1)/3);"
List1.AddItem "a2=x+(x2-x)/2;"
List1.AddItem "xm1=m1+m2;"
List1.AddItem "xx1=((a1*m1)+(a2*m2))/xm1;"
List1.AddItem "xx = (xm*xx + xm1*xx1)/(xm+xm1);"
List1.AddItem "xm = xm+xm1;"
End If

'segundo caso

If x1 > x2 And y1 < y2 Then
List1.AddItem "x = x1 - indice*(x1-x2);"
List1.AddItem "xm1 = indice*(-2*x2+x1+x)/2;"
List1.AddItem "m1 = abs((x-x2)*indice);"
List1.AddItem "m2=abs((x1-x)*indice/2);"
```

```
List1.AddItem "a2=x+((x1-x)/3);"
List1.AddItem "a1=x2+(x-x2)/2;"
List1.AddItem "xm1=m1+m2;"
List1.AddItem "xx1=((a1*m1)+(a2*m2))/xm1;"
List1.AddItem "xx = (xm*xx + xm1*xx1)/(xm+xm1);"
List1.AddItem "xm = xm+xm1;"
End If

'terceiro caso

If y1 = y2 Then
List1.AddItem "xm1 = abs(indice*(x2-x1));"
List1.AddItem "xx1 = (x2+x1)/2;"
List1.AddItem "xx = (xm*xx + xm1*xx1)/(xm+xm1);"
List1.AddItem "xm = xm+xm1;"
End If
List1.AddItem "}"
List1.AddItem "}"

Data6.Recordset.MoveNext
Loop
Data5.Recordset.MoveNext
Loop

If teste.Value = True Then
List1.AddItem "printf(" + Chr$(34) + "resposta " + campo + " = %f \n" + Chr$(34) + ", xx);"
List1.AddItem "gets (pausa);"
End If

If teste.Value = False Then
Data6.Recordset.MoveFirst
List1.AddItem Trim$(Data6.Recordset.Fields("msvas")) + "=xx;"
End If
'fim
List1.AddItem " "
```

# B

# Apêndice

*Neste apêndice será mostrado a rotina fuzzy gerado no capítulo 3*

## B.1 Código Fuzzy Gerado

```
% BLOCO 1 - INICIALIZACAO DOS INDICES DE  PERTINENCIA */
%        - INICIALIZACAO DOS LIMITES DOS MEMBERSHIPS  */
function [aj_1]=fuzzy(temp_1,umi_1)
aag=0;
Pes_temperatura = 1;
Pes_umidade = 1;
Pes_ajuste = 1;

IPe_ajuste_esfria = 0 ;
Min_ajuste_esfria = -100;
Max_ajuste_esfria = 0;
IPe_ajuste_esquenta = 0 ;
Min_ajuste_esquenta = 0;
Max_ajuste_esquenta = 100;
IPe_ajuste_mantem = 0 ;
Min_ajuste_mantem = -60;
Max_ajuste_mantem = 60;
IPe_temperatura_fria = 0 ;
Min_temperatura_fria = 0;
Max_temperatura_fria = 22.5;
IPe_temperatura_normal = 0 ;
```

```
Min_temperatura_normal = 15.75;
Max_temperatura_normal = 29.25;
IPe_temperatura_quente = 0 ;
Min_temperatura_quente = 22.5;
Max_temperatura_quente = 45;
IPe_umidade_alta = 0 ;
Min_umidade_alta = 50;
Max_umidade_alta = 90;
IPe_umidade_baixa = 0 ;
Min_umidade_baixa = 10;
Max_umidade_baixa = 50;
IPe_umidade_normal = 0 ;
Min_umidade_normal = 26;
Max_umidade_normal = 74;

% FIM DO BLOCO 1 */
% BLOCO 2 - CALCULO DOS INDICES DE PERTINENCIA */

if (temp_1 > 0 & temp_1 <= 15.75 )
  if (IPe_temperatura_fria < ((temp_1-(0))*((1)-(1))/((15.75)-(0)))+(1))
    IPe_temperatura_fria = ((temp_1-(0))*((1)-(1))/((15.75)-(0)))+(1);
end;
end;

if (umi_1 > 10 & umi_1 <= 26 )
  if (IPe_umidade_baixa < ((umi_1-(10))*((1)-(1))/((26)-(10)))+(1))
    IPe_umidade_baixa = ((umi_1-(10))*((1)-(1))/((26)-(10)))+(1);
end;
end;

if (temp_1 > 15.75 & temp_1 <= 22.5 )
  if (IPe_temperatura_normal < ((temp_1-(15.75))*((1)-(0))/((22.5)-(15.75)))+(0))
    IPe_temperatura_normal = ((temp_1-(15.75))*((1)-(0))/((22.5)-(15.75)))+(0);
end;
end;

if (temp_1 > 22.5 & temp_1 <= 29.25 )
  if (IPe_temperatura_quente < ((temp_1-(22.5))*((1)-(0))/((29.25)-(22.5)))+(0))
```

