UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA QUÍMICA

ÁREA DE CONCENTRAÇÃO SISTEMAS DE PROCESSOS QUÍMICOS E INFORMÁTICA

DESENVOLVIMENTO E IMPLEMENTAÇÃO DE CONTROLADORES NEBULOSOS EM UMA COLUNA PILOTO DE DESTILAÇÃO EM BATELADA

Autor: Renato Dutra Pereira Filho.
Orientadora: Profª. Drª. Ana Maria Frattini Fileti.
Co-Orientador: Prof. Dr. João Alexandre F. R. Pereira.

Dissertação de Mestrado

Campinas – SP Brasil

Março - 1999

UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA QUÍMICA

DESENVOLVIMENTO E IMPLEMENTAÇÃO DE CONTROLADORES NEBULOSOS EM UMA COLUNA PILOTO DE DESTILAÇÃO EM BATELADA

Autor: Renato Dutra Pereira Filho
Orientadora: Profª. Drª. Ana Maria Frattini Fileti.
Co-Orientador: Prof. Dr. João Alexandre F. R. Pereira.

Dissertação de Mestrado apresentada à Faculdade de Engenharia Química como parte dos requisitos exigidos para a obtenção do título de Mestre em Engenharia Química.

Área de Concentração: Sistemas de Processos Químicos e Informática.

Campinas – SP Brasil

Março - 1999

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

P414d

Pereira Filho, Renato Dutra
Desenvolvimento e implementação de controladores nebulosos em uma coluna piloto de destilação em batelada. / Renato Dutra Pereira Filho.—Campinas, SP: [s.n.], 1999.

Orientadores: Ana Maria Frattini Fileti, João Alexandre F. R. Pereira.
Dissertação (mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Química

1. Destilação. 2. Sistemas difusos. 3. Controle de processos químicos. 4. Sistemas inteligentes de controle. I. Fileti, Ana Maria Frattini. II. Pereira, João Alexandre F. R. III. Universidade Estadual de Campinas. Faculdade de Engenharia Química. IV. Título.

Dissertação de Mestrado defendida e aprovada em 26 de março de 1999 pela banca examinadora constituída pelos professores doutores:

Profª. Drª. Ana Maria Frattini Fileti
Orientadora

Prof. Dr. (titular) Claudio Augusto Oller do Nascimento

Profª. Drª. (titular) Sandra Lúcia da Cruz

DEDICATÓRIA

À Cláudia, esposa companheira que me fortalece com sua energia e amor.

Esta versão corresponde a redação final da Dissertação de Mestrado em Engenharia Química defendida pelo Eng. Renato Dutra Pereira Filho e aprovada pela Comissão Julgadora em 26/02/1999.

Profª. Drª. Ana Maria Frattini Fileti
Orientadora

AGRADECIMENTOS

Este trabalho não teria sido realizado sem a ajuda de muitas pessoas e instituições às quais presto minha homenagem:

Aos meus pais, Renato e Diva, e à minha irmã, Márcia, pelo afeto, incentivo e compreensão durante toda minha vida.

Ao meu sogro, Dinarte, pela confiança e amizade.

À minha orientadora, Profª. Drª Ana Maria Frattini Fileti, que percorreu esta jornada ao meu lado, me auxiliando sempre que se fez necessário.

Ao Prof. Dr. João Alexandre F. R. Pereira pela colaboração na parte experimental do trabalho.

A todos os professores do Departamento de Engenharia de Sistemas Químicos e Informática, que colaboraram direta ou indiretamente na execução deste trabalho.

A todos os colegas pós-graduandos da FEQ, em especial ao Marcone, ao Frede, e ao Luciano.

Aos colegas professores e funcionários do Departamento de Química da Fundação Universidade do Rio Grande, pelos exemplos diários de dedicação e companheirismo.

Ao Eng. Márcio Sarres Pessoa pelas notícias, dicas e incentivos, através de longas conversas via telefone.

À CAPES pelo auxílio financeiro através de bolsa PICDT.

À Universidade Pública Gratuita e de Qualidade, por apesar dos reveses ainda resistir.

ÍNDICE

LISTA DE FIGURAS E GRÁFICOS

LISTA DE TABELAS

NOMENCLATURA

$(\Delta H)_{n+1}$	calor latente de vaporização da mistura no refervedor;
$(\Delta H)_0$	calor latente de vaporização da mistura no condensador;
"a"	centro da função de pertinência;
A, B	conjuntos nebulosos de exemplo;
a_{de}	limite da variação do erro;
a_e	limite do erro;
A_p	área do prato;
D	fluxo de destilado;
e	constante da equação de Francis;
e_k	erro na variável controlada no instante k;
e_{k-1}	erro na variável controlada no instante k-1;
erro	erro entre a fração molar de n-hexano no topo e o "setpoint";
F	subconjunto nebuloso;
f_i^o	fugacidade de ref. do comp. i puro à temperatura e pressão de equilíbrio;
fpa(i,1)	extremo esquerdo da função de pertinência;
fpa(i,2)	centro da função de pertinência;
fpa(i,3)	extremo direito da função de pertinência;
h_j	entalpia total molar da fase líquida no estágio j;
H_j	entalpia total molar da fase vapor no estágio j;
h_v	altura do vertedouro;
k	instante atual de amostragem;
k_c	constante usada no termo prop. da expressão do controlador PID vel.;
K_c	ganho proporcional;
k_D	constante usada no termo der. da expressão do controlador PID vel.;
k_{de}	fator de escala da variação do erro;
k_{du}	fator de escala da ação;
k_e	fator de escala da variação do erro;
k_I	constante usada no termo int. da expressão do controlador PID vel.;
L	fluxo de líquido que retorna à coluna;

L_j	fluxo de líquido que deixa o estágio j;
L_w	largura transversal do vertedouro;
M_j	acúmulo molar de líquido no estágio j;
n	número funções de pertinência da ação do controlador ativadas;
P	pressão de operação;
pertinência_ação(i,1)	suporte ou base da função de pertinência triangular "i";
pertinência_ação(i,2)	centro da função de pertinência;
P_i^s	pressão de saturação calculada pela equação de Antoine;
Q_1	somatório do erro quadrático;
Q_2	maior desvio positivo relativo;
Q_c	fluxo de calor retirado no condensador;
Q_R	fluxo de calor fornecido ao refervedor;
R	razão de refluxo externa;
r	razão de refluxo interna;
R_c	regra de controle;
r_I	grau de ativação de cada regra;
T	temperatura;
t	tempo de operação;
t_A	intervalo de amostragem;
t_D	tempo em que a válvula passa na posição destilado;
T_{fundo}	temperatura de fundo da coluna;
t_L	intervalo de tempo em que a válvula permanece na posição refluxo;
T_{topo}	temperatura de topo da coluna ;
u	elemento genérico de U ;
U	universo de discurso;
V	fluxo de vapor no topo da coluna;
V_j	fluxo de vapor que deixa o estágio j;
V_t	sinal de voltagem proveniente do termopar;
x	fração molar na fase líquida;
X, Y, Z	variáveis manipulas pelas regras nebulosas;
x_{ij}	fração molar do componente i na fase líquida do estágio j;

y	fração molar na fase vapor;
$y^*_{i,j}$	fração molar de equilíbrio do componente i na fase vapor do estágio j;
y_{ij}	fração molar do componente i na fase vapor do estágio j;
y_{max}	maior valor de conc. de n-hexano após 1ª atuação do controlador;
Z_I	centro da função de pertinência "i";
Δe_k	variação do erro no instante k de amostragem;
Δe_{k-1}	variação do erro no instante k-1 de amostragem;
Δerro	diferença entre o erro no instante atual e o erro no instante anterior;
Δp_k	variação na variável de saída do controlador no instante atual k;
Δt	intervalo de amostragem;
Δu^*_k	variação na ação de controle normalizada;
Δu_k	variação na ação de controle não normalizada;
$\Delta^2 e_k$	2ª variação do erro no instante k de amostragem;
α, β	distâncias entre os extremos e o centro da função de pertinência;
ϕ_i	coeficiente de fugacidade do componente i;
γ_i	coeficiente de atividade do componente i;
μ_A, μ_B	grau de pertinência de A e B;
μ_F	grau de pertinência do subconjunto F;
μ_{Rc}	grau de ativação da regra Rc;
ρ_j	massa específica molar da mistura líquida no prato j;
σ	parâmetro de largura das funções de pertinência gaussianas;
τ_D	constante de tempo derivativo;
τ_I	constante de tempo integral;

RESUMO

DUTRA PEREIRA,R. **Desenvolvimento e implementação de controladores nebulosos em uma coluna piloto de destilação em batelada.** Campinas: FEQ, UNICAMP, 1999. Dissertação (Mestrado) - Faculdade de Engenharia Química, Universidade Estadual de Campinas, 1999. 169 p.

Esta dissertação estudou a aplicação da lógica nebulosa no controle de processos químicos e descreveu o desenvolvimento e a implementação de controladores nebulosos aplicados à destilação em batelada de misturas binárias. Em uma primeira etapa os algoritmos de controle nebulosos, análogos aos controladores P, PI e PID tradicionais, foram desenvolvidos, codificados em linguagem de programação FORTRAN e testados utilizando um simulador dinâmico da destilação em batelada, disponível em linguagem de programação C. A etapa de simulação permitiu avaliar os parâmetros de projeto dos controladores nebulosos desenvolvidos, a fim de permitir um melhor domínio das influências e interações destes parâmetros com o desempenho dos controladores nebulosos. Os resultados em simulação demostraram que os parâmetros de escala têm grande influência na resposta dinâmica do controlador, tanto em termos de qualidade do desempenho do controlador com respeito a manutenção do valor de referência e tempo de resposta, quanto à ocorrência de oscilações. A etapa experimental do trabalho consistiu em utilizar os algoritmos de controle nebulosos na automação da operação de uma coluna piloto de destilação em batelada. Foram separadas misturas binárias compostas de n-hexano e n-heptano. A composição no topo e no fundo da coluna foi calculada através de inferenciação a partir dos valores de temperatura, usando dados de equilíbrio líquido-vapor. Foram usadas duas políticas de produção, para especificar a concentração do destilado: uma usando valor de referência fixo e outra usando trajetórias de valor de referência, o que permitiu trabalhar com vários "cortes" de destilado durante a destilação. Sob condições de "setpoint" fixo os controladores nebulosos do tipo PI e PID foram comparados com os controladores PI velocidade e PID posição e velocidade, respectivamente. O desempenho dos controladores nebulosos foi superior ao dos controladores tradicionais. Os controladores nebulosos PI e PID mostraram resultados muito bons , tanto em termos da variável controlada (concentração de n-hexano no topo) como no perfil da variável manipulada, razão de refluxo, onde se alcançaram perfis mais suaves sem ações bruscas, com valores em média menores do que aqueles obtidos pelos controladores tradicionais. Quando dos experimentos usando trajetória de "setpoint" os controladores nebulosos conseguiram manter a trajetória especificada, apesar dos seus parâmetros serem fixos.

Palavras Chave:

Destilação, Sistemas Difusos, Controle de Processos Químicos.

ABSTRACT

Development and implementation of fuzzy controllers in a pilot batch distillation column.

This work is concerned in the application of fuzzy logic in chemical process control. Fuzzy controllers development and implementation were described and applied to a binary system batch distillation. Algorithms of fuzzy control that are similar to conventional P, PI, PID controllers were developed and implemented in FORTRAN. First of all, algorithms were tested coupled to a batch distillation dynamic simulator, available in C language. Both influence and interaction of design parameters were evaluated over fuzzy controller performance. Simulation results showed that scale parameters hardly affect the controller dynamic response: set point maintenance, time response and oscillations showed different behaviours. Fuzzy control algorithms were then experimentally used in the on line operation of a pilot batch distillation column. N-hexane/n-heptane mixtures were distilled. Top and bottom compositions were obtained from temperature measurements and inferred through liquid-vapour equilibrium data. Two different composition policies were used for distillate withdrawn: a fixed set point composition and a predetermined set point trajectory of top composition. Different distillate cuts could be obtained during a batch operation under second strategy. Using fixed set point strategy, PI and PID fuzzy controllers performances were compared to conventional PI (velocity form) and PID (position and velocity form) respectively. Experimental results showed that fuzzy controllers performance is superior to conventional P, PI and PID controllers. Good results were obtained to controlled (n-hexane molar fraction) and manipulated (reflux ratio) variable profiles. Smooth curves of reflux ratio were obtained. Average implemented values of reflux ratio from fuzzy controllers showed to be smaller than those obtained from conventional controllers. In spite of using fixed design parameters, fuzzy controllers were able to follow variable set point strategy.

Key words:

Distillation, Fuzzy Systems, Chemical Process Control.

1. INTRODUÇÃO

As indústrias químicas fazem uso das operações unitárias para separarem misturas nas diversas etapas dos seus processos. Dentre as operações unitárias, a mais estudada e utilizada é a destilação, onde há a separação de diferentes substâncias de acordo com as suas diferentes volatilidades.

A operação de uma coluna de destilação pode ser feita de forma contínua ou descontínua (batelada). Na maioria dos processos químicos é mais lucrativo utilizar a operação contínua, onde se processam grandes quantidades de material, evitando perturbações e paradas. Essa situação se modifica para processos em pequena escala, ou sob condições em que as manutenções sejam freqüentes, por exemplo, na destilação de substâncias corrosivas ou altamente viscosas.

A destilação em batelada é mais versátil, pois pode separar uma variedade maior de misturas com composições diferentes , possibilitando produzir vários tipos de produtos, ou montar plantas de múltiplo propósito, mais adequadas para satisfazer as exigências do mercado atual.

Por suas características a destilação em batelada é usada na separação de pequenas quantidades de substâncias de alto valor intrínseco, tais como matérias-primas para a indústria de química fina (ácidos graxos, por exemplo), produtos farmacêuticos e na recuperação de resíduos industriais.

Por sua operação descontínua, a destilação em batelada é um processo transiente, ou seja, as variáveis de estado não apresentam valor constante ao longo do tempo. Isso implica em parâmetros operacionais variáveis, e baixo grau de automação, motivando o uso de especialistas na operação.

Além dessa característica de transitoriedade, a destilação batelada apresenta um comportamento não linear, dificultando ainda mais o projeto de sistemas de controle.

Muitos estudos enfocaram a destilação em batelada, mas se dedicaram muito mais ao desenvolvimento da modelagem matemática e ao estudo das políticas de operação ótimas, do que desenvolver sistemas "on-line" de controle.

O estudo do controle da destilação em batelada se justifica devido suas características de transitoriedade e não linearidade, que prejudicam o desempenho dos controladores por retroalimentação tradicionais. Estes controladores apresentam parâmetros fixos e portanto não conseguem manter um bom desempenho ao longo de toda a destilação.

Além disso, com o aumento do grau de automação da destilação batelada, é possível obter produtos com especificação fixa , a um custo energético inferior e com melhores condições de segurança.

Algumas alternativas para o desenvolvimento e melhoria do controle "on-line" da destilação em batelada já foram estudadas, na UNICAMP: controlador programável adaptativo (Cunha,1996), controle preditivo baseado em redes neurais (Fileti,1996), controle auto ajustável (Pedrosa, 1998), controle por antecipação baseado no balanço de massa (Oisovici,1998).

O controle de processos é uma das áreas com um grande potencial para aplicações dos chamados sistemas "inteligentes", incluindo as redes neurais artificiais (Fileti, 1996 e Pedrosa, 1998) e o controle nebuloso, como alternativa à automação da destilação em batelada.

O controle nebuloso é utilizado em produtos eletrônicos de consumo, mas só recentemente tem sido aventado como solução para alguns problemas de modelagem e controle na engenharia química.

Optou-se então por realizar uma abordagem do problema da destilação em batelada usando esses controladores nebulosos, originários no trabalho de Zadeh (1965). Nestes através da aplicação de conceitos da lógica nebulosa e do conhecimento de um especialista humano,

desenvolve-se um controlador baseado em regras "se-então", embutindo o conhecimento do operador na automação do processo.

1.1 Organização da Dissertação

No próximo capítulo é feita a revisão da literatura envolvendo a destilação batelada, e as aplicações da lógica nebulosa em engenharia química.

No terceiro capítulo é descrita a unidade piloto de destilação em batelada, o sistema de aquisição de dados e o sistema de atuação utilizados na etapa de acompanhamento experimental dos controladores desenvolvidos. A modelagem matemática que foi utilizada para compor a simulação computacional por Fileti (1992), e serviu de base para o desenvolvimento dos controladores nebulosos, também é sucintamente descrita.

No quarto capítulo são apresentados os aspectos mais importantes dos sistemas nebulosos: fundamentação teórica, estrutura, parâmetros de projeto, tipos; bem como uma descrição breve da implementação computacional dos mesmos.

O quinto capítulo traz a análise dos resultados preliminares obtidos em simulação, durante os estudos dos efeitos no desempenho do controlador dos diversos parâmetros dos controladores nebulosos.

A seguir, no sexto capítulo, está a apresentação dos experimentos executados, na coluna piloto de destilação em batelada, os resultados alcançados e a análise do desempenho dos controladores testados na destilação de misturas binárias.

Por fim, no sétimo capítulo, estão as conclusões finais e as sugestões para trabalhos posteriores.

2. REVISÃO DA LITERATURA

2.1 Introdução

A operação de uma coluna de destilação em batelada é composta de duas etapas distintas: a partida ("start-up") que compreende carregar o refervedor com a mistura e o aquecimento com o sistema sob refluxo total, até que sejam obtidos os perfis iniciais de concentração e temperatura (regime estacionário); e a destilação propriamente dita, a partir do momento em que é dada vazão a produção de destilado através da válvula que controla o refluxo para a coluna. A concentração depende da composição do material restante na coluna e da razão de refluxo que determina a quantidade de líquido que retorna para a coluna.

Existem várias estratégias de operação para a coluna de destilação em batelada: operação à composição constante, operação à refluxo constante e operação segundo um planejamento ótimo para a razão de refluxo.

A operação à composição constante implica em manipular a razão de refluxo a fim de enriquecer o destilado a medida que o componente de interesse esgota-se na coluna. Isso significa utilizar no início da destilação binária valores pequenos de razão de refluxo (pouco retorno de líquido à coluna) e ir aumentando, até utilizar-se valores altos de razão de refluxo, o que significa retornar todo o destilado à coluna, decretando o final da destilação.

Operar em refluxo constante significa estar em malha aberta e determina que a concentração de destilado varie ao longo do transcorrer da corrida, iniciando em valores altos e terminando com valores de concentração abaixo do desejado. Por compensação, o valor médio da composição de produto, obtido dentro do recipiente de recolhimento, encontra-se dentro das especificações de produto. Entretanto, apesar de ser o modo de operação mais simples e barato, a recuperação normalmente é ineficiente.

O planejamento ótimo de razão de refluxo segue critérios de desempenho, visando ou minimizar o tempo para a produção de determinada quantidade de produto, ou produzir o

máximo de produto dentro da especificação, no mínimo tempo ou produzir visando o lucro máximo.

Os trabalhos consultados na bibliografia referentes à destilação em batelada podem ser agrupados em: estudos de modificações no projeto, simulação, otimização, controle e inferência de composição. Dar-se-á a seguir um panorama dos trabalhos desenvolvidos dentro destes grandes grupos abrangendo a destilação em batelada, juntamente com as aplicações da lógica nebulosa em engenharia química.

2.2 Alterações no Projeto de Coluna

A fim de melhorar as condições de produtividade da destilação em batelada, vários trabalhos sugeriram modificações no projeto convencional da coluna.