```
   IPe_temperatura_quente = ((temp_1-(22.5))*((1)-(0))/((29.25)-(22.5)))+(0);
 end;
 end;

 if (temp_1 > 15.75 & temp_1 <= 22.5 )
  if (IPe_temperatura_fria < ((22.5-temp_1)*((1)-(0))/((22.5)-(15.75)))+(0))
   IPe_temperatura_fria = ((22.5-temp_1)*((1)-(0))/((22.5)-(15.75)))+(0);
 end;
 end;

 if (umi_1 > 26 & umi_1 <= 50 )
  if (IPe_umidade_normal < ((umi_1-(26))*((1)-(0))/((50)-(26)))+(0))
   IPe_umidade_normal = ((umi_1-(26))*((1)-(0))/((50)-(26)))+(0);
 end;
 end;

 if (temp_1 > 22.5 & temp_1 <= 29.25 )
  if (IPe_temperatura_normal < ((29.25-temp_1)*((1)-(0))/((29.25)-(22.5)))+(0))
   IPe_temperatura_normal = ((29.25-temp_1)*((1)-(0))/((29.25)-(22.5)))+(0);
 end;
 end;

 if (temp_1 > 29.25 & temp_1 <= 45 )
  if (IPe_temperatura_quente < ((45-temp_1)*((1)-(1))/((45)-(29.25)))+(1))
   IPe_temperatura_quente = ((45-temp_1)*((1)-(1))/((45)-(29.25)))+(1);
 end;
 end;

 if (umi_1 > 26 & umi_1 <= 50 )
  if (IPe_umidade_baixa < ((50-umi_1)*((1)-(0))/((50)-(26)))+(0))
   IPe_umidade_baixa = ((50-umi_1)*((1)-(0))/((50)-(26)))+(0);
 end;
 end;

 if (umi_1 > 50 & umi_1 <= 74 )
  if (IPe_umidade_alta < ((umi_1-(50))*((1)-(0))/((74)-(50)))+(0))
   IPe_umidade_alta = ((umi_1-(50))*((1)-(0))/((74)-(50)))+(0);
 end;
```

```
end;

if (umi_1 > 50 & umi_1 <= 74 )
  if (IPe_umidade_normal < ((74-umi_1)*((1)-(0))/((74)-(50)))+(0))
    IPe_umidade_normal = ((74-umi_1)*((1)-(0))/((74)-(50)))+(0);
end;
end;

if (umi_1 > 74 & umi_1 <= 90 )
  if (IPe_umidade_alta < ((90-umi_1)*((1)-(1))/((90)-(74)))+(1))
    IPe_umidade_alta = ((90-umi_1)*((1)-(1))/((90)-(74)))+(1);
end;
end;
% FIM DO BLOCO 2 */


% BLOCO 3 - APLICACAO DAS REGRAS */


if ((Min_temperatura_quente <= temp_1 & temp_1 <= Max_temperatura_quente))
matriz=[IPe_temperatura_quente];
retorno= min(matriz);
IPe_ajuste_esfria = max(IPe_ajuste_esfria ,retorno);
end;


if ((Min_temperatura_fria <= temp_1 & temp_1 <= Max_temperatura_fria))
matriz=[IPe_temperatura_fria];
retorno= min(matriz);
IPe_ajuste_esquenta = max(IPe_ajuste_esquenta ,retorno);
end;


if ((Min_temperatura_normal <= temp_1 & temp_1 <= Max_temperatura_normal) &
(Min_umidade_alta <= umi_1 & umi_1 <= Max_umidade_alta))
matriz=[IPe_temperatura_normal IPe_umidade_alta];
retorno= min(matriz);
IPe_ajuste_esfria = max(IPe_ajuste_esfria ,retorno);
end;
```

```
if ((Min_temperatura_normal <= temp_1 & temp_1 <= Max_temperatura_normal) & (Min_umidade_normal <= umi_1 & umi_1 <= Max_umidade_normal))
matriz=[IPe_temperatura_normal IPe_umidade_normal];
retorno= min(matriz);
IPe_ajuste_mantem = max(IPe_ajuste_mantem ,retorno);
end;