Uma destas modificações é a chamada destilação invertida, quando se carrega a mistura a ser separada no condensador, de onde é continuamente alimentada ao prato de topo. No refervedor da coluna há a vaporização de parte do líquido e a outra parte é retirada como produto. Pela retirada efetuar-se no fundo, há a inversão na seqüência de retirada, recolhendo-se primeiramente o componente mais pesado, após o segundo e assim por diante.

Sorensen e Skogestad (1996) efetuaram simulações computacionais onde compararam a destilação invertida com a destilação batelada convencional, e chegaram a conclusão que cada um dos dois tipos de montagem se adapta a um tipo de composição da mistura a ser destilada. Quando a mistura é rica no componente mais pesado, o tempo de destilação é menor na destilação invertida. Quando a mistura é rica no componente mais leve, a montagem convencional implica num tempo de batelada menor.

Outro rearranjo na montagem da coluna sugerido nas simulações de Macchietto (1996) é o uso de um vaso intermediário entre duas seções de separação. A alimentação é efetuada neste vaso e há a retirada simultânea no fundo e no topo da coluna. As dificuldades de montagem e

operação são fatores que determinam a inexistência de trabalhos experimentais neste tipo de montagem.

2.3 Simulação Computacional

Domenech e Enjalbert (1981) desenvolveram um simulador para a destilação em batelada, abrangendo misturas complexas, acúmulo de líquido negligenciável ou não no prato, refluxo constante ou variável.

Fredenslund e Galindez (1988) desenvolveram um simulador da destilação em batelada multicomponente onde o estado transiente é simulado como uma sucessão de um número finito de estados estacionários de curta duração. Para cada intervalo de tempo, a solução é alcançada usando um modelo matemático correspondente a destilação contínua, levando em consideração o acúmulo de líquido nos pratos.

Fileti (1992) desenvolveu um programa computacional para simulação de uma destilação em batelada, com múltiplos estágios e componentes, onde foram consideradas as variações de líquido no prato e das vazões molares na coluna.

Barolo e Guarise (1996) propuseram um método simplificado para a simulação da destilação em batelada multicomponente. O método leva em consideração que uma dada coluna possa ser aproximada por um número infinito de estágios, e então a equação de Underwood, desenvolvida para o projeto rápido de colunas de destilação contínua, é usada para avaliar as composições de produto durante a batelada. O método foi comparado com um modelo tradicional, onde foram consideradas volatilidades relativas constantes, fluxos molares constantes e acúmulo de líquido negligenciável.

2.4 Otimização da Operação

Converse e Gross (1963) aplicaram o princípio do máximo contínuo de Pontryagin para obter o planejamento ótimo para a razão de refluxo, a fim de maximizar a quantidade total de destilado com a concentração especificada. Vários autores, tais como, Coward (1967) e Robinson (1970) utilizaram o modelo desenvolvido por Converse e Gross (1963) para minimizar o tempo de destilação necessário para uma dada quantidade de destilado.

Luyben (1971) abordou os aspectos práticos da operação, revisando os principais trabalhos sobre a destilação batelada desde a década de 40 e propondo um índice para caracterizar a operação e o projeto de colunas em batelada: o fator de capacidade.

Diwekar, Malik e Madhavan (1987) descreveram um procedimento para obter o refluxo ótimo ou a estratégia ótima para a produção de destilado, também usando o princípio do máximo contínuo de Pontryagin. Para isso fizeram uso de um modelo reduzido para a análise e projeto da coluna. Esse modelo era composto das correlações de Fenske, Underwood e Gilliand.

Shinskey (1988) apresentou uma comparação entre os três modos de operação: razão de refluxo constante, composição constante e máxima recuperação de produto, apontando as seguintes desvantagens na operação à razão de refluxo constante: se a razão de refluxo é relativamente baixa, haverá pouca separação, e a retirada de destilado deverá ser interrompida quando ainda houver grande quantidade do componente mais volátil no interior da coluna; ou seja, a recuperação do componente mais volátil será pequena. Se aumentarmos a razão de refluxo para melhorar a eficiência da separação e em conseqüência a recuperação do componente mais leve, os custos variáveis (consumo de energia e água de resfriamento) aumentarão significativamente.

Novamente Luyben (1988) retomou o uso do fator de capacidade, analisando o efeito das misturas intermediárias produzidas ao longo da destilação.

Marmol e Luyben (1990) prosseguiram o trabalho, utilizando novas estratégias de aproveitamento das frações intermediárias, comparando-as com os trabalhos anteriores. Observaram que em purezas elevadas de destilado (95 a 99%) o melhor consistia em separar as frações intermediárias e, desde que houvesse a quantidade suficiente, destilá-las novamente.

A operação a composição constante é a maneira de obter frações de destilado de concentração determinada ("cortes") . Neste tipo de operação a composição no fundo da coluna continuamente varia, de forma que a eficiência de separação deve também variar para que a composição de destilado se mantenha constante no topo da coluna. O perfil da razão de refluxo ao longo do tempo de destilação será variável. Quando o componente de interesse estiver em concentração significativa no topo, a razão de refluxo terá valores pequenos no início, que irão ser gradualmente aumentados a medida que o produto seja retirado.

Segundo Robinson (1970) a máxima recuperação de produto é obtida com valores de razão de refluxo intermediários entre a razão de refluxo constante e aqueles usados na operação à composição constante.

Shinskey (1988) apresenta um critério econômico para encerrar a batelada, comparar o lucro com a produção do destilado, com os custos operacionais.

Farhat, Pibouleau e Domenech (1991) estudaram o controle ótimo da destilação batelada via programação não linear, a fim de determinar um planejamento ótimo para o refluxo em função do período da destilação.

2.5 Controle e Automação da Unidade de Destilação em Batelada

Fileti (1992) desenvolveu um simulador dinâmico do processo e utilizou o mesmo para avaliar estratégias de controle na destilação em batelada, sob a política de operação de

composição constante no topo. Usou o controlador adaptativo programável, que apresentou desempenho superior às técnicas de controle convencional.

Cunha (1996) usou experimentalmente o controlador adaptativo programável para destilar uma mistura binária composta de n-hexano e n-heptano. Foram utilizadas as relações de equilíbrio líquido-vapor para inferenciar as composições de topo e fundo. O desempenho do controlador adaptativo programável foi bom, mas sensível às mudanças de condições operacionais da coluna, já que a estimação dos parâmetros do controlador era feita "off-line".

Fileti (1996) comparou em simulação os controladores adaptativo programável, o auto-ajustável, bem como o preditivo baseado em redes neurais artificiais, usando o modo de operação em composição constante no topo. Como leis de controle foram utilizadas o equivalente digital de um controlador PI, na forma posição e velocidade, e o algoritmo de Dahlin.

Oisovici (1998) testou experimentalmente três estratégias de controle que apresentaram bom desempenho: o controle adaptativo auto-ajustável, o controle por antecipação baseado no balanço de massa e o controle proporcional baseado na simulação da coluna. Os testes foram feitos com o sistema não ideal etanol e água e com o sistema ideal etanol e 1-propanol. Comparando os resultados obtidos com as três estratégias de controle, Oisovici (1998) determinou que a diferença entre os tempos de batelada não foi significativa. Porém, o perfil da razão de refluxo do controlador por antecipação foi mais suave do que os perfis obtidos com os controladores adaptativo e proporcional.

Oisovici (1998) embutiu no algoritmo de controle adaptativo auto-ajustável um conjunto de regras heurísticas, para determinar os critérios de aceitação das ações de controle calculadas pelo algoritmo dos mínimos quadrados recursivos (RLS), que se aproxima em muito à base de regras de Mamdani (1975), utilizada em controladores nebulosos.

Pedrosa (1998) desenvolveu e testou experimentalmente um algoritmo de controle auto-ajustável baseado em identificação recursiva de parâmetros do processo, o qual foi representado por um modelo discretizado de uma função de transferência de 1ª ordem com atraso de

transporte. Para realizar a estimação "on-line" dos parâmetros foi usado o método de programação quadrática sucessiva aplicada aos mínimos quadrados, da biblioteca matemática NAG. Como estratégias de controle foram testados o equivalente digital de um controlador PI, o algoritmo de Dahlin e o controlador adaptativo programável.

Pedrosa (1998) trabalhou com o sistema binário n-hexano e n-heptano, onde as composições de topo e fundo foram inferenciadas usando os dados de equilíbrio líquido-vapor e com o sistema ternário (n-hexano, ciclohexano e n-heptano) para o qual a composição no topo foi inferenciada via rede neural artificial treinada com os dados provenientes de simulações usando o modelo desenvolvido por Fileti (1992).

2.6 Inferência de Composição

A fim de efetuar o controle regulatório ou supervisório da composição no topo da coluna de destilação em batelada é necessário medir essa variável. Isso pode ser feito com o uso da análise cromatográfica "on-line", mas isso encarece muito a instrumentação para o processo, além de embutir um atraso de tempo no sistema, referente ao tempo de resposta necessário para se obter o valor da composição por cromatografia.

Uma alternativa mais adequada aos tempos de amostragem geralmente usados nas implementações usuais de controle "on-line" na destilação em batelada é medir as temperaturas de topo e fundo e inferenciar a composição usando dados termodinâmicos de equilíbrio líquido-vapor.

Esse foi o procedimento usado por Cunha (1996) e Pedrosa (1998), mas somente se adapta bem a misturas binárias ideais, pois para sistemas com múltiplos componentes ou não-ideais aumenta a complexidade e o esforço computacional do modelo termodinâmico de equilíbrio líquido-vapor que necessita ser utilizado.

Uma alternativa utilizada por Pedrosa (1998) foi treinar uma rede neural artificial que foi utilizada para determinar a concentração no topo da coluna, durante experimentos práticos. Para

tanto utilizou um simulador da coluna de destilação em batelada gerando dados de razão de refluxo interna, temperatura de topo e fundo e composição usados no treinamento da rede neural artificial.

Ponton e Klemes (1993) compararam a inferenciação de composição usando razões de polinômios, modelos fenomenológicos e redes neurais artificiais. Determinaram que as redes neurais além de passarem pela etapa exaustiva de treinamento ficam restritas às faixas comuns de operação, enquanto os modelos fenomenológicos são mais abrangentes.

2.7 Aplicações da Lógica Nebulosa em Engenharia Química

Usando o conhecimento especialista dos operadores de estações de tratamento de água, Yagishita, Itoh e Sugeno (1985) desenvolveram um sistema especialista nebuloso para a especificação das quantidades de produtos químicos (sulfato de alumínio, cal, cloro, etc) a serem utilizadas em uma estação de tratamento de água. O sistema baseia-se nas medidas de turbidez, alcalinidade, pH, etc, da água no seu estado bruto.

Peng e Liu (1988) construíram controladores auto-regulatórios de temperatura, em simulação e experimentalmente, baseando-se na idéia de Procyk e Mamdani (1979) dos chamados controladores lingüísticos auto-regulatórios, para controlar a temperatura de um forno elétrico.

Mahesh e Madhavan (1993) desenvolveram um controlador nebuloso para regular a composição em um reator semicontínuo de copolimerização em solução. Este controlador nebuloso utilizava somente uma variável de entrada (o erro na composição) e uma de saída (a ação de controle), por isso a ação do mesmo foi semelhante a de um controlador proporcional tradicional.

McCullough (1993) estudou o uso da lógica nebulosa no controle do processo de cura de resinas poliméricas, utilizando como variáveis medidas temperatura, tempo e fator de perda

capacitiva para determinar a pressão a ser utilizada no processo, a fim de se obter o polímero de características desejadas.

Karr (1993) desenvolveu um controlador nebuloso adaptativo aplicado ao controle de pH. Este é um problema difícil nos processos químicos, por causa da não-linearidade inerente, a medida que o pH é uma medida logarítmica, além das freqüentes mudanças na dinâmica do processo. A adaptação no controlador é feita através de algoritmos genéticos, que modificam as funções de pertinência do controlador nebuloso.

Ketonen (1993) estudou uma planta de ácido fosfórico e seu controle. Novas estratégias de controle e sistemas de monitoramento, das concentrações de ácido sulfúrico em dois reatores e de duas unidades de filtração, foram parcialmente desenvolvidos com o uso da lógica nebulosa.

Roffel (1993) argumentou que os reatores de polimerização são candidatos ideais para a aplicação do controle nebuloso, por que muitas de suas reações são difíceis de modelar, as análises químicas dos produtos são feitas a intervalos infreqüentes de tempo e traços de impureza podem ter efeito marcante na reação. Todos esses fatores manifestam-se modificando as propriedades dos polímeros, o que torna o controle da reação, usando abordagens convencionais, difícil. Ao longo dos anos, no entanto, operadores do processo desenvolveram a habilidade de controlar os reatores sob condições variáveis. Roffel (1993) utilizou esse conhecimento especialista no desenvolvimento de uma estratégia de controle nebulosa, abordando o problema de sintonia dos controladores nebulosos e algumas dificuldades de implementação.

Stenz e Kuhn (1993) desenvolveram e implementaram experimentalmente um controlador nebuloso aplicado a destilação em batelada multicomponente. Foram escolhidas como variáveis lingüísticas a variação de pressão no interior da coluna, a variação de temperatura no topo e a diferença de temperatura entre o topo e o prato número 8 (numa coluna de 10 pratos, com medida de temperatura no topo, no fundo e em todos os pratos pares). Como ferramenta para implementar o controlador nebuloso foi utilizada o produto Fuzzytech ®, que após a definição de todos os parâmetros do controlador nebuloso, cria um programa em linguagem C, o qual é usado experimentalmente. Para fins de comparação, estudaram também um controlador PID

tradicional. Com a configuração utilizada concluíram que não havia diferença entre o desempenho do controlador nebuloso e os controladores tradicionais. Cabe lembrar que esse resultado pode estar associado as escolhas dos parâmetros do controlador nebuloso: um número pequeno de funções de pertinência para definir as variáveis lingüísticas do processo, ou o tipo de funções de pertinência utilizadas (trapezoidais). .

Galichet (1994) usou um controlador nebuloso para o problema do controle do nível do tanque de alimentação da unidade de destilação atmosférica. Esse sistema em particular não necessita de um valor de referência fixo, mas oscilante entre o limite máximo e o limite mínimo. A restrição mais importante é levar em conta a necessidade de minimizar a faixa de ação na variável manipulada (vazão), a fim de estabilizar a vazão de saída do tanque. Além disso, o sistema deve antecipar propriamente as perturbações que ocorrem na alimentação da unidade de destilação (mudanças no óleo cru).

Li (1997) aplicou um controlador nebuloso a um reator nuclear, tanto em simulação quanto experimentalmente, a fim de melhorar as condições de segurança e custos operacionais em plantas nucleares de geração de energia elétrica.

Tanaka (1995) modelou a concentração de monóxido de carbono no ar, em uma interseção de tráfego de uma grande cidade do Japão, utilizando uma técnica "neuro-fuzzy". Esta técnica consistiu em um modelo nebuloso, acoplado a regra δ, um método básico de treinamento em redes neurais artificiais, que permitiu a identificação dos parâmetros do modelo nebuloso.

Wilson e Martinez (1997) utilizaram uma abordagem nova para a automação de processos em batelada, mais especificamente de uma coluna de destilação em batelada reativa, usando a modelagem nebulosa e o reforço de aprendizado. O núcleo central da estratégia de automação é um agente autônomo que continuamente "aprende" a implementar ações de controle que podem levar o estado do processo muito próximo ao desejado, com um desempenho quase ótimo. Um algoritmo para reforço por aprendizado chamado "fuzzy Q-learning" é proposto para construir o agente (controlador).

Biasizzo (1997) e outros propuseram uma nova abordagem para o controle preditivo de processos altamente não lineares baseada em um modelo nebuloso. O método do controle preditivo DMC usa modelos lineares do processo e então é incapaz de lidar com as fortes não-linearidades de alguns processos. Em sua abordagem Biasizzo (1997) acopla um modelo nebuloso não linear do processo ao algoritmo DMC. A fim de testar o desempenho do controlador foi estudado o problema do controle de pH.

Lu (1997) apresentou o desenvolvimento de um sistema especialista de controle nebuloso ótimo, aplicado em tempo real a uma unidade de craqueamento catalítico de uma refinaria de petróleo. Através da comparação com sistemas especialistas tradicionais, Lu (1997) determinou que o modelo relacional nebuloso desenvolvido foi capaz de realizar predição numérica da distribuição de produtos no craqueamento com grande robustez.

2.8 Proposta da Dissertação

A análise da literatura feita neste capítulo deixa claro que são poucos os trabalhos experimentais envolvendo controladores nebulosos em engenharia química, havendo somente uma aplicação no controle de colunas de destilação em batelada (Stenz e Kuhn , 1993) sem muito sucesso.

Um aspecto que sugere o uso dos controladores nebulosos na destilação em batelada é o fato de que esses controladores não necessitam da modelagem matemática do processo (Driankov, 1996) e dos dados físico-químicos do sistema a ser destilado.

Frente à importância da destilação em batelada na indústria química e diante das dificuldades operacionais que a mesma apresenta, o presente trabalho teve por objetivo explorar aspectos pouco relatados em literatura, através das seguintes etapas:

- desenvolver e testar experimentalmente um sistema de controle nebuloso aplicado à destilação batelada de misturas binárias;

- estudar as aplicações da lógica nebulosa em engenharia química, em especial no controle de processos químicos;

- determinar as influências dos parâmetros de projeto dos controladores nebulosos no desempenho dos mesmos, em simulação, bem como experimentalmente;

- confirmar as vantagens e as desvantagens do uso dos controladores nebulosos na destilação em batelada.

3. DESCRIÇÃO DA DESTILAÇÃO EM BATELADA:

Neste capítulo será descrita a unidade de destilação em batelada do Laboratório de Controle da Área de Concentração de Sistemas Químicos e Informática, utilizada no desenvolvimento dos experimentos realizados com os controladores nebulosos. Será descrita também, de maneira sucinta, a modelagem matemática vinculada ao processo, a qual foi usada na etapa de desenvolvimento do controlador nebuloso.

3.1 Descrição Física da Unidade Piloto

A aparelhagem experimental utilizada no transcorrer do trabalho consistiu de uma coluna de destilação de vidro e isolada termicamente (paredes tipo vaso de Dewar), composta de 12 pratos equipados com campânulas e perfurações, de diâmetro interno de 7.5 cm e altura de 3m (considerando o condensador).

O refervedor , de capacidade de 10 l, é munido de uma manta elétrica, composta de duas resistências elétricas, cada uma contando com uma potência de 750 W, podendo-se utilizar uma ou duas resistências no aquecimento.

O condensador utilizado foi um condensador total de 65 cm de altura e 7.5 cm de diâmetro interno, equipado com camisa e serpentina dupla.

O destilado obtido foi recolhido em frascos coletores, de 5 l, de onde era disposto para reutilização nas próximas bateladas experimentais.

Na seção intermediária entre a coluna propriamente dita e o condensador, encontrava-se o divisor de corrente, ao qual estava conectada a válvula magnética. Este dispositivo permite manipular o valor da razão de refluxo, de forma que quando acionada retorna o destilado para o interior da coluna e quando não acionada desvia o fluxo de destilado para os frascos coletores.

No anexo A encontram-se duas fotos da unidade de destilação em batelada, uma vista geral da montagem e um detalhe da válvula magnética.

Como a válvula magnética não permite valores de vazão intermediários, mas sim somente refluxo total ou não, é necessário adequar o tempo que a mesma passará em cada uma das duas posições possíveis (refluxo ou não), para implementar o valor da razão de refluxo proveniente do algoritmo de controle. Isso representa uma restrição a todo e qualquer algoritmo de controle que seja usado nesta unidade de destilação em batelada.

A razão de refluxo interna é descrita como:

$$r = \frac{L}{V} \qquad \text{(eq. 3.1)}$$

onde r é a razão de refluxo interna (adimensional, que varia entre 0 e 1), L é a vazão de líquido que retorna a coluna e V a vazão de vapor no topo.

Já a razão de refluxo externa é :

$$R = \frac{L}{D} \qquad \text{(eq. 3.2)}$$

onde R é a razão de refluxo externa (adimensional, que varia entre 0 e $+\infty$), L é a vazão de líquido que retorna a coluna e D a vazão de destilado.

A razão de refluxo externa se relaciona com a razão de refluxo interna por:

$$r = \frac{R}{R+1} \qquad \text{(eq. 3.3)}$$

A razão de refluxo é implementada, na prática, fazendo-se a seguinte aproximação: o valor numérico da razão de refluxo interna (r) equivale a fração do intervalo de amostragem que a válvula passará acionada na posição refluxo. O restante do período de amostragem a válvula permanece na posição não acionada (posição destilado).

Chamando de t_L o intervalo de tempo em que a válvula permanece na posição refluxo, de t_D o intervalo de tempo em que a válvula passa na posição destilado, as relações entre as razões de refluxo internas e externas e esses tempos são:

$$r = \frac{t_L}{t_R} \quad \text{(eq. 3.4)}$$

$$R = \frac{t_L}{t_D} \quad \text{(eq. 3.5)}$$

A relação entre t_L , t_D e o intervalo de amostragem (t_A) é:

$$t_A = t_D + t_L \quad \text{(eq. 3.6)}$$

Cabe lembrar que, em se tratando de controle "on-line", o projetista do sistema de controle tem a forte restrição de trabalhar com o tempo real, e como o algoritmo de controle (qualquer que seja o mesmo) utilizará um certo tempo para realizar todas os passos necessários para determinar qual o valor da razão de refluxo a ser usado, este tempo deve ser levado em consideração.

O projetista tem duas opções: ou determina que durante esse tempo a válvula permanecerá "fechada" (posição refluxo) ou "aberta" (posição destilado). No presente trabalho, como no trabalho de Pedrosa (1998), optou-se por manter a válvula fechada durante esse intervalo de tempo. Assim o período de tempo durante o qual a válvula permanecerá na posição refluxo (calculado pela razão de refluxo interna) deve ter descontado o tempo gasto no algoritmo de controle (nas etapas de aquisição de dados, processamento, etc).

Por exemplo, para um tempo de amostragem de 10 s (utilizado durante todo o transcorrer deste trabalho, tanto em simulação quanto experimentalmente), uma razão de refluxo interna de 0.5 (correspondente a um valor de R igual a 1.0) calculada por um algoritmo de controle que gastou 1.8 s para ser executado, o tempo que a válvula permanecerá em refluxo será de 5 s (50% de 10). Enquanto processa o algoritmo, o microcomputador mantém a válvula fechada, logo de

antemão a válvula permanecerá fechada por 1.8s por isso o tempo que ela permanecerá ainda fechada será de 5.0s – 1.8s = 3.2 s .

3.2 Instrumentação

Para efetuar as medidas de temperatura, foram utilizados dois termopares do tipo J, um no topo da coluna e outro no refervedor, com uma precisão de +/- 1,0 °C, calibrados na faixa de 4-20 mA/0-150°C, de acordo com a seguinte equação:

$$T = 150 . (V_T - 1050) / (4220 - 1050) \quad \text{(eq. 3.7)}$$

Onde V_T é o valor do sinal de voltagem oriundo do termopar e T a temperatura em graus Celsius.

Esses termopares geram sinais analógicos contínuos de corrente, que devem ser transformados para a forma digital (números binários), para poderem ser manipulados pelo microcomputador. Para isso há a necessidade de se dispor de uma placa conversora analógica-digital.

Essa placa não só faz a conversão no sentido analógico-digital, mas também no sentido inverso, por isso é mais corretamente chamada de placa analógica-digital-analógica (ADA). Ela é composta de um conversor analógico-digital, responsável pela conversão do sinal de tensão oriundo do processo em sinal digital numérico que pode ser utilizado pelo microcomputador; um conversor digital-digital, que converte sinais digitais numéricos em sinais digitais de voltagem (0 a 5 V) pelos quais é acionada a válvula magnética e seleciona-se o canal de temperatura a ser lida; e por um conversor digital-analógico cuja finalidade é a reconstrução do sinal.

A montagem conta também com um painel de controle analógico para acompanhamento durante a destilação, e também com um dispositivo de segurança que desliga o aquecimento em caso de elevação excessiva de temperatura no refervedor da coluna.

3.3 – Descrição do Sistema de Controle Digital

O objetivo principal das ações do controlador desenvolvido é controlar a composição molar no topo da coluna de destilação em batelada, manipulando a razão do refluxo de destilado para o interior da coluna.

Para isso, é necessário que o programa controlador execute uma seqüência de tarefas, descritas a seguir:

a) selecionar o canal analógico a ser lido, a fim de ler os valores de voltagem provenientes do termopar selecionado (termopar de topo da coluna), sendo que isso é feito enviando-se um sinal, através do conversor digital-digital da placa ADA.

b) amostrar os valores de voltagem analógicos, que são convertidos para a forma digital pelo conversor analógico-digital , em uma quantidade bastante grande (por exemplo 10.000 ou 30.000 leituras) , a fim de que esse sinal possa passar por um filtro digital de ruído (no caso das 10.000 leituras por um filtro exponencial duplo, no caso das 30.000 leituras por um filtro de média) ;

c) repetir as etapas "a" e "b" para o outro termopar (de fundo).

d) transformar os valores médios de voltagens obtidos em valores de temperatura de acordo com a equações de calibração dos termopares;

e) inferenciar a composição no topo;

f) utilizar os valores de composição no algoritmo do controlador nebuloso, para determinar qual a ação que deve ser tomada;

f) enviar um sinal, durante o tempo necessário para implementar a razão de refluxo, pelo conversor digital-digital da placa ADA para que a válvula fique na posição acionada.

Na figura 3.1 encontra-se uma representação do sistema de aquisição de dados e atuação do controle digital implementado na coluna de destilação em batelada.

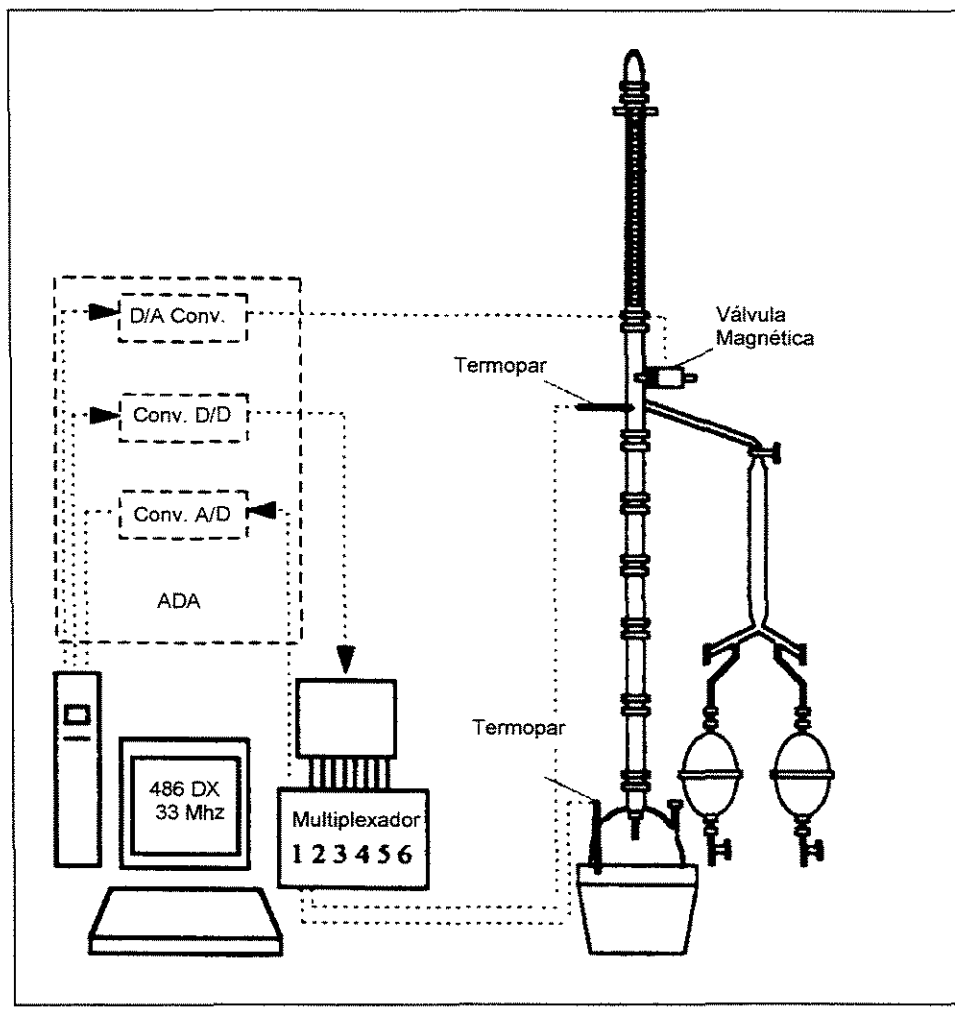


Figura 3.1: Esquema da Coluna de Destilação em Batelada

3.4 – Modelagem Matemática da Destilação em Batelada.

Para desenvolver o controlador nebuloso, foi necessário utilizar o simulador da destilação batelada, desenvolvido por Fileti (1992); a modelagem utilizada no mesmo será descrita a seguir.

As especificações de operação e geometria da coluna de destilação, juntamente com as restrições durante a etapa da modelagem matemática, influenciam a simulação do sistema.

No simulador desenvolvido por Fileti (1992), foram feitas as seguintes suposições:

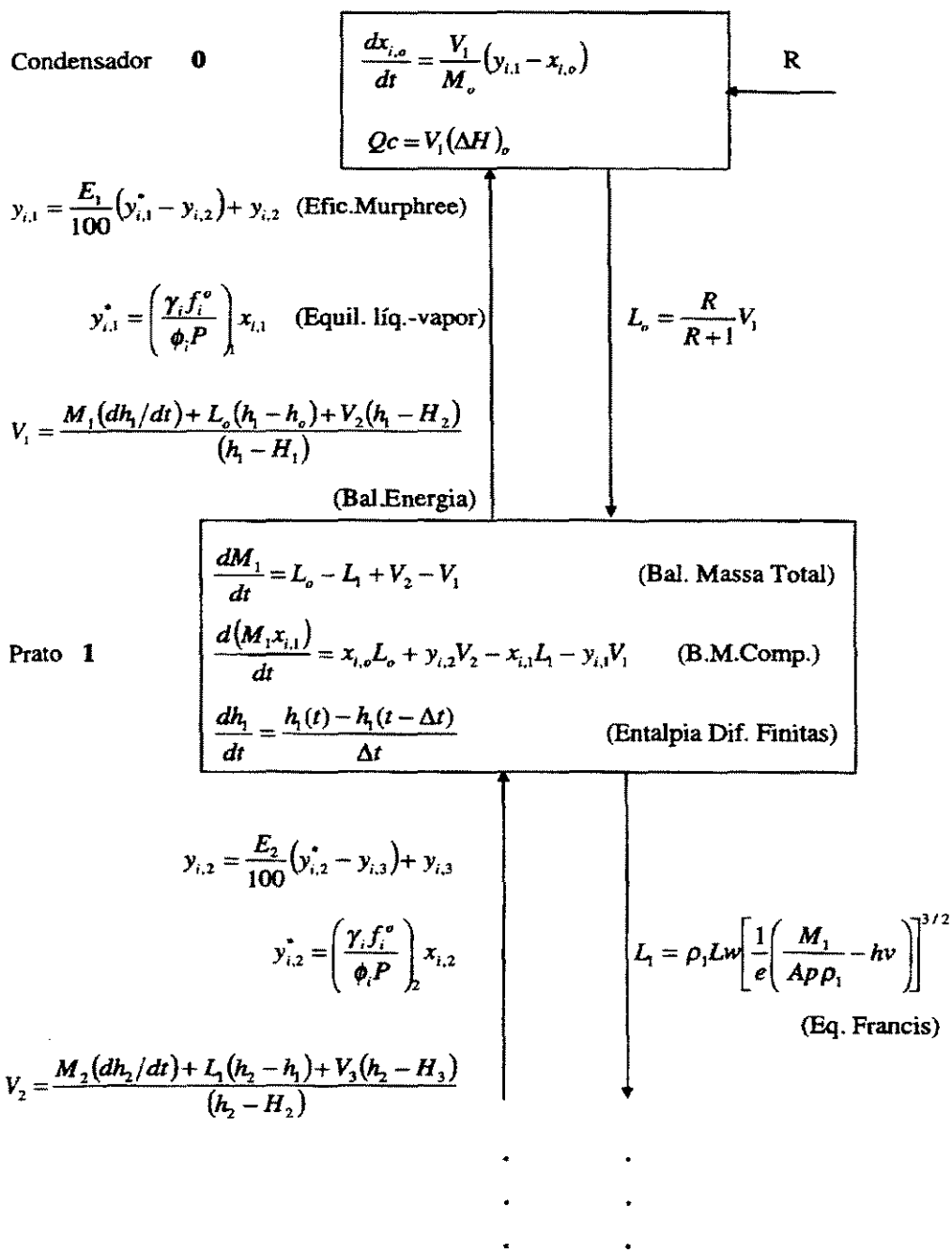
- acúmulo de vapor no prato desprezível;
- perda de carga desprezível ao longo da coluna;
- eficiência de separação constante nos pratos;
- acúmulo pequeno de líquido no condensador;
- condensador total;
- o refervedor é considerado estágio ideal;

A modelagem matemática foi desenvolvida com base nas equações de balanços diferenciais de massa e energia para cada um dos estágios de equilíbrio da coluna. Cabe salientar que para facilitar a resolução do problema, as taxas de variação de entalpia em cada prato foram aproximadas às equações de diferenças finitas.

Para calcular as fluxos molares de líquido no interior da coluna foi utilizada a equação de Francis, também usada para calcular o acúmulo de líquido em cada prato.

Como se percebe, através da figura 3.2 abaixo , a modelagem da coluna é composta principalmente de equações diferenciais ordinárias, que necessitam de um método numérico para sua solução. Fileti (1992) utilizou o método de Runge-Kutta de 4ª ordem, usando um passo de integração de 0,07 minutos.

A comparação dos resultados obtidos por este simulador e dados de literatura provou coerência entre os mesmos apesar das hipóteses simplificadoras consideradas.


Condensador 0
$\frac{dx_{i,o}}{dt} = \frac{V_1}{M_o}(y_{i,1} - x_{i,o})$
$Qc = V_1(\Delta H)_o$
R
$y_{i,1} = \frac{E_1}{100}(y^*_{i,1} - y_{i,2}) + y_{i,2}$ (Efic.Murphree)
$y^*_{i,1} = \left(\frac{\gamma_i f_i^o}{\phi_i P}\right)_1 x_{i,1}$ (Equil. líq.-vapor)
$L_o = \frac{R}{R+1} V_1$
$V_1 = \frac{M_1(dh_1/dt) + L_o(h_1 - h_o) + V_2(h_1 - H_2)}{(h_1 - H_1)}$
(Bal.Energia)
Prato 1
$\frac{dM_1}{dt} = L_o - L_1 + V_2 - V_1$ (Bal. Massa Total)
$\frac{d(M_1 x_{i,1})}{dt} = x_{i,o}L_o + y_{i,2}V_2 - x_{i,1}L_1 - y_{i,1}V_1$ (B.M.Comp.)
$\frac{dh_1}{dt} = \frac{h_1(t) - h_1(t - \Delta t)}{\Delta t}$ (Entalpia Dif. Finitas)
$y_{i,2} = \frac{E_2}{100}(y^*_{i,2} - y_{i,3}) + y_{i,3}$
$y^*_{i,2} = \left(\frac{\gamma_i f_i^o}{\phi_i P}\right)_2 x_{i,2}$
$L_1 = \rho_1 Lw\left[\frac{1}{e}\left(\frac{M_1}{Ap\rho_1} - hv\right)\right]^{3/2}$
(Eq. Francis)
$V_2 = \frac{M_2(dh_2/dt) + L_1(h_2 - h_1) + V_3(h_2 - H_3)}{(h_2 - H_2)}$

(Continua...)

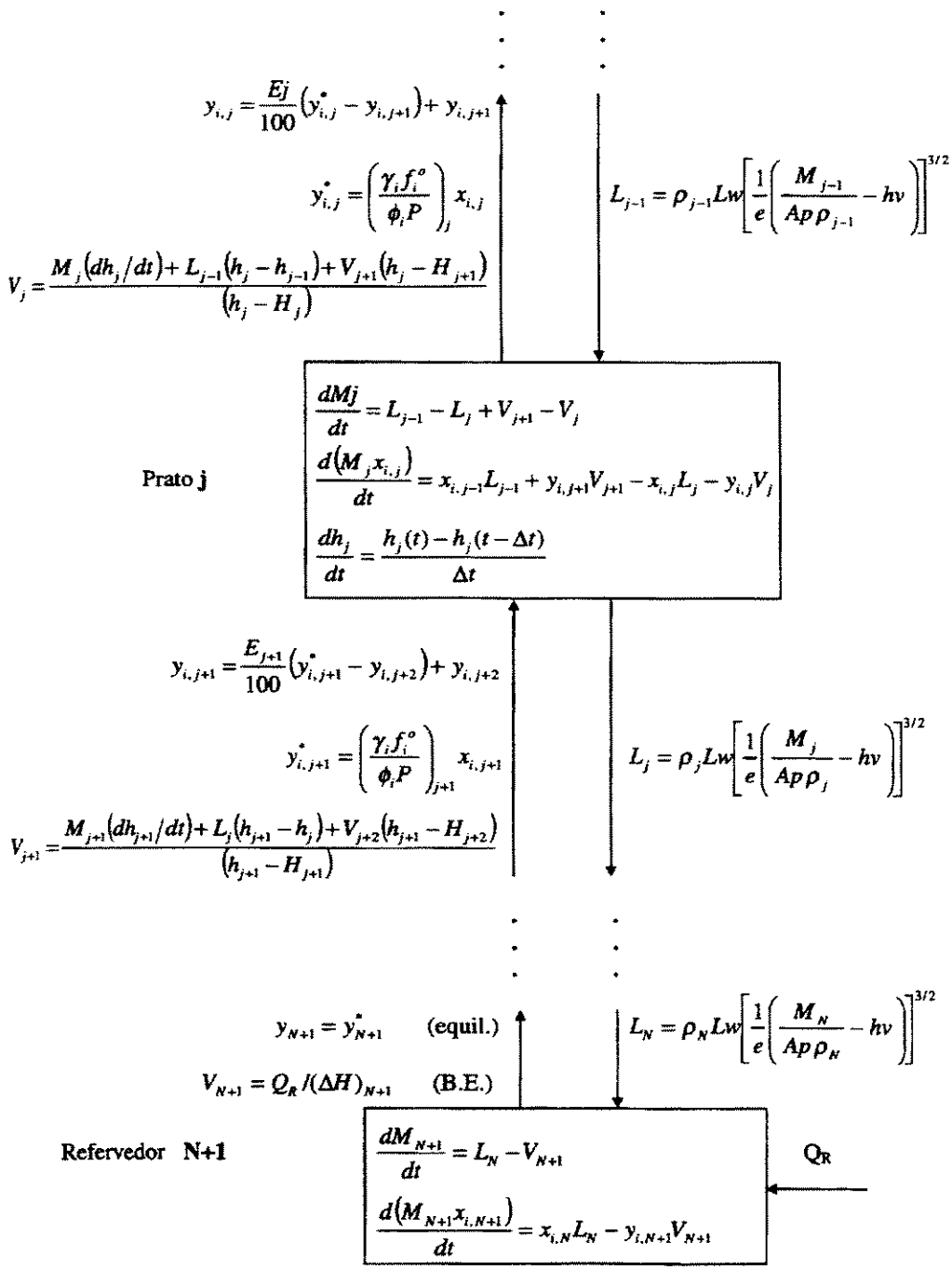


Figura 3.2 – Modelagem Matemática da Destilação Multicomponente em Batelada

4. CONCEITOS BÁSICOS DOS CONTROLADORES NEBULOSOS

4.1 Introdução

A lógica nebulosa foi um dos maiores desenvolvimentos decorrentes da teoria dos conjuntos nebulosos e foi primariamente projetada para representar e lidar com o conhecimento que não pode ser expresso por medidas quantitativas. A principal idéia dos algoritmos baseados em lógica nebulosa é imitar o raciocínio humano para controlar processos de difícil modelagem fenomenológica ou de grande transitoriedade.