if ((Min_temperatura_fria <= temp_1 & temp_1 <= Max_temperatura_fria) & (Min_umidade_baixa <= umi_1 & umi_1 <= Max_umidade_baixa))
matriz=[IPe_temperatura_fria IPe_umidade_baixa];
retorno= min(matriz);
IPe_ajuste_esquenta = max(IPe_ajuste_esquenta ,retorno);
end;
% FIM DO BLOCO 3 */
% INICIO DO BLOCO 4 - DEFUZZYFICACAO*/
xx = 0;
xm = 0;
xmin =99999;
xmax =-99999;
x1 =-100;
x2 =-60;
indice= IPe_ajuste_esfria;
if (indice > 0)
xmin=min(xmin, Min_ajuste_esfria);
xmax=max(xmax,Max_ajuste_esfria);
end
x1 =0;
x2 =60;
indice= IPe_ajuste_esquenta;
if (indice > 0)
xmin=min(xmin, Min_ajuste_esquenta);
xmax=max(xmax,Max_ajuste_esquenta);
end;
x1 =-60;
x2 =0;
indice= IPe_ajuste_mantem;
```

```
if (indice > 0)
xmin=min(xmin, Min_ajuste_mantem);
xmax=max(xmax,Max_ajuste_mantem);
end;
pre=(xmax-xmin)/50;
xm=0;
a=1;
yant=0;
ymax =0;
for xx=xmin:pre:xmax
indice= IPe_ajuste_esfria ;
if (indice > 0)
x1 =-100;
x2 =-60;
if (min(x1,x2)<=xx) & (xx<=max(x1,x2))
y1 =1;
y2 =1;
  ymax1=indice;
  if (ymax1 > ((xx -(x1))*((y2)-(y1))/((x2)-(x1)))+(y1))
    ymax1 = ((xx -(x1))*((y2)-(y1))/((x2)-(x1)))+(y1);
  end;
ymax=max(ymax,ymax1);
end;
end;
indice= IPe_ajuste_esfria ;
if (indice > 0)
x1 =0;
x2 =-60;
if (min(x1,x2)<=xx) & (xx<=max(x1,x2))
y1 =0;
y2 =1;
  ymax1=indice;
  if (ymax1 > ((xx -(x1))*((y2)-(y1))/((x2)-(x1)))+(y1))
    ymax1 = ((xx -(x1))*((y2)-(y1))/((x2)-(x1)))+(y1);
  end;
ymax=max(ymax,ymax1);
end;
end;
```

```
indice= IPe_ajuste_esquenta ;
if (indice > 0)
x1 =0;
x2 =60;
if (min(x1,x2)<=xx) & (xx<=max(x1,x2))
y1 =0;
y2 =1;
  ymax1=indice;
  if (ymax1 > ((xx -(x1))*((y2)-(y1))/((x2)-(x1)))+(y1))
    ymax1 = ((xx -(x1))*((y2)-(y1))/((x2)-(x1)))+(y1);
  end;
ymax=max(ymax,ymax1);
end;
end;
indice= IPe_ajuste_esquenta ;
if (indice > 0)
x1 =100;
x2 =60;
if (min(x1,x2)<=xx) & (xx<=max(x1,x2))
y1 =1;
y2 =1;
  ymax1=indice;
  if (ymax1 > ((xx -(x1))*((y2)-(y1))/((x2)-(x1)))+(y1))
    ymax1 = ((xx -(x1))*((y2)-(y1))/((x2)-(x1)))+(y1);
  end;
ymax=max(ymax,ymax1);
end;
end;
indice= IPe_ajuste_mantem ;
if (indice > 0)
x1 =-60;
x2 =0;
if (min(x1,x2)<=xx) & (xx<=max(x1,x2))
y1 =0;
y2 =1;
  ymax1=indice;
  if (ymax1 > ((xx -(x1))*((y2)-(y1))/((x2)-(x1)))+(y1))
    ymax1 = ((xx -(x1))*((y2)-(y1))/((x2)-(x1)))+(y1);
```

```
  end;
ymax=max(ymax,ymax1);
end;
end;
indice= IPe_ajuste_mantem ;
if (indice > 0)
x1 =60;
x2 =0;
if (min(x1,x2)<=xx) & (xx<=max(x1,x2))
y1 =0;
y2 =1;
  ymax1=indice;
  if (ymax1 > ((xx -(x1))*((y2)-(y1))/((x2)-(x1)))+(y1))
    ymax1 = ((xx -(x1))*((y2)-(y1))/((x2)-(x1)))+(y1);
  end;
ymax=max(ymax,ymax1);
end;
end;
xm=xm+(pre*(ymax+yant)/2);
yant=ymax;
ymax=0;
matriz(a,1)=xx;
matriz(a,2)=xm;
a=a+1;
end;

flag=0;
for b=1:a-1
if (matriz(b,2)>xm/2 & flag==0)
xx=matriz(b,1);
flag=1;
end;
end;
aj_1=xx+aag;
```

# Índice

# Índice de figuras e tabelas

## 1



## 2

## 3



## 4