Os sistemas de inferência nebulosa modelam os aspectos qualitativos do conhecimento humano através de regras lingüísticas "se-então" (Zadeh, 1965). Eles capturam a imprecisão do processo de raciocínio sem usar análise quantitativa. Os pesquisadores deste ramo não pretendem que a abordagem da lógica nebulosa seja uma metodologia emprestada dos sistemas biológicos ou tendo fundamentações biológicas. É um ramo rigoroso da matemática oferecendo soluções para a tecnologia de controle.

Os algoritmos clássicos de controle lidam com sistemas que tem um modelo matemático (fenomenológico ou empírico) precisamente definido. No projeto de um controlador nebuloso não se necessita conhecer o modelo matemático do processo a ser controlado. Se o modelo existe, no entanto, pode ser usado para a simulação e para o teste da estratégia de controle. As principais vantagens dos controladores nebulosos , segundo Godjevac (1995) são :

- não necessitam de um modelo matemático do processo;
- permitem implementar o conhecimento e a experiência de um especialista humano usando regras lingüísticas compreensíveis;
- possibilitam controlar processos não-lineares.

Um dos resultados do grande crescimento no uso de sistemas de controle nebulosos nos anos 90 foi a opinião geral que a abordagem nebulosa poderia resolver todos os problemas de

controle e que as abordagens tradicionais de controle iriam ser logo abandonadas. No entanto não é possível solucionar todos os problemas de controle usando lógica nebulosa por que existem algumas desvantagens inerentes ao seu emprego:

- não há um método padronizado para transformar o conhecimento humano ou a experiência de um operador humano em um banco de regras para uso em um sistema de inferência nebulosa, e não há um procedimento geral para escolha de um número ótimo de regras, pois um grande número de fatores está envolvido na decisão (por exemplo, desempenho do controlador, eficiência de computação, comportamento do operador humano, a escolha das variáveis lingüísticas, etc).

- quando a operação do sistema em estudo é feita por um operador humano, o conhecimento deste é freqüentemente incompleto e episódico, raramente sistemático;

- existe a necessidade de se encontrar um bom método de ajuste das funções de pertinência a fim de minimizar o erro na variável medida e/ou otimizar o desempenho do sistema em análise;

- não é possível testar a estabilidade do sistema controlado, pois o modelo não é conhecido;

- o tempo computacional pode ser longo, devido as operações de fuzificação e particularmente de defuzificação;

Em conseqüência destes problemas, muito pesquisadores tem tentado automatizar o processo de construir sistemas nebulosos, por exemplo através da fusão entre redes neurais e sistemas nebulosos, ou entre algoritmos genéticos e sistemas nebulosos .

4.2 O Controle Nebuloso e o Controlador PID Tradicional

Nos dias atuais, uma parte bastante representativa dos controladores são implementados por algoritmos computacionais. Isso implica que as variáveis de entrada dos controladores são medidas através de taxas de amostragem.

O controlador clássico proporcional integral derivativo (PID) pode ser representado na forma velocidade:

$$\Delta p_k = K_c.\left[(e_k - e_{k-1}) + \frac{\Delta t}{\tau_I}.e_k + \frac{\tau_D}{\Delta t}.(e_k - 2e_{k-1} + e_{k-2})\right] \quad \text{(eq. 4.1)}$$

onde Δp_k é a variação na saída do controlador no instante k atual , K_c o ganho proporcional, e_k o erro na variável controlada no instante k, e_{k-1} o erro no instante k-1, Δt o tempo de amostragem, τ_I a constante de tempo integral e τ_D a constante de tempo derivativa.

Observando o termo que multiplica a parcela derivativa da expressão acima, pode-se concluir que a mesmo representa a diferença entre a variação do erro no instante k e a variação do erro no instante k-1. A essa diferença, podemos chamar de segunda variação do erro:

$$\Delta e_k = e_k - e_{k-1} \quad \text{(eq. 4.2)}$$

$$\Delta e_{k-1} = e_{k-1} - e_{k-2} \quad \text{(eq. 4.3)}$$

$$\Delta^2 e_k = \Delta e_k - \Delta e_{k-1} = e_k - e_{k-1} - (e_{k-1} - e_{k-2}) = e_k - 2e_{k-1} + e_{k-2} \quad \text{(eq. 4.4)}$$

Desta forma, inserindo as expressões da variação do erro e da segunda variação do erro na equação do controlador PID na forma velocidade, agrupando as constantes, obtém-se a equações de diferenças:

$$\Delta p_k = k_c.\Delta e_k + k_I.e_k + k_D.\Delta^2 e_k \quad \text{(eq. 4.5)}$$

onde:

$$k_c = K_c \quad \text{(eq. 4.6)}$$

$$k_I = \frac{K_c.\Delta t}{\tau_I} \quad \text{(eq. 4.7)}$$

$$k_d = \frac{K_c.\tau_D}{\Delta t} \quad \text{(eq. 4.8)}$$

Como se pode ver, a ação do controlador PID na forma velocidade é uma função linear do erro, da variação do erro e da segunda variação do erro.

Já os controladores nebulosos são não lineares, conforme demonstrado por Jager (1995), e dependendo das variáveis consideradas na premissa das regras nebulosas, podem ser análogos aos controladores P, PI, PD e PID convencionais.

Tipo de Regra Nebulosa	P	I	D
Se o erro é __ então a ação de controle é __	X		
Se o erro é __ então a variação na ação de controle é __		X	
Se o erro é __ e a variação do erro é __ então a ação de controle é __	X		X
Se o erro é __ e a variação do erro é __ então a variação na ação de controle é __	X	X	
Se o erro é __ e a variação do erro é__ e a segunda variação do erro é __ então a variação na ação de controle é __	X	X	X

Tabela 4.1: Tipo de Regra e Correspondente Tipo de Controlador Nebuloso.

Neste trabalho, os controladores nebulosos serão identificados, de acordo com a tabela 4.1, como, por exemplo controlador PID nebuloso, que é aquele controlador nebuloso que na premissa das regras estão o erro, a variação do erro e a segunda variação do erro.

4.3 Fundamentos da Lógica Nebulosa

O principal conceito da teoria nebulosa é a noção de conjunto nebuloso. Zadeh (1965) deu a seguinte definição:

"Um conjunto nebuloso é uma classe de objetos com graus contínuos de pertinência. Tal conjunto é então caracterizado por uma função característica de pertinência que vincula cada objeto a um grau de pertinência variando entre zero e um".

Na teoria clássica de conjuntos há uma clara distinção entre os elementos que pertencem e os que não pertencem a um conjunto. A lógica nebulosa admite a possibilidade de uma pertinência parcial (o chamado grau de pertinência).

Seja U uma coleção de objetos denotados genericamente por {u}, onde U é chamado de universo e "u" representa um elemento genérico de U. Um subconjunto nebuloso F num universo U é caracterizado por uma função de pertinência μ_F que assume valores no intervalo [0,1], isto é:

$$\mu_F : U \rightarrow [0,1] \quad \text{(eq. 4.9)}$$

Um subconjunto nebuloso F em U pode ser representado como um conjunto de pares ordenados de um elemento genérico u e seu grau de pertinência μ_F na função:

$$F = \{ (u, \mu_F(u) / u \in U\} \quad \text{(eq. 4.10)}$$

Um conjunto nebuloso discreto é, em geral, expresso através dos elementos que compõem seu universo e do grau de pertinência de cada um desses elementos. Por exemplo:

A = { 0,00/(0); 0,01/(1); 0,14/(2); 0,60/(3); 1,00/(4); 0,60/(5); 0,14/(6); 0,01/(7); 0,00/(8)}

os números do lado direito das barras são os elementos do universo {0,1,2,3,4,5,6,7,8}, enquanto os números da esquerda correspondem ao grau de pertinência de cada elemento do universo. Na figura 4.1 abaixo (em azul) estão representados os elementos do conjunto discreto "A".

Um conjunto nebuloso contínuo é expresso por uma função matemática que relaciona os elementos do universo com seus graus de pertinência. Um conjunto nebuloso contínuo pode, por exemplo, ser representado pela equação (representada na figura 4.1):

$$\mu_{(u_i)} = e^{\frac{-(u_i-4)^2}{2}} \qquad \text{(eq. 4.11)}$$

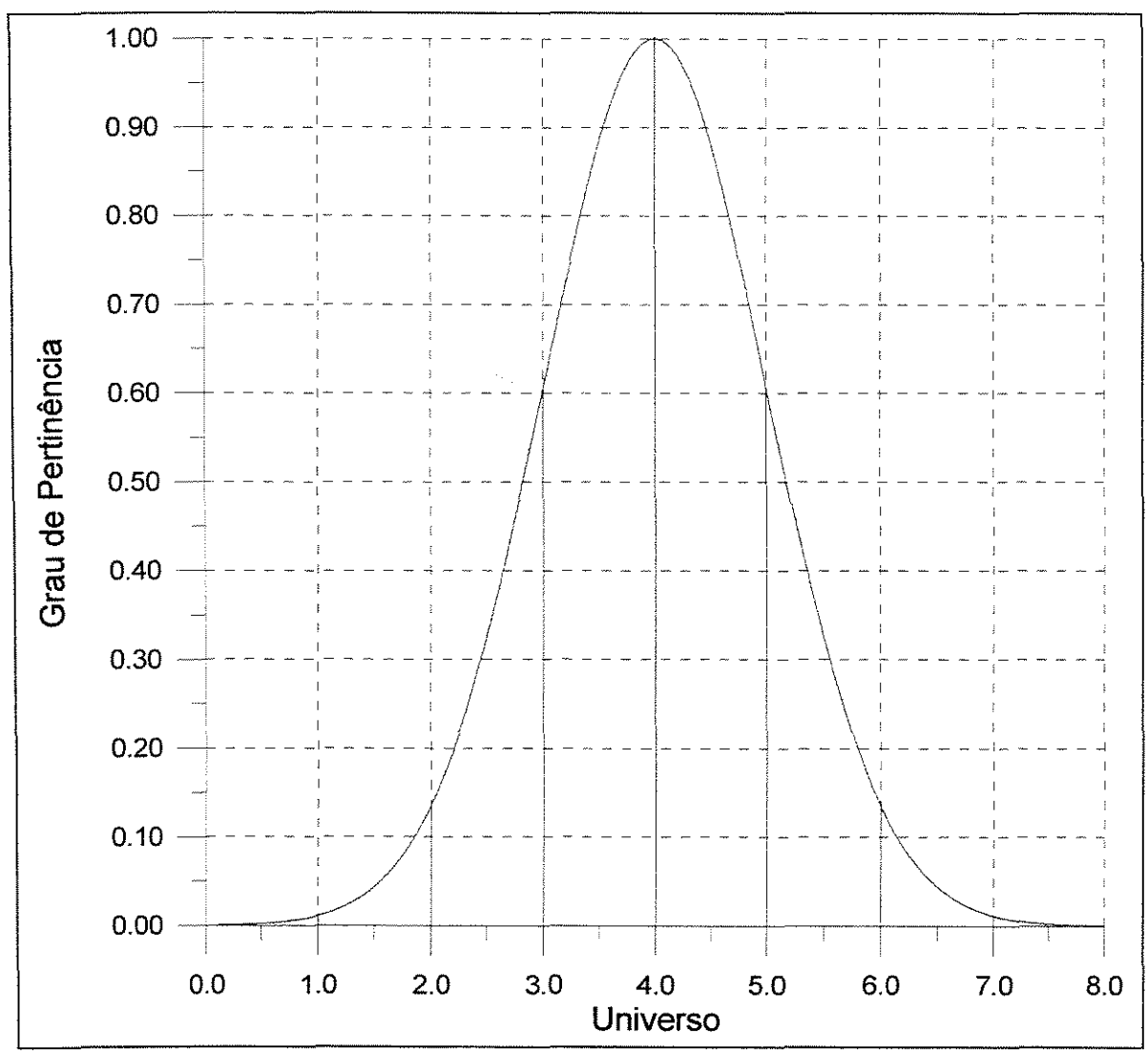


Figura 4.1: Conjuntos Nebulosos Discretos e Contínuos.

onde os elementos do universo são representados pela variável u_i. Os graus de pertinência são denotados por $\mu(u_i)$.

4.4 Principais Operações com Conjuntos Nebulosos

Sejam A e B dois conjuntos nebulosos em U com funções de pertinência μ_A e μ_B, respectivamente. As operações de união, interseção e complemento para conjuntos nebulosos são definidas a seguir:

a) união: resulta num conjunto de elementos que pertencem a qualquer um dos conjuntos envolvidos na operação; equivale a operação lógica "ou". A função de pertinência $\mu_{A\cup B}$ resultante da operação $A \cup B$ é definida para todos $u \in U$ por:

$$\mu_{A\cup B}(u) = \text{máximo}\{\mu_A(u), \mu_B(u)\} \quad \text{(eq. 4.12)}$$

b) interseção: na teoria clássica de conjuntos, a interseção de dois conjuntos contém os elementos que estão em ambos os conjuntos. Mas, por definição, um elemento de um subconjunto nebuloso pode estar parcialmente em outro. A interseção, em lógica nebulosa, eqüivale ao operador lógico "e". A função de pertinência $\mu_{A\cap B}$ da interseção $A\cap B$ é definida, segundo Zadeh (1965), para todos $u \in U$ por:

$$\mu_{A\cap B}(u) = \text{mínimo}\{\mu_A(u), \mu_B(u)\} \quad \text{(eq. 4.13)}$$

c) complemento: em geral, o complemento de um conjunto contém todos os elementos que não estão no conjunto. Já para um conjunto nebuloso o seu complemento é usualmente definido como um conjunto que contém elementos onde o grau de pertinência é igual a 1 menos o grau de pertinência do conjunto original. Isto corresponde a operação lógica "não".

$$\mu_{A^-}(u) = 1-\mu_A(u); \; u \in U \quad \text{(eq. 4.14)}$$

4.5 Variáveis Lingüísticas e Cardinalidade

Em engenharia é muito comum a manipulação de variáveis (temperatura, pressão, concentração, razão de refluxo, etc) que são escalares bem definidos ("crisp"). Já no dia a dia muitos conceitos são mal definidos, por exemplo a temperatura do dia pode estar baixa, média, ou alta. Variáveis que assumem valores lingüísticos mal definidos, chamam-se variáveis lingüísticas.

Pode-se usar conjuntos nebulosos para representar as variáveis lingüísticas, as quais podem ser usadas para descrever o estado de um processo ou as variáveis de controle.

Por exemplo: se a temperatura T, de um sistema qualquer, é interpretada como uma variável lingüística, então o conjunto nebuloso que a representa poderia ser:

$$T(t) = (\text{baixa, média, alta})$$

onde cada termo de T(t) é caracterizado no universo de discurso U = (298; 398). Interprete-se "baixa" como uma temperatura inferior a 348 K, "média" como uma temperatura próxima a 373 K e "alta" como uma temperatura acima de 398 K. Esses termos podem ser caracterizados como conjuntos nebulosos, enquanto as funções de pertinência são mostradas abaixo:

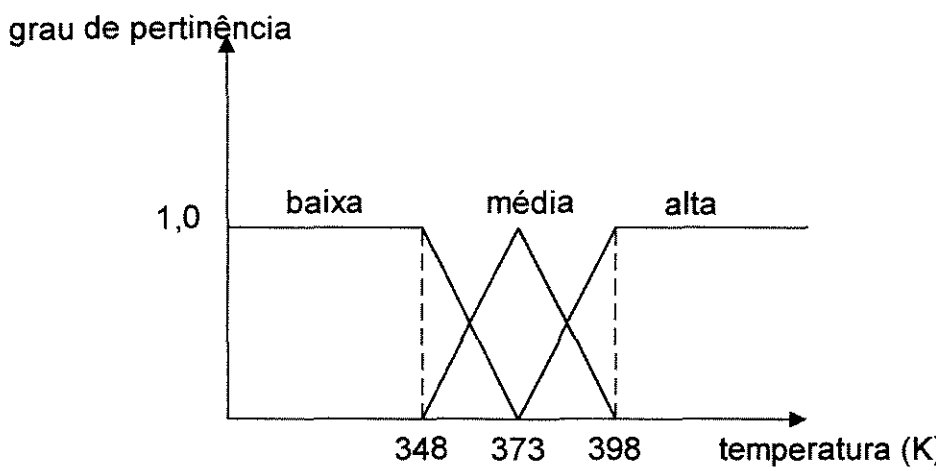


Figura 4.2: Exemplo de Representação Nebulosa da Variável Lingüística Temperatura.

Na figura 4.2, nota-se que a variável lingüística temperatura está representada utilizando-se três adjetivos possíveis: "baixa", "média" e "alta". A esse número de adjetivos que constituem o conjunto nebuloso, dá-se o nome de cardinalidade. Cabe salientar também que a ordenada correspondente ao ponto de cruzamento entre as duas funções de pertinência consecutivas é chamado grau de cruzamento ("crossover").

Como é fácil deduzir, as definições de alta, média e baixa em termos das variáveis lingüísticas dependerão do projetista do sistema nebuloso. Essa é uma das maiores vantagens dos controladores nebulosos, a flexibilidade, que permite ajustar os conjuntos nebulosos de acordo com a vontade do projetista do controlador.

4.6 Regras Lingüísticas

Como já mencionado, a idéia principal dos sistemas de lógica nebulosa é expressar o conhecimento humano na forma de regras lingüísticas "se-então". Cada regra tem duas partes:

- a premissa, expressa pelo "se...";

- a conseqüência, expressa pelo "então...";

A premissa é a descrição do estado do sistema, e a conseqüência é a ação que o operador que controla o sistema deve tomar. Existem muitas formas de regras "se-então". A mais geral é:

SE (um conjunto de condições é satisfeito) ENTÃO (uma série de conseqüências pode ser inferida)

Zadeh (1973) foi o primeiro a introduzir a noção de regra nebulosa, da forma geral:

SE X é A ENTÃO Y é B

onde X é uma variável de entrada do controlador, Y é a ação de controle, A e B são adjetivos nebulosos que indicam determinada função de pertinência em questão.

Takagi e Sugeno (1983) propuseram uma forma de regra que somente tem conjuntos nebulosos na sua premissa, enquanto que a conseqüência é descrita através de uma equação da variável de entrada, por exemplo:

SE velocidade é alta ENTÃO força é constante*(velocidade)^2

A generalização da regra de controle Rc a qual tem duas condições na parte antecedente tem a seguinte forma:

SE X é A e Y é B ENTÃO Z é C

ou também: $Rc = (A \text{ e } B) \rightarrow C$ (eq. 4.15)

A regra de controle é implementada pela implicação nebulosa (relação nebulosa) e é definida como se segue:

$$\mu_{Rc} = \mu_{A \text{ e } B \rightarrow C}(X,Y,Z) = [\mu_A(X) \underline{e} \mu_B(Y)] \rightarrow \mu_C(Z) \quad \text{(eq. 4.16)}$$

A,B e C são conjuntos nebulosos definidos no universo de discurso para X, Y e Z. O grau de pertinência μ_{Rc} é chamado de nível de fidelidade, ou grau de ativação para a regra Rc enquanto μ_A e μ_B são os níveis de pertinência, respectivamente, da variável x no conjunto nebuloso A e da variável y no conjunto nebuloso B.

Essa operação nebulosa de implicação é definida na bibliografia de diversas formas. Por esse motivo, desde que Zadeh (1973), introduziu o conceito de "raciocínio aproximado", um grande número de pesquisadores propôs várias funções de implicação e operadores para o "e" nebuloso. Aproximadamente 40 funções de implicação nebulosas diferentes estão descritas na

literatura (Jager, 1995). Para o operador lógico "e", as operações mais freqüentemente usadas são os operadores produto e mínimo.

4.7 Estrutura de um Controlador Nebuloso

Segundo Gomide (1992) a estrutura de um controlador nebuloso é a seguinte:

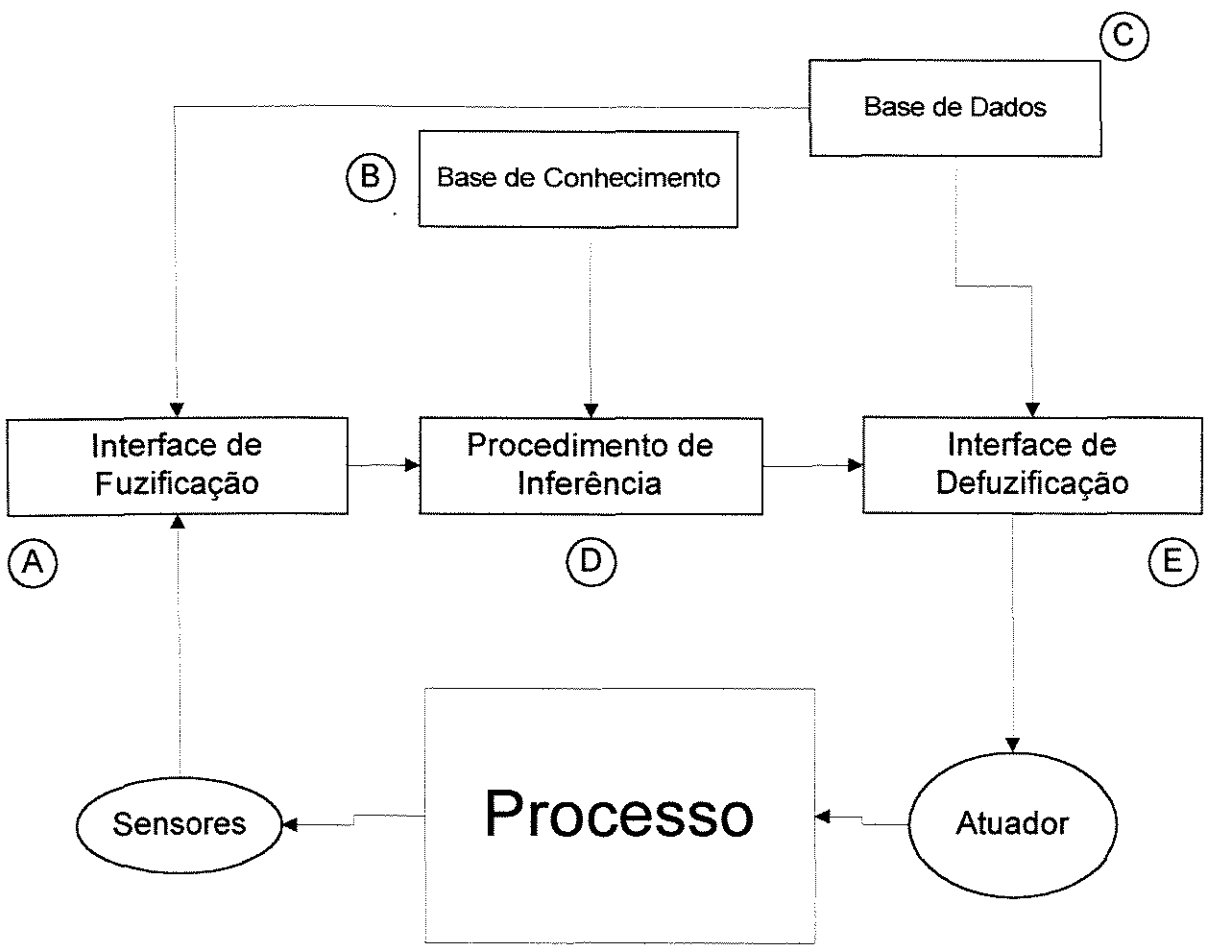


Figura 4.3: Estrutura Genérica de um Controlador Nebuloso ("Fuzzy").

As principais partes do controlador nebuloso descrito na figura 4.3 acima são:

a) Interface de fuzificação, toma os valores da variáveis medidas pelos sensores, e transforma esses valores usando as funções de pertinência, em uma representação usando conjuntos nebulosos ;

b) Base de conhecimento, consiste da base de regras, caracterizando a estratégia de controle e suas metas;

c) Base de dados, armazena as definições das partições nebulosas dos espaços de entrada e saída (parâmetros das funções de pertinência), bem como os fatores de escala a serem usados no controlador;

d) Procedimento de inferência nebulosa, processa os dados nebulosos de entrada, junto com as regras de modo a inferir as ações de controle nebuloso, aplicando o operador de implicação nebulosa e as regras de inferência da lógica nebulosa;

e) Interface de defuzificação, transforma as ações de controle nebulosas e efetua uma compatibilização entre os valores vindos do estágio anterior com os valores dos universos de discurso reais das variáveis.

4.8 Estrutura Geral do Processamento Nebuloso

Os blocos A,D e E da figura 4.3, constituem a parte que realmente faz uso da lógica nebulosa na estrutura de um controlador nebuloso.

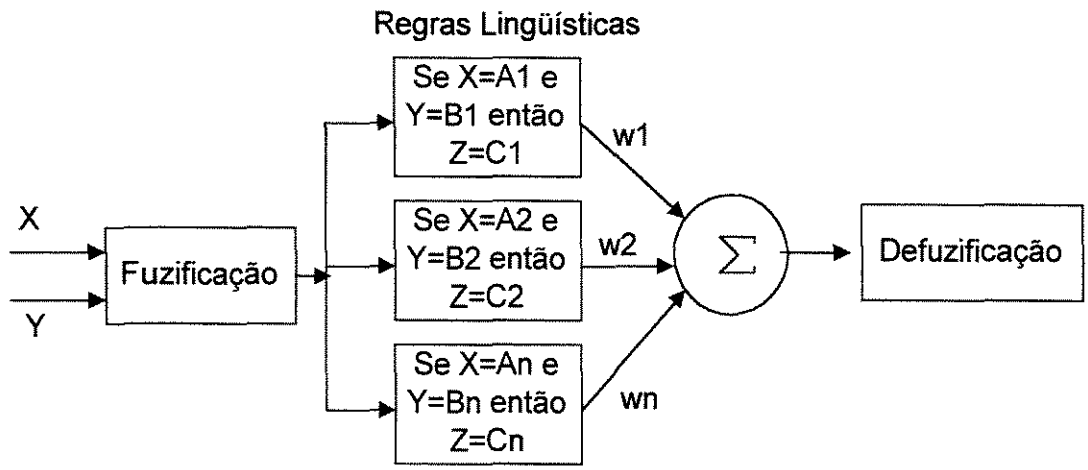


Figura 4.4: Estrutura Geral da Parte Nebulosa.

O raciocínio nebuloso pode ser dividido nas seguintes etapas (figura 4.4):

- Fuzificação, a cada valor de variável de entrada é atribuído um grau de pertinência, para todos os conjuntos nebulosos definidos no universo de discurso;

- Regras lingüísticas, aplica-se a operação de inferência (usualmente o operador mínimo) sobre os valores de pertinência na premissa;

- Geração do valor conseqüente para cada regra, o qual pode ser exato ou nebuloso;

- Defuzificação: gera os valores de saída exatos, a partir da saída nebulosa da inferência.

4.9 Tipos de Raciocínio Aproximado Nebuloso e Métodos de Defuzificação

Existem muitos tipos de raciocínio nebuloso (Jager, 1995). O mais importante citado na literatura é aquele onde a operação de mínimo é aplicada para obter as saídas nebulosas de cada regra, e a saída exata é obtida através do método do centro de área. Existem outros métodos de

defuzificação nesse caso, tais como: critério do máximo, média dos máximos, bisector da área, etc (Jager,1995);

4.9.1 O Método do Centro de Gravidade (Centro de Área)

Amplamente utilizado, calcula o centro de gravidade (ou centro de área) da figura formada pelas áreas abaixo das funções de pertinência das ações, que fazem parte das conseqüências das regras "se-então" ativadas.

$$z = \frac{\sum_{i=1}^{n} r_i . z_i}{\sum_{i=1}^{n} z_i} \qquad \text{(eq. 4.17)}$$

onde n é o número funções de pertinência de ação ativadas, r_i é o grau de ativação de cada regra, e z_i representa o centro da função de pertinência "i".

4.9.2 O Método do Máximo

O método do máximo determina a saída não nebulosa, z, a partir do função de pertinência para a qual o grau de ativação for maior.

4.10 Exemplo de Raciocínio Aproximado Nebuloso

Suponha um sistema nebuloso de duas entradas e uma saída. A combinação de 2 regras:

Rc1: Se X é A_1 e Y é B_1 então z é C_1

Rc2: Se X é A_2 e Y é B_2 então z é C_2

será feita no procedimento de defuzificação, demonstrado na Figura 4.5. O operador nebuloso "e" utilizado é o mínimo.

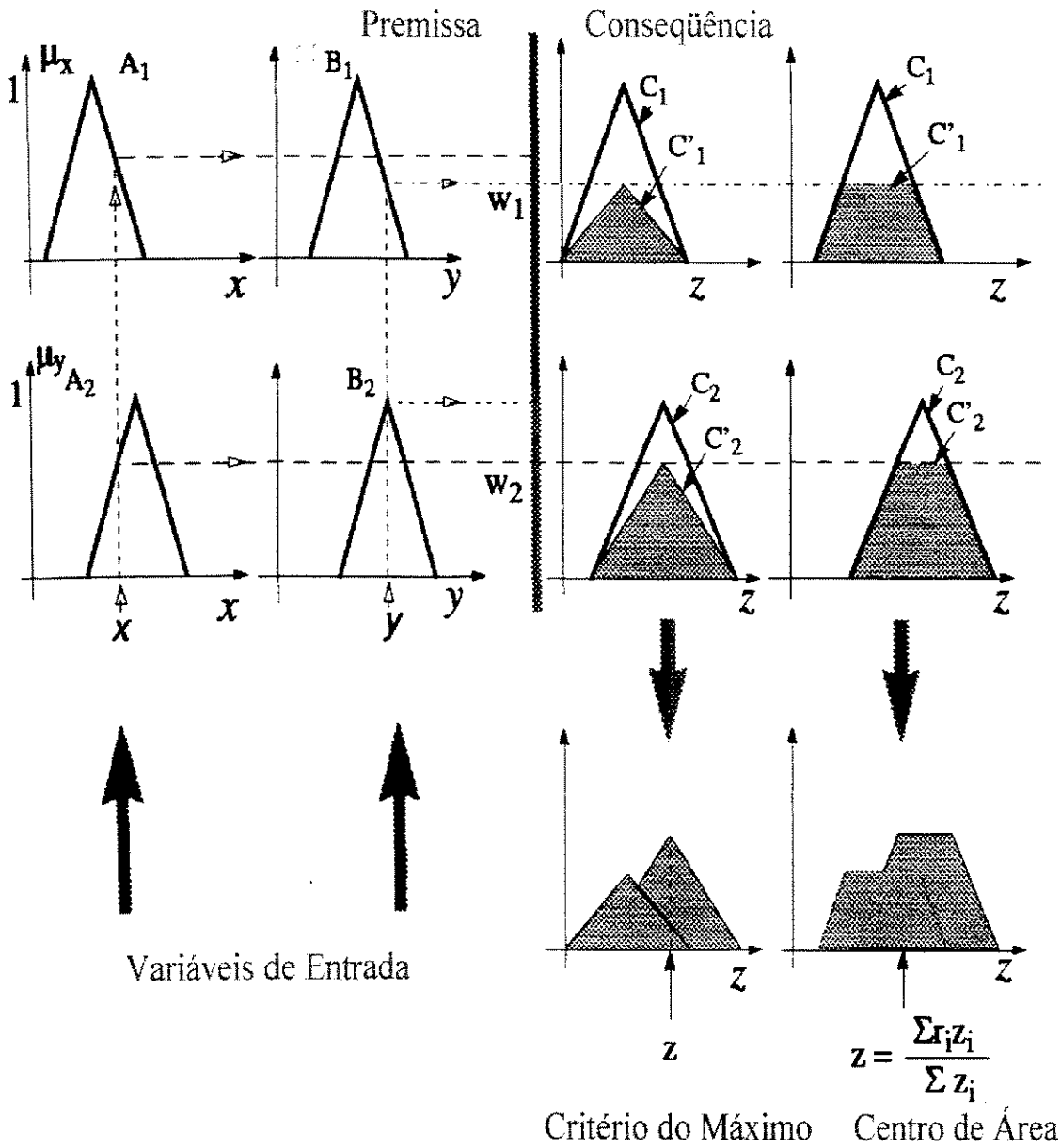


Figura 4.5: Tipos de Defuzificação em Sistemas de Inferência Nebulosa.

4.11 Fatores de Escala ou Limites das Funções de Pertinência

Conforme discutido anteriormente, os controladores nebulosos análogos aos controladores PID tradicionais manipulam variáveis tais como o erro, a variação do erro, a segunda variação do erro, a ação do controlador e a variação na ação do controlador.

Também foi definido que as variáveis lingüísticas tem um universo de discurso, dentro do qual ocorre a partição nos conjuntos nebulosos. Assim, as variáveis manipuladas pelos controladores nebulosos análogos aos PID tradicionais também devem ter seu universo de discurso definido.

Por exemplo, o universo de discurso do erro, e_k é definido pelos valores máximo e mínimo abrangidos pelas funções de pertinência que descrevem o erro.

Supondo que a variável a ser controlada seja fração molar, e que o "setpoint" seja 0.80. Durante o desenvolvimento do controlador nebuloso, o universo de discurso do erro poderia ser definido, por exemplo: $e_{mínimo}$=-0,01 e $e_{máximo}$=+0,01.

Neste trabalho os limites inferior e superior das variáveis erro, variação do erro e segunda variação do erro serão simétricos em relação ao zero, para fins de simplificação de implementação. Isso significa que um limite do erro igual a 0,01 definiria o limite inferior do erro em –0,01 e o limite superior do erro em +0,01.

Esses limites operacionais são definidos com respeito aos valores extremos das variáveis relevantes, e devem ser ajustados levando em conta a dinâmica do sistema controlador e o intervalo de amostragem.

Muitos autores (Yager, 1994; Jager, 1995) recomendam para a simplificação e unificação do projeto de controladores nebulosos o uso de universos de discursos normalizados, tanto das variáveis de entrada quanto de saída dos controladores nebulosos.

Considerando-se que e_k, Δe_k tenham por limites, respectivamente $[-a_e, a_e]$, $[-a_{de}, a_{de}]$, haveria a necessidade de usar os chamados fatores de escala, para transformar essas variáveis para o universo de discurso normalizado [-1, +1]:

$$k_e = 1/a_e \quad \text{(eq. 4.18)}$$

$$k_{de} = 1/a_{de} \quad \text{(eq. 4.19)}$$

Quanto à variável de saída do controlador (por exemplo, Δu_k) obtida pelo algoritmo do controlador nebuloso, cabe salientar que o processo é inverso: a mesma é obtida dentro do universo de discurso normalizado e deve ser transformada de acordo com o seu fator de escala:

$$\Delta u_k = k_{du} \, . \, \Delta u^*_k \quad \text{(eq. 4.20)}$$

onde Δu^*_k é a variação na ação normalizada, e k_{du} o fator de escala, idêntico ao limite da variação da ação.

Os fatores de escala aqui apresentados são relevantes em um ponto de suma importância na utilização dos controladores nebulosos, que é a sintonia.

A sintonia de um controlador nebuloso é um procedimento mais difícil do que a sintonia de um controlador convencional. A razão para isto é que o controlador nebuloso é um sistema extremamente flexível, cujo comportamento é determinado por um grande número de parâmetros que definem as funções de pertinência, a operação de inferência nebulosa e a defuzificação.

Não existe um método para sintonizar um controlador nebuloso. Os resultados mais bem sucedidos são baseados na combinação de um bom conhecimento a respeito do sistema a ser controlado e o uso de analogias entre os controladores nebulosos e os controladores PID tradicionais.

Algumas sugestões, citadas por Procyk e Mamdani (1979) são:

- altos valores de k_e, ou seja, valores pequenos de limites para o erro, resultam em boa resposta do sistema (pequeno erro em regime permanente e pequeno tempo de resposta), mas levam a instabilidade; analogamente pequenos valores de k_e (valores maiores para os limites do erro) levam a uma resposta ruim;

- rápida convergência é alcançada com grandes valores de k_e e k_{de};

- pequenos valores de k_{du} aumentam o tempo de resposta e o valor da integral do erro quadrático.

No entanto a eficiência da sintonia baseada nos fatores de escala é prejudicada pelas características contraditórias que estes fatores exercem nos diferentes índices de medida de desempenho.

4.12 Funções de Pertinência

A literatura sobre controladores nebulosos (Driankov, 1996) cita vários tipos de funções de pertinência, algumas lineares , tais como, funções triangulares, trapezoidais ou monotônicas, outras não lineares, tais como as funções gaussianas.

Cabe analisar quais as vantagens e desvantagens destes tipos. A grande maioria dos trabalhos da literatura utilizam as funções de pertinência lineares, em especial as triangulares. Isso

se explica devido a diversos fatores, tais como: a facilidade de manuseio, e a velocidade de cálculo das áreas envolvidas no processo de defuzificação, etc.

As funções de pertinência trapezoidais, segundo Jager (1995), apresentam uma lenta convergência ao valor de referência, o que em determinados sistemas pode ser indesejável (por exemplo na destilação em batelada), e em outros pode ser uma restrição a ser satisfeita (por exemplo, em alguns sistema elétricos ou robóticos).

Jager (1995) analisa a influência do formato das funções de pertinência e argumenta que o uso de funções de pertinência não lineares introduz um caráter não linear no controlador nebuloso.

Logo o uso de funções de pertinência não lineares seria uma forma de acrescentar não linearidade ao controlador. Mas com certeza não é a única forma, existem outras maneiras de acrescentar características não lineares ao controlador, tais como através da partição do universo de discurso das variáveis de entrada do controlador quando da definição dos conjuntos nebulosos, através das regras nebulosas que representam o conhecimento embutido no controlador, ou ainda através da escolha do tipo apropriado da operação para o conector "e", por exemplo a operação de mínimo, o que embute não linearidade ao controlador (Jager, 1995).

No presente trabalho optou-se por trabalhar com dois tipos de funções de pertinência, as funções de pertinência triangulares e as gaussianas.

4.12.1 Funções de Pertinência Triangulares

Na figura 4.6 está representada a função de pertinência triangular com pico (ou centro) em "a", a distância α, do centro para a esquerda , e a distância β do centro para a direita.

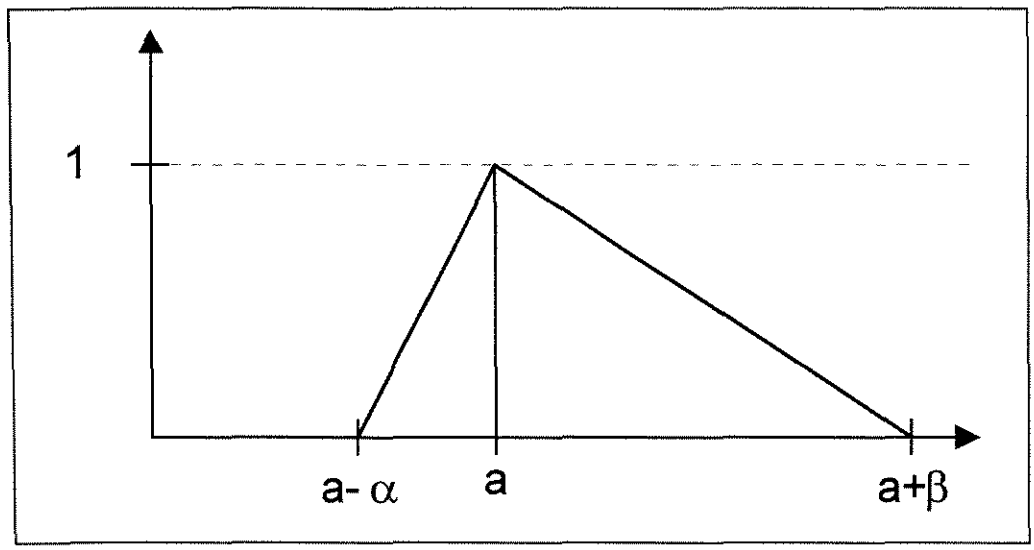


Figura 4.6: Representação Gráfica de uma Função de Pertinência Triangular.

A função de pertinência da figura 4.6 tem as seguintes equações:

$$\mu(x) = 1 - (a - x)/\alpha \text{, se } (a\text{-}\alpha) \leq x \leq a \quad \text{(eq. 4.21)}$$

$$\mu(x) = 1 - (x - a)/\beta \text{, se } a \leq x \leq (a+\beta) \quad \text{(eq. 4.22)}$$

$$\mu(x) = 0 \text{, em outros casos} \quad \text{(eq. 4.23)}$$

Quando da partição do universo de discurso da variável descrita pela função de pertinência triangular, a opção mais simples é dividir o intervalo de interesse num determinado número de funções de pertinência de mesma base, e grau de cruzamento a 0.50, o que facilita em muito a implementação computacional.

4.12.2 Funções de Pertinência Gaussianas

A equação que descreve as funções de pertinência gaussianas pode ser escrita como:

$$\mu(x) = e^{(-(x-a)^2/(2\sigma^2))} \quad \text{(eq. 4.24)}$$

Onde "x" é a variável a ser fuzzificada, "a" é valor de abcissa onde o grau de pertinência é máximo e σ é o parâmetro que indica a o formato da curva, conforme pode ser observado na figura 4.7.

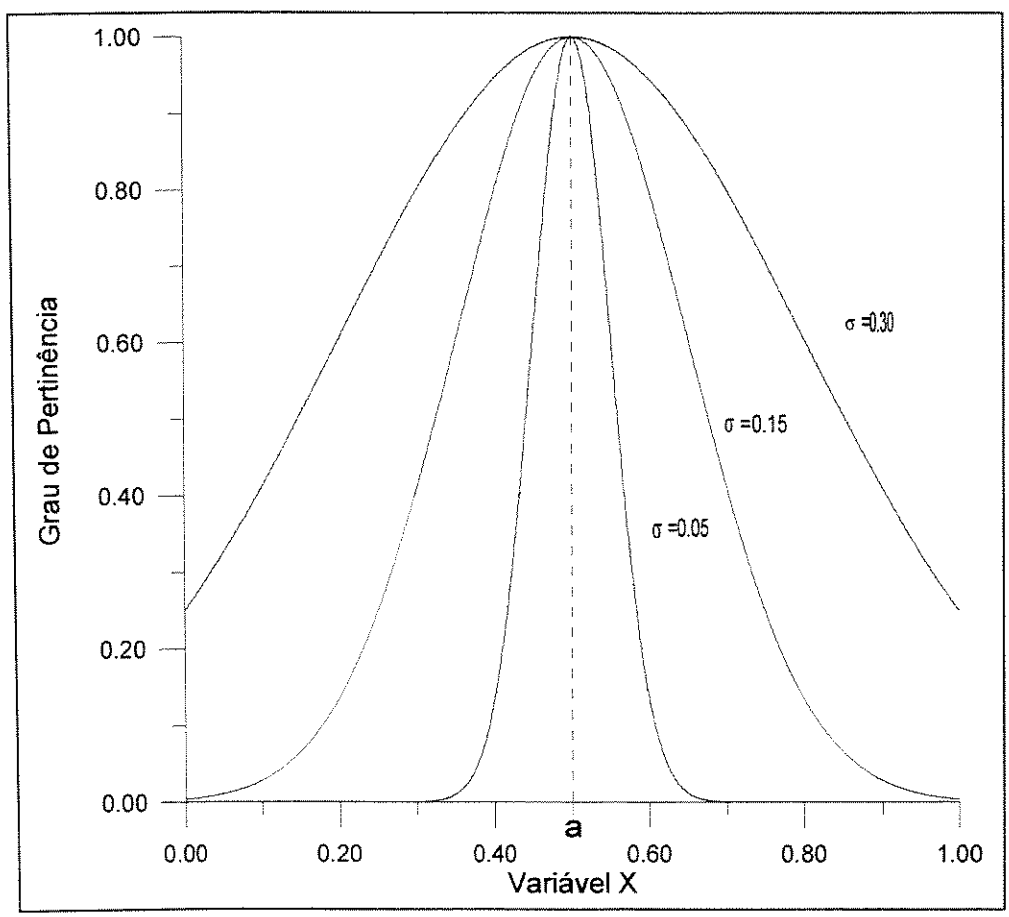


Figura 4.7 - Exemplos de Funções de Pertinência Gaussianas.

4.13 Base de Regras de Mamdani

A literatura (Driankov, 1996) cita três abordagens para obtenção da base de regras a ser utilizada em um controlador nebuloso:

- a abordagem baseada na dedução das regras a partir da experiência de um operador do processo;

- a abordagem que utiliza uma descrição lingüística (modelo nebuloso) do processo a ser controlado;

- a abordagem baseada em um modelo fenomenológico do processo a ser controlado.

A escolha da base de regras utilizada neste trabalho foi fundamentada no trabalho de Mamdani (1975, 1979), o qual utilizou um método intuitivo na determinação das regras a serem usadas no controlador nebuloso.

Antes de determinar a base de regras a ser usada no controlador nebuloso aplicado a destilação batelada, cabe fazer algumas definições. O erro entre a variável controlada (concentração expressa em fração molar) e o valor de referência será definido por:

$$\text{erro} = y_{\text{referência}} - y_{\text{topo}} \quad \text{(eq. 4.25)}$$

onde $y_{\text{referência}}$ é o valor da concentração no "setpoint", e y_{topo} a concentração de interesse no topo da coluna.

A variação do erro será definida como sendo a diferença entre o erro no instante atual e o erro no instante anterior:

$$\Delta\text{erro} = \text{erro}_{\text{atual}} - \text{erro}_{\text{anterior}} \quad \text{(eq. 4.26)}$$

Outra consideração importante a ser feita é a respeito da variável manipulada razão de refluxo. Devido a natureza da destilação em batelada, conforme aumenta a razão de refluxo aumenta também a concentração no topo do componente de interesse e vice-versa.

Para determinar a base de regras a ser utilizada, serão consideradas três opções para as variáveis erro e variação do erro: que as mesmas sejam negativas, zero ou positivas. Combinando todas as opções possíveis fica claro que teremos um máximo possível de 9 regras que serão deduzidas a seguir.

A figura 4.8 abaixo mostra a resposta da variável controlada de um processo, em malha fechada com um controlador nebuloso, e será usada para deduzir a base de regras de um controlador nebuloso.

Devido a dinâmica do processo não se alcança o valor de referência instantaneamente. Isso é resultado de um comportamento comum no controle de processos: ações de controle muito bruscas levam a respostas mais rápidas, mas com a presença de sobrelevação e oscilações.

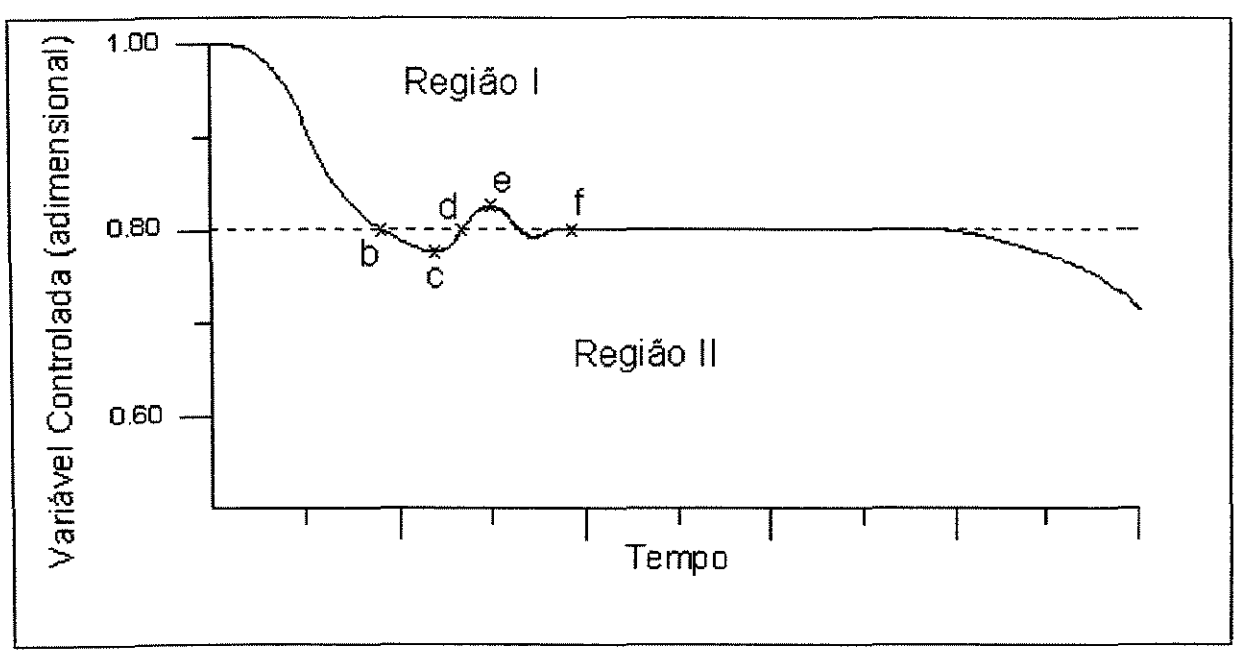


Figura 4.8: Resposta de um Processo em Malha Fechada com Controlador Nebuloso.

O bom senso diz que a partir do ponto "f" onde o sistema apresenta o erro "zero" e a variação do erro também é zero, nenhuma ação deve ser tomada. Por conseguinte se pode formular a regra:

"Se o ERRO é ZERO e a VARIAÇÃO DO ERRO é ZERO então a VARIAÇÃO NA VARIÁVEL MANIPULADA deve ser ZERO"

Analisando o ponto "b" onde o erro é zero, mas a variação do erro é positiva, se nota que haverá uma ultrapassagem do valor de referência. Isso mostra que a ação de controle não foi suficiente para deter a queda da variável controlada, logo a variação da ação deve ser positiva:

"Se o ERRO é ZERO e a VARIAÇÃO DO ERRO é POSITIVA então a VARIAÇÃO NA VARIÁVEL MANIPULADA deve ser POSITIVA".

Do mesmo modo, a partir do ponto "d " haverá sobrelevação, pois o erro é zero, mas a variação do erro é negativa, o que significa que a ação de controle foi muito drástica, logo deve ser reduzida:

"Se o ERRO é ZERO e a VARIAÇÃO DO ERRO é NEGATIVA então a VARIAÇÃO NA VARIÁVEL MANIPULADA deve ser NEGATIVA".

No ponto "c", a variação do erro é nula, mas o erro é positivo, o que indica que a ação deve ficar maior, para diminuir o erro, logo sua variação deve ser positiva:

"Se o ERRO é POSITIVO e a VARIAÇÃO DO ERRO é NULA então a VARIAÇÃO NA VARIÁVEL MANIPULADA deve ser POSITIVA".

No ponto "e" ocorre que novamente a variação do erro é nula, mas o erro é negativo o que indica que a ação está com valor muito alto, logo deve ser reduzida:

"Se o ERRO é NEGATIVO e a VARIAÇÃO DO ERRO é NULA então a VARIAÇÃO NA VARIÁVEL MANIPULADA deve ser NEGATIVA".

Além dos pontos já analisados, pode-se determinar duas regiões, a região I onde o erro é negativo e a região II onde o erro é positivo.

Para a região I, a medida que o erro já é negativo e a variação do erro seja negativa, a variação do erro deve ser negativa já que a variável controlada aproxima-se do valor de referência:

"Se o ERRO é NEGATIVO e a VARIAÇÃO DO ERRO é NEGATIVA então a VARIAÇÃO NA VARIÁVEL MANIPULADA deve ser NEGATIVA".

Ainda na região I, quando a variação do erro é positiva, apesar do erro ser negativo, cabe manter a variável manipulada no valor atual:

"Se o ERRO é NEGATIVO e a VARIAÇÃO DO ERRO é POSITIVA então a VARIAÇÃO NA VARIÁVEL MANIPULADA deve ser ZERO".

Para a região II, quando a variação do erro é positiva, o sistema afasta-se do valor de referência, portanto cabe aumentar o valor da variável manipulada, portanto a variação da variável manipulada deve ser positiva:

"Se o ERRO é POSITIVO e a VARIAÇÃO DO ERRO é POSITIVA então a VARIAÇÃO NA VARIÁVEL MANIPULADA deve ser POSITIVA".

Finalmente, ainda na região II, quando o erro é positivo e a variação do erro é negativa, uma opção é manter a variável manipulada no valor atual, já que o sistema aproxima-se do valor de referência:

"Se o ERRO é POSITIVO e a VARIAÇÃO DO ERRO é NEGATIVA então a VARIAÇÃO NA VARIÁVEL MANIPULADA deve ser ZERO".

Pode-se resumir a base de regras deduzida acima na tabela 4.2 abaixo (onde N = negativo, Z = zero e P = positivo):

		Erro		
		N	Z	P
Δerro	N	***N***	***N***	***Z***
	Z	***N***	***Z***	***P***
	P	***Z***	***P***	***P***

Tabela 4.2: Base de Regras de Mamdani para Cardinalidade 3.

A tabela 4.2 é idêntica a base de regras determinada por Mamdani (1975) e pode ser facilmente ampliada para qualquer cardinalidade.

A metodologia mostrada acima para determinação da base de regras somente serve para problemas pequenos, com um pequeno número de variáveis envolvidas e onde as interações entre as variáveis consideradas sejam claras para o projetista.

A determinação da base de regras é um dos pontos onde outras técnicas de inteligência artificial podem ser acopladas aos controladores nebulosos; por exemplo Godjevac (1995) simplificou o problema da determinação das regras utilizando redes neurais e Gomide (1998) cita os algoritmos genéticos como uma alternativa para a solução deste problema.

4.14 Procedimentos Computacionais de Fuzificação, Inferência e Defuzificação

Como já mostrado anteriormente, o núcleo do controlador nebuloso é composto de 3 etapas, a fuzificação, a inferência nebulosa e a defuzificação.

O controlador nebuloso, da mesma forma que qualquer controlador, transforma uma ou mais variáveis de entrada, em uma ou mais ações, que serão a resposta do controle ao comportamento dinâmico do sistema sob controle.

Essas variáveis de entrada do controlador são, na maioria das vezes em engenharia química, valores numéricos exatos, que para serem manipulados pelo controlador nebuloso devem ser transformados em valores nebulosos. Para isso faz-se uso das funções de pertinência.

Por exemplo, na figura 4.7, considerando um determinado valor de entrada da variável x igual a 0.20, na curva de σ=0.05, teremos um grau de pertinência $\mu_{\sigma=0.05} \approx 0.0$, na curva de σ=0.15, $\mu_{\sigma=0.15} \approx 0.15$ e na curva de σ=0.30, $\mu_{\sigma=0.30} \approx 0.60$.

Suponhamos um controlador nebuloso análogo ao controlador proporcional integral derivativo, que faz uso de três entradas, o erro, a variação do erro e a segunda variação do erro. Após a fuzificação, deve ser feita a inferência, para isso é necessário realizar a operação de composição, que na maioria das vezes é a operação de mínimo.

Na figura 4.9 encontra-se o algoritmo de um processo de inferência nebulosa, para um controlador com três variáveis de entrada, o erro, a variação do erro e a segunda variação do erro (controlador PID nebuloso).

```
Faça  i=1 até funções de pertinência
 Faça j=1 até funções de pertinência
  Faça k=1 até funções de pertinência
   Se pertinência_do_erro(i,3) diferente de 0 então
    Se pertinência_da_variação_do_erro(j,3) diferente de 0 então
     Se pertinência_da_segunda_variação_do_erro(k,3) diferente de 0 então
      Menor= pertinência_do_erro(i,3)
      Se Menor > pertinência_da_variação_do_erro(j,3) então
            menor=   pertinência_da_variação_do_erro(j,3)
      fim se
      Se Menor > pertinência_da_segunda_variação_do_erro(k,3) então
           menor= pertinência_da_segunda_variação_do_erro(k,3)
      fim se
      grau_de_ativação=menor
      valor_da_ação = base_de_regras(j,i,k)
      peso=pertinência_ação(valor_da_ação,2)
      if  (i=1) ou (i=funções) então base=pa(i,1)/2 senão  base=pa(i,1)
      area=((base*(1-grau_de_ativação))+base)*(grau/2)
      atot=atot + area
      watot=watot + area*peso
     fim se
    fim se
   fim se
fim faça
fim faça
fim faça
acao = watot / atot
```

Figura 4.9: Algoritmo da Inferência e da Defuzificação para o Controlador PID Nebuloso.

A inferência nebulosa usando a operação de mínimo para o conectivo "e" encontra-se nas linhas em marcadas em vermelho. Esta operação determina o grau de ativação para cada uma das regras.

Na defuzificação é usado o método do centro de área, que necessita calcular a área compreendida entre a função de pertinência da ação que corresponde à determinada pela matriz da base de regras e o seu grau de ativação.

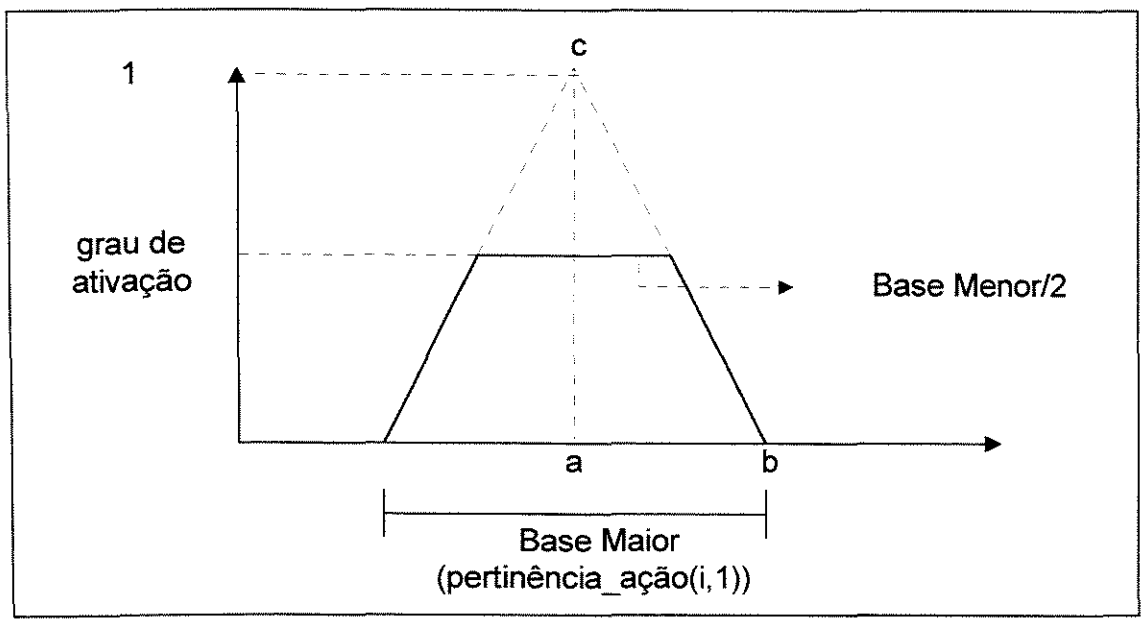


Figura 4.10: Área Abaixo da Função de Pertinência da Ação na Forma de Triângulo Isósceles.

Esse cálculo de área envolvido no processo de defuzificação, nada mais é do que o cálculo da área de um trapézio, na figura 4.10, do qual se conhece a altura (grau de ativação), a base maior (pertinência_ação(i,1) é o suporte da função de pertinência triangular) mas não se conhece a base menor.

Examinando a figura 4.10 rapidamente observa-se que é possível calcular a base menor a partir do triângulo "abc":

$$\frac{BaseMaior/2}{1} = \frac{BaseMenor/2}{1 - grau\_de\_ativação} \qquad \text{(eq. 4.27)}$$

Por conseguinte a área do trapézio é

Área = (base menor + base maior)*grau de ativação /2 (eq. 4.28)

A implementação da equação acima encontra-se em azul na figura 4.9 .

O centro da função de pertinência da ação usada para cálculo da área ("a" na figura 4.10) corresponde ao elemento pertinência_ação(i,2) na figura 4.9 e também é usado no processo de defuzificação.

Quando funções de pertinência triangulares quaisquer são utilizadas, o procedimento é semelhante e será mostrado a seguir.

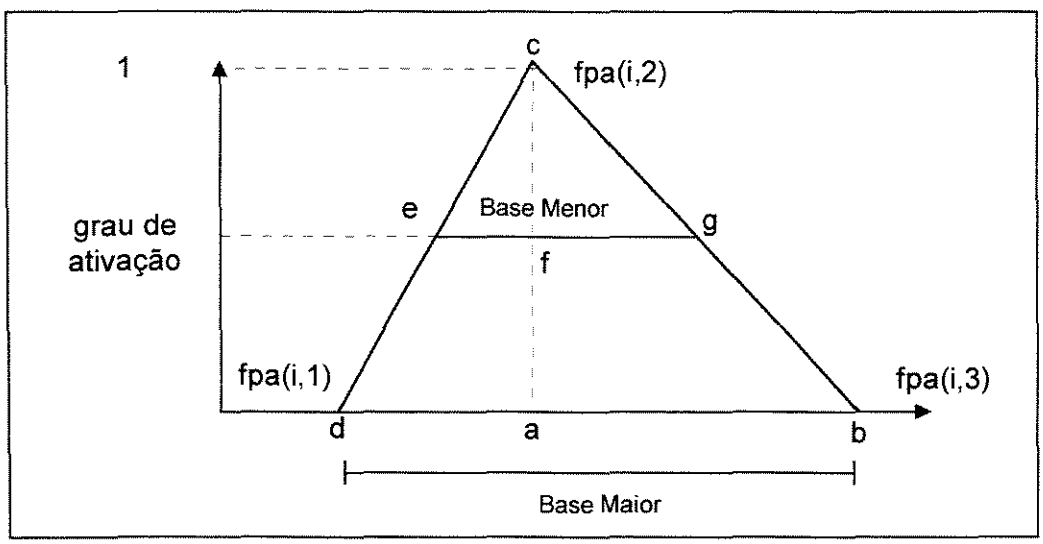


Figura 4.11: Área Abaixo da Função de Pertinência da Ação na Forma de Triângulo Escaleno.

A variáveis fpa(i,1), fpa(i,2) e fpa(i,3) na figura 4.11 correspondem respectivamente aos vértices da função de pertinência triangular "dcb", onde i é o número da função de pertinência.

Novamente o cálculo da área reduz-se ao cálculo da área de um trapézio (de vértices degb na figura 4.11).

O intervalo db, a base maior do trapézio, é conhecido pois é a diferença fpa(i,3) – fpa(i,1), mas a base menor eg não é diretamente conhecida; é necessário calcular a base menor com base nos dois triângulos diferentes dac e acb.

Para o triângulo esquerdo dca:

$$\frac{fpa(i,2) - fpa(i,1)}{1} = \frac{ef}{1 - grau} \quad \text{(eq. 4.29)}$$

O segmento ef corresponde ao lado esquerdo da base menor do trapézio. Fazendo o mesmo procedimento para calcular o segmento restante da base menor, fg, no triângulo do lado direito da figura:

$$\frac{fpa(i,3) - fpa(i,2)}{1} = \frac{fg}{1 - grau} \quad \text{(eq. 4.30)}$$

A base menor do trapézio é então conhecida:

$$\text{Base menor} = ef + fg \quad \text{(eq. 4.31)}$$

Se porventura no projeto do controlador nebuloso forem consideradas funções de pertinência não lineares, o problema do cálculo das áreas pode tornar-se complexo e passar a ser uma restrição muito importante no desempenho do controlador "on-line".

Para a defuzificação envolvendo funções de pertinência gaussianas utilizadas em algumas implementações durante este estudo, utilizou-se o método de Simpson de integração numérica, que se encontra no anexo B.

O uso de um método numérico iterativo aumenta muito o tempo gasto na rotina de defuzificação, o que se torna um fator restritivo para utilizar controladores nebulosos baseados em funções de pertinência não lineares, em especial com cardinalidades muito grandes.

5. Resultados Obtidos em Simulação

5.1 Introdução

Para acompanhar a influência de diversos fatores no desempenho dos controladores nebulosos, foram realizadas simulações computacionais da destilação em batelada binária de misturas compostas de n-hexano e n-heptano, de composição inicial igual a 30% molar de n-hexano e 70% molar de n-heptano, usando uma coluna de 11 estágios de equilíbrio, com "setpoint" igual a 80% molar de n-hexano no topo, utilizando o programa simulador desenvolvido por Fileti (1992) em linguagem C, acoplado ao módulo de controle nebuloso desenvolvido durante esse trabalho em linguagem FORTRAN (a listagem encontra-se no Anexo C).

A linguagem FORTRAN foi escolhida por que o trabalho anterior de Pedrosa (1998), envolvendo o mesmo equipamento experimental, já utilizou a linguagem FORTRAN 90. Desta forma o módulo de controle nebuloso poderia ser mais facilmente implementado experimentalmente.

Para trabalhar em conjunto com as linguagens C e FORTRAN foi utilizado o compilador de domínio público DJGPP – GNU , em conjunto com o ambiente de desenvolvimento de programas RHIDE.

5.2 A Base de Regras e o Desempenho do Controlador PI Nebuloso

As regras utilizadas no controlador nebuloso são de importância fundamental no desempenho deste controlador pois correlacionam premissas e ações, sendo de difícil determinação, como já visto anteriormente.

Para facilitar a avaliação de como a resposta dinâmica é influenciada pela base de regras do controlador, há a necessidade de se trabalhar como um número de regras relativamente pequeno (por exemplo cardinalidade 5), e com um controlador PI nebuloso, já que para o

mesmo o número máximo de regras seria, no exemplo de 25 (5^2) , enquanto para um controlador PID nebuloso esse número seria de 125 (5^3) .

Para servir como referência, utilizou-se a mesma base de regras do trabalho de Mamdani (1975), utilizando cardinalidade 5 (os 5 adjetivos nebulosos seriam: negativamente grande, NG, negativamente médio, NM, zero, Z, positivamente médio, PM, e positivamente grande, PG) para as três variáveis manipuladas pelo controlador: o erro, a variação do erro e a variação na ação. Essa base de regras encontra-se na tabela 5.1 abaixo.

		ERRO				
		NG	**NM**	**Z**	**PM**	**PG**
ΔERRO	**NG**	*NG*	*NG*	*NG*	*NM*	*Z*
	NM	*NG*	*NG*	*NM*	*Z*	*PM*
	Z	*NG*	*NM*	*Z*	*PM*	*PG*
	PM	*PM*	*Z*	*PM*	*PG*	*PG*
	PG	*Z*	*PM*	*PG*	*PG*	*PG*

Tabela 5.1: Base de Regras do Controlador PI Nebuloso ,Mamdani (1975).

Para avaliar o que ocorre no comportamento do controlador quando as regras são modificadas, no gráfico 5.1 estão os perfis de concentração e razão de refluxo, tanto para a base de regras de referência (tabela 5.1) e uma base de regras modificada, onde as ações NG (em azul na tabela 5.1) foram substituídas por NM e as ações PG (em vermelho na tabela 5.1) substituídas por ações PM .

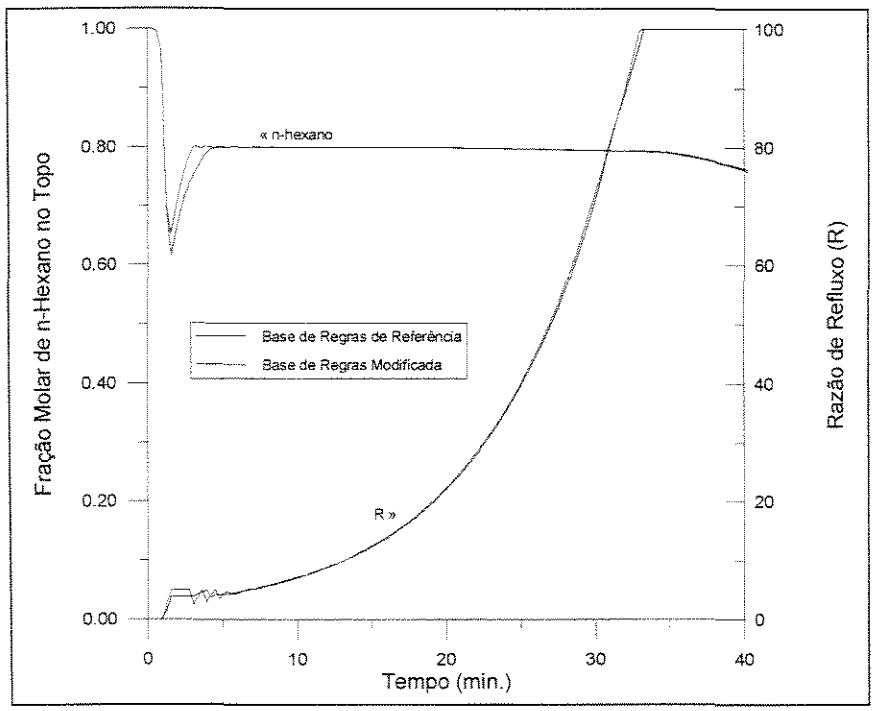


Gráfico 5.1: Perfis de Concentração e Razão de Refluxo x Modificações na Base de Regras.

A simulação usando o controlador PI nebuloso com a base de regras modificada, apresenta um tempo de resposta maior, quando comparado com o comportamento do sistema usando o controlador PI nebuloso com a base de regras de referência. Isso ocorre devido a necessidade do controlador atribuir valores para ação positivamente grandes quando a concentração cai rapidamente abaixo do valor de referência.

As alterações na base de regras testadas na simulação do gráfico 5.1 tiveram uma influência bastante grande em uma região afastada do valor de referência, por que foram modificadas as regras que são ativadas quando o erro e a variação do erro caem nos extremos dos seus universos de discurso . Pode-se concluir que, por analogia , substituindo-se as ações negativamente médias por negativamente grandes e ações positivamente médias por positivamente grandes, que as diferenças de comportamento se darão numa região bem próxima ao valor de referência da concentração, como mostrado no gráfico 5.2 .

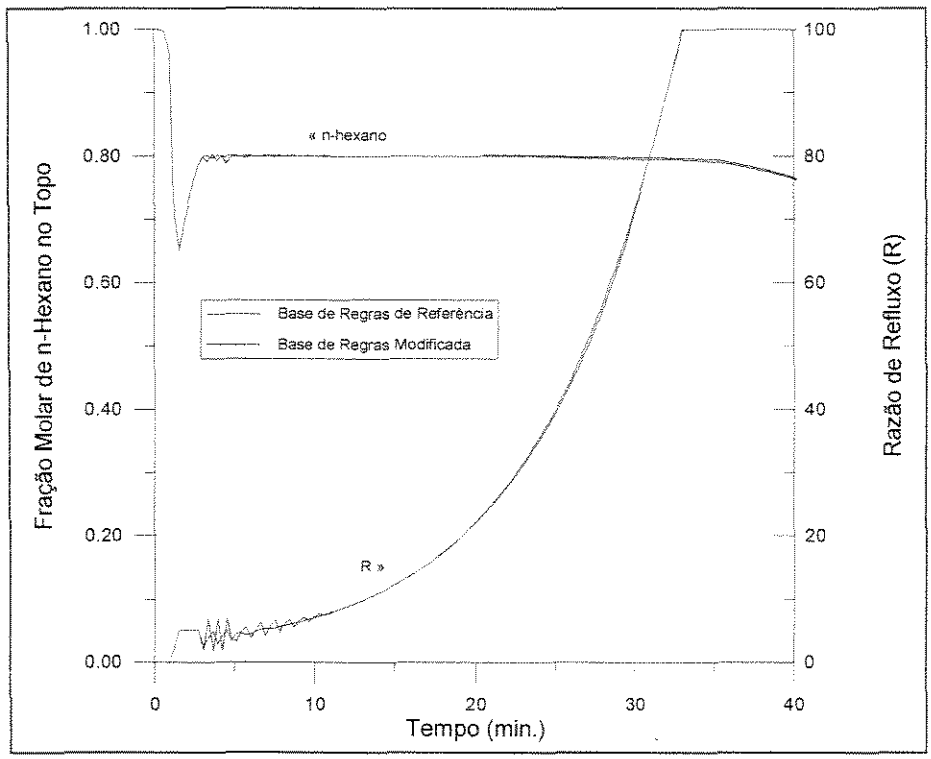


Gráfico 5.2: Perfis de Concentração e Razão de Refluxo x Modificações na Base de Regras.

5.3 Fatores de Escala e os Controladores Nebulosos

A fim de avaliar o desempenho de um controlador nebuloso de ação proporcional e integral e determinar a influência dos fatores de escala do erro e da variação do erro, foi realizado um conjunto de simulações computacionais, onde todos os parâmetros do processo e controlador nebuloso (cardinalidade 7, funções de pertinência triangulares igualmente espaçadas, método de defuzificação centro de área) eram idênticos, mas se utilizaram fatores de escala diferentes.

Para acompanhar a resposta do controlador quando das modificações nos fatores de escala e portanto nos limites das variáveis lingüísticas, utilizaram-se alguns índices de desempenho, os quais também foram utilizados por Pedricz (1993). Esses fatores foram:

a) o somatório do erro quadrático entre a variável controlada e o valor de referência:

$$Q_1 = \sum_{i=0}^{k} e^2(i) \quad \text{(eq. 5.1)}$$

b) o maior desvio positivo relativo na resposta do controle:

$$Q_2 = \frac{y_{max} - y_{referência}}{y_{referência}} \quad \text{(eq. 5.2)}$$

O índice Q_2 (maior desvio positivo) detecta tanto a presença de sobrelevação quanto a resposta dinâmica do sistema oscilante não amortecida (oscilação crescente). .

Pelo gráfico 5.3, obtido a partir de várias simulações da destilação em batelada do sistema descrito acima (em malha fechada por controlador nebuloso com diferentes limites das variáveis lingüísticas), nota-se que tanto o somatório do erro quadrático (Q_1) quanto o maior desvio positivo apresentam alguns mínimos locais, o que dificulta em muito a busca pelo mínimo global a fim de otimizar o desempenho do controlador em termos do limite do erro.

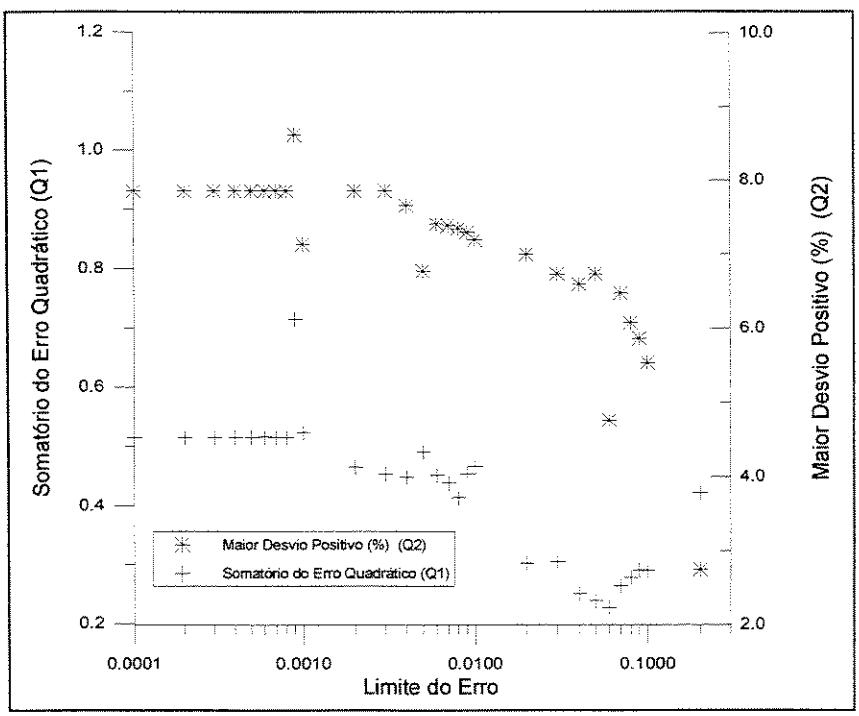


Gráfico 5.3: Somatório do Erro Quadrático e Maior Desvio Positivo Relativo x Limite do Erro.

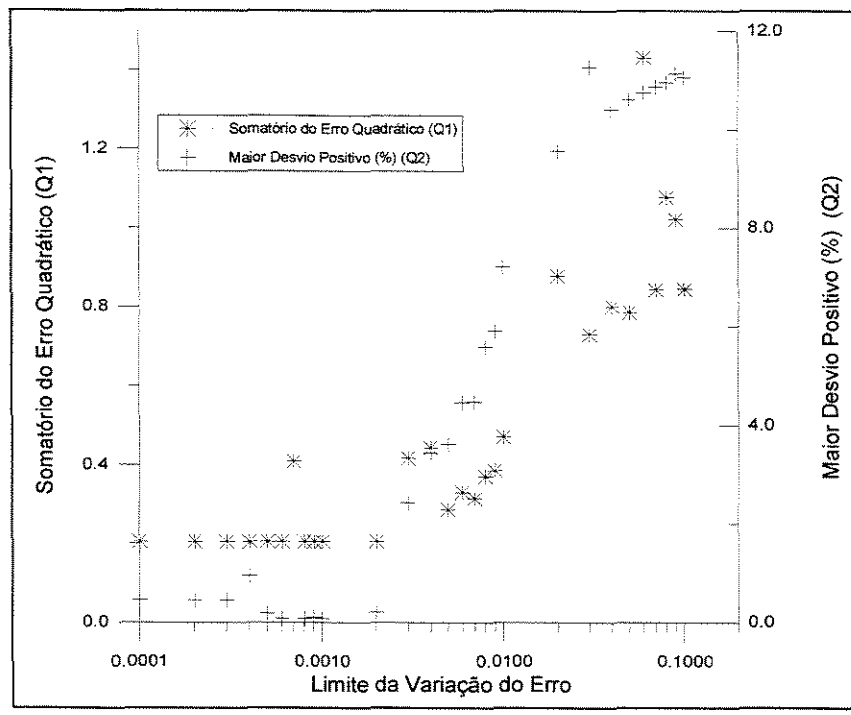


Gráfico 5.4: Somatório do Erro Quadrático e Maior Desvio Positivo Relativo x Limite da Variação do Erro

O mesmo acompanhamento dos índices Q_1 e Q_2 foi feito em função do limite da variação do erro (gráfico 5.4) . O somatório do erro quadrático (Q_1) apresenta um perfil crescente conforme aumenta o limite da variação do erro, enquanto o maior desvio positivo (Q_2), apresenta uma região de mínimo.

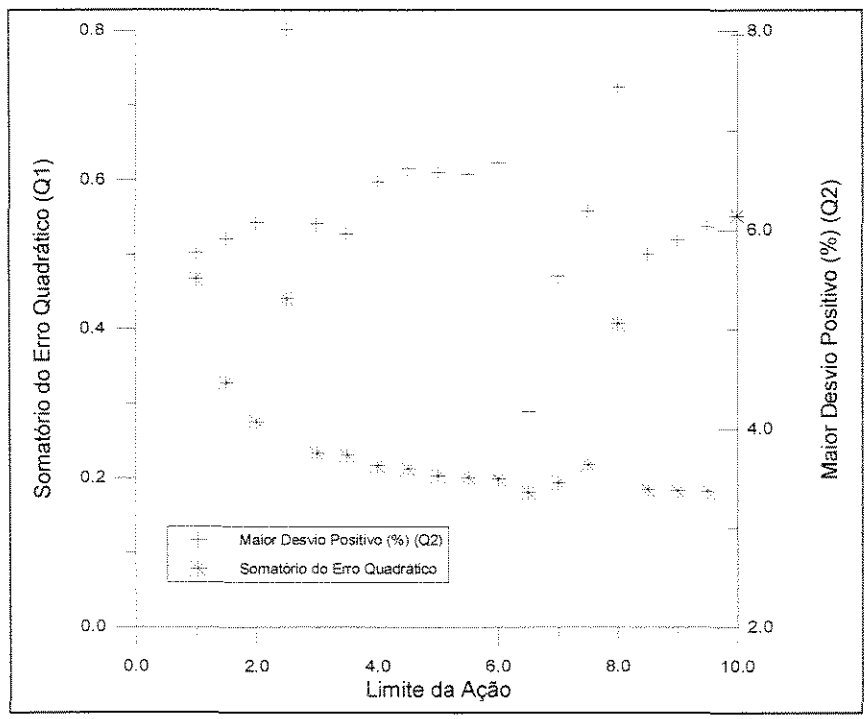


Gráfico 5.5: Somatório do Erro Quadrático e Maior Desvio Positivo (%) x Limite da Ação.

O fator de escala referente a ação (variável de saída do controlador), o limite da ação, também influencia os índices Q_1 e Q_2, conforme o gráfico 5.5. É interessante notar que mantidos constante o limite do erro e o limite da variação do erro, encontrou-se um ponto ótimo para o limite da ação do erro, ao redor do valor 6.5.

Esse comportamento pode ser explicado, já que para valores pequenos do limite da ação, o controlador apresentará uma atuação mais suave, significando ações pouco bruscas, mas também pouco efetivas, já que o somatório do erro quadrático é relativamente grande, mas o percentual do maior desvio positivo não é muito grande.

Para valores altos de limite da ação, as variações da variável manipulada serão em média maiores. Isto pode significar inclusive a presença de oscilação, o que aumenta tanto o maior desvio positivo quanto o somatório do erro quadrático.

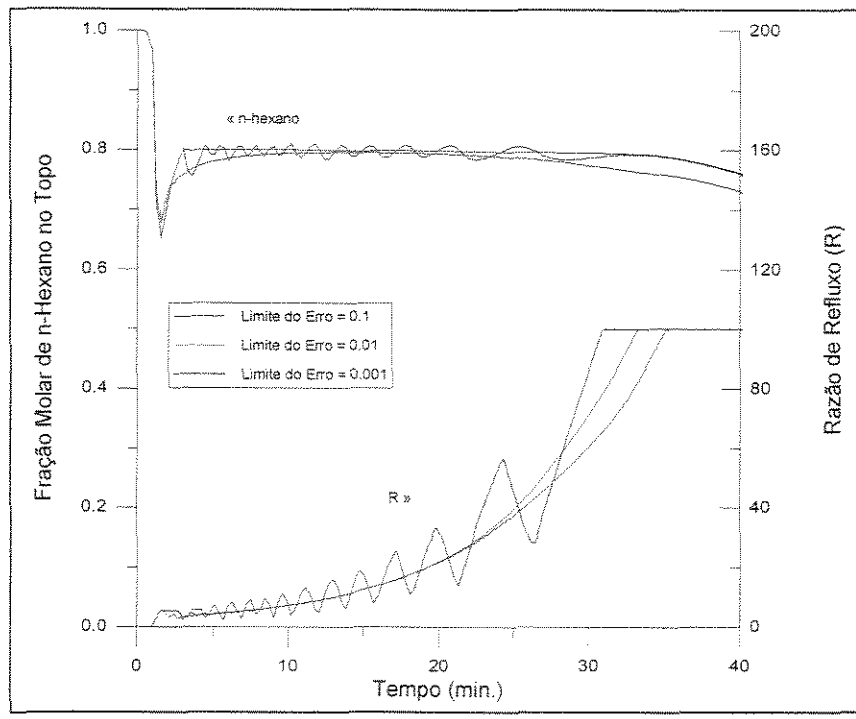


Gráfico 5.6: Perfis de Composição e Razão de Refluxo x Limite do Erro.

A resposta da coluna de destilação em malha fechada com um controlador nebuloso análogo ao PI, em termos dos perfis de composição e de razão de refluxo em função de três opções para o parâmetro limite do erro, encontra-se no gráfico 5.6.

Fica nítido que para um valor de limite do erro muito estreito (0,001) há oscilação na resposta do sistema e que para um valor muito grande (0,1) o tempo de resposta do sistema é muito ruim. Neste último caso, a resposta inclusive não chega a alcançar o valor de referência (0,80).

As curvas de razão de refluxo para limite do erro igual a 0,01 e limite do erro igual a 0,1 no intervalo de tempo que inicia no tempo de 5 minutos e vai até aproximadamente o tempo de 22,5 minutos são muito semelhantes, apesar dos comportamentos da composição de n-hexano tão diferentes. Isso ocorre devido as ações do controlador tomadas no intervalo de tempo anterior ao tempo de 5 minutos.

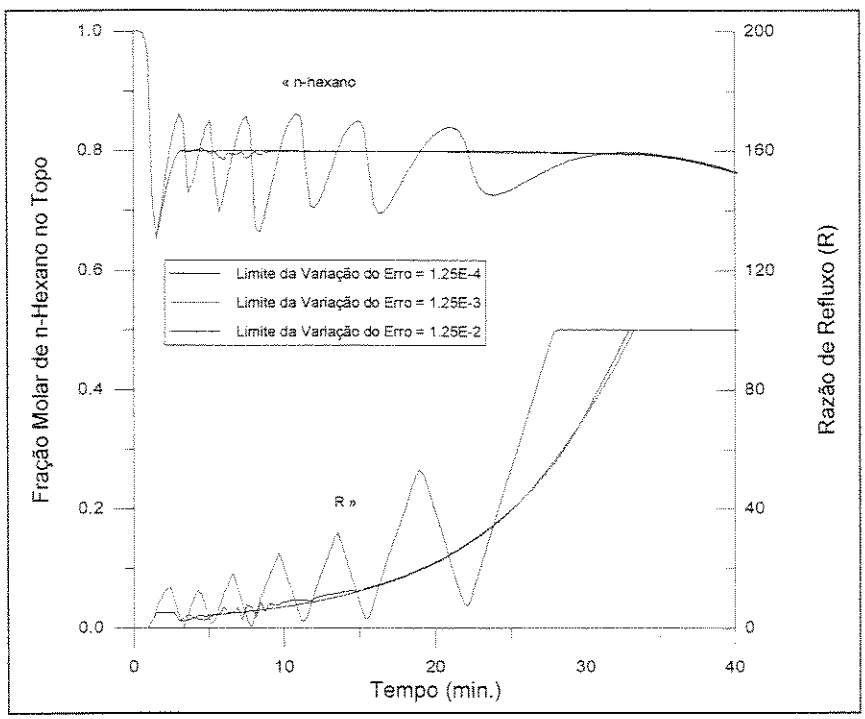


Gráfico 5.7: Perfis de Composição e Razão de Refluxo x Limite da Variação do Erro.

Avaliando a influência do limite da variação do erro nos perfis de composição e razão de refluxo (gráfico 5.7), nota-se que esse parâmetro provoca instabilidade, o que é observado pelo comportamento das curvas para a situação de limite da variação do erro igual a $1{,}25 \times 10^{-2}$.

Quando se compara a situação de limite da variação do erro igual a $1{,}25 \times 10^{-3}$ e limite da variação do erro igual a $1{,}25 \times 10^{-4}$, verifica-se que essa segunda condição implica em um desempenho inferior ao redor do "setpoint".

A análise conjunta dos gráficos 5.6 e 5.7 permite concluir que a redução excessiva do limite do erro, juntamente com o aumento excessivo do limite da variação do erro provocam problemas de estabilidade (oscilação). Analogamente, um aumento excessivo do limite do erro, juntamente com uma redução excessiva do limite da variação do erro provocam uma atuação ineficiente do sistema, tanto em tempo de resposta, quanto à presença do maior desvio positivo relativo .

No gráfico 5.8 estão os perfis de composição e razão de refluxo para três simulações utilizando controladores nebulosos PI de mesmo limite do erro, mesmo limite da variação do erro e limite da ação diferentes (2,0 , 6,5, 10,0). Fica clara a grande influência deste parâmetro no desempenho do controlador, inclusive para a situação de limite da ação igual a 10,0 verifica-se uma forte instabilidade no desempenho do controlador.

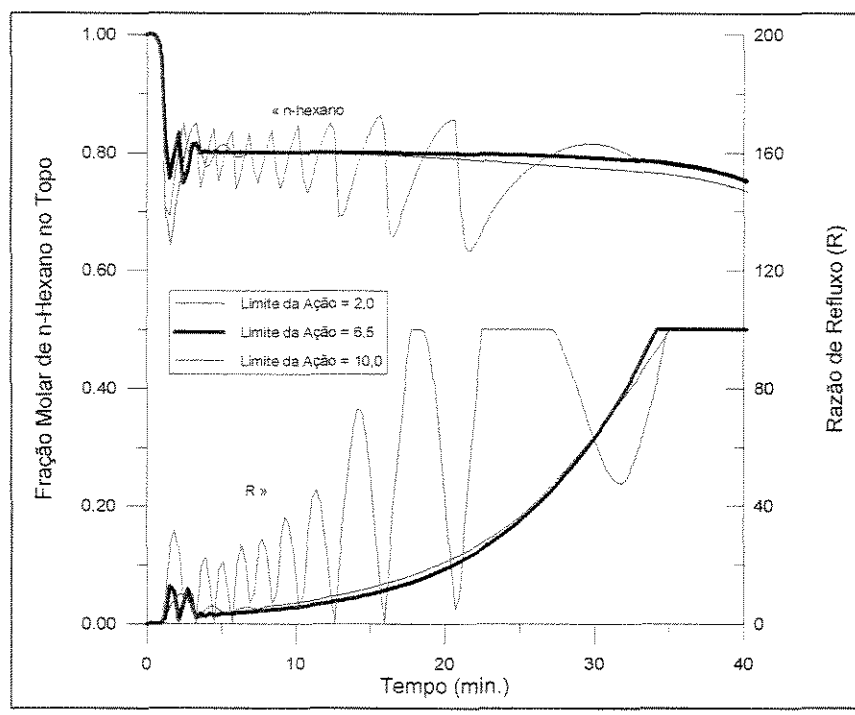


Gráfico 5.8: Perfis de Composição e Razão de Refluxo x Limite da Ação.

Esse comportamento concorda com a análise da influência dos fatores de escala realizada por Pedrycz (1983) e também Yager (1994).

Yager (1994) acrescenta a informação importante de que os fatores de escala (limite do erro, limite da variação do erro e limite da ação) podem implicar em uma sintonia conflitante, de acordo com a análise de vários índices de desempenho, por exemplo, a redução do maior desvio positivo mas com um aumento do somatório do erro quadrático.

Maneiras de resolver essa situação seriam por exemplo utilizar fatores de escala não lineares, ou ajustar o posicionamento das funções de pertinência dentro do universo de discurso de uma forma mais aproximada à região do valor de referência .

5.4 Cardinalidade e o Desempenho do Controlador PI Nebuloso

A cardinalidade representa um parâmetro muito importante no projeto de um controlador nebuloso, pois pode interferir tanto na qualidade da resposta do controlador, quanto no tempo computacional gasto no algoritmo de controle, e como já mencionado isso é um fator muito importante no controle "on-line".

A literatura (Driankov, 1996) menciona o uso de cardinalidades pequenas (tais como 3, 5 ou 7 funções de pertinência na definição de um conjunto nebuloso), devido ao crescimento excessivo da base de regras em cardinalidades maiores, aos tempos maiores de processamento e às dificuldades de sintonia. Entretanto avaliou-se a resposta do controlador com cardinalidades maiores (gráfico 5.9) , onde todos os outros fatores de projeto do controlador permaneceram constantes.

Observa-se no gráfico 5.9 que, para os fatores de escala (limite do erro e limite da variação do erro) utilizados nas simulações efetuadas, a cardinalidade tem uma influência pequena no desempenho do controlador.

Jager (1995) menciona que não existem critérios sistemáticos de se escolher a cardinalidade de um controlador nebuloso, e que essa escolha deverá ser feita pelo projetista do controlador com base no seu conhecimento da dinâmica do processo a ser controlado.

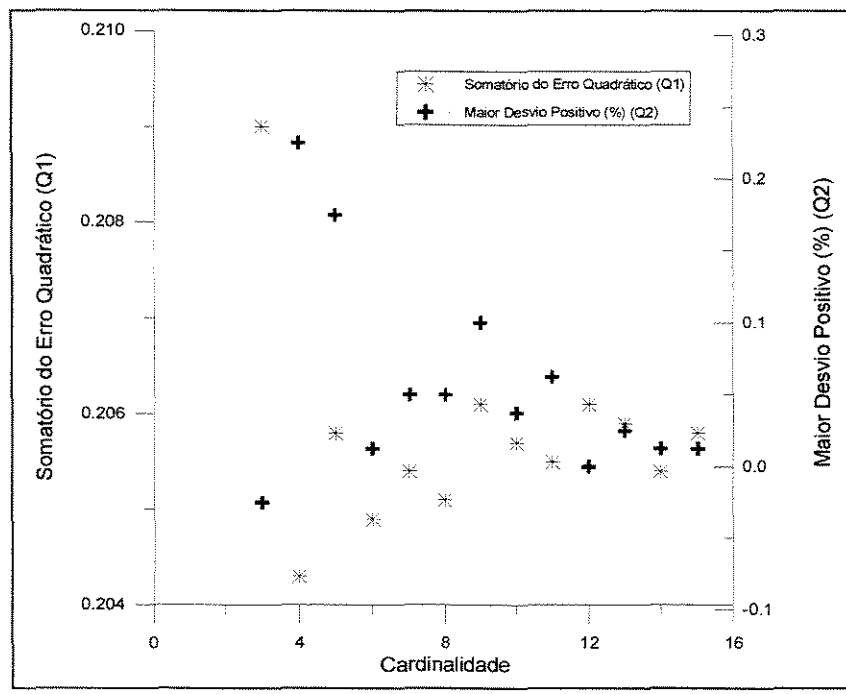


Gráfico 5.9: Influência da Cardinalidade no Desempenho do Controlador PI Nebuloso.

5.5 Grau de Cruzamento e Desempenho do Controlador PI Nebuloso

O grau de cruzamento ("crossover") é outro parâmetro que deve ser levado em consideração no projeto do controlador nebuloso. Jager (1995) identificou o grau de cruzamento como fundamental para a especificação das características do controlador nebuloso. Para avaliar a influência deste parâmetro no desempenho do controlador, foram feitas várias simulações. Verificou-se então como os índices somatório do erro quadrático e o maior desvio positivo relativo comportam-se frente ao grau de cruzamento variável (Gráfico 5.10).

Da observação menos cuidadosa do comportamento do somatório do erro quadrático no gráfico 5.10, pode-se inferir que graus de cruzamento baixos (menores que 50%) não prejudicam o desempenho do controlador, mas muitos autores (Jager, 1995; Yager, 1994; Driankov, 1996) recomendam o valor mínimo de "crossover" de 50%. A razão para isso fica clara no gráfico 5.11, onde tem-se o perfil da fração molar de n-hexano no topo, para três graus de cruzamento (10%, 50% e 90%). Para o "crossover" de 10%, há instabilidade na resposta do sistema sob controle. Essa característica se repetiu em todos os casos em que o "crossover" usado foi menor do que 50%.

Os valores mostrados de maior desvio positivo no gráfico 5.10 são pequenos, o que poderia significar que o grau de cruzamento não influência o mesmo. Mas o índice de desempenho maior desvio positivo detecta também a presença de instabilidade. Então para graus de cruzamento menores do que 30% a avaliação do índice de performance Q_2 mostra a presença de instabilidade.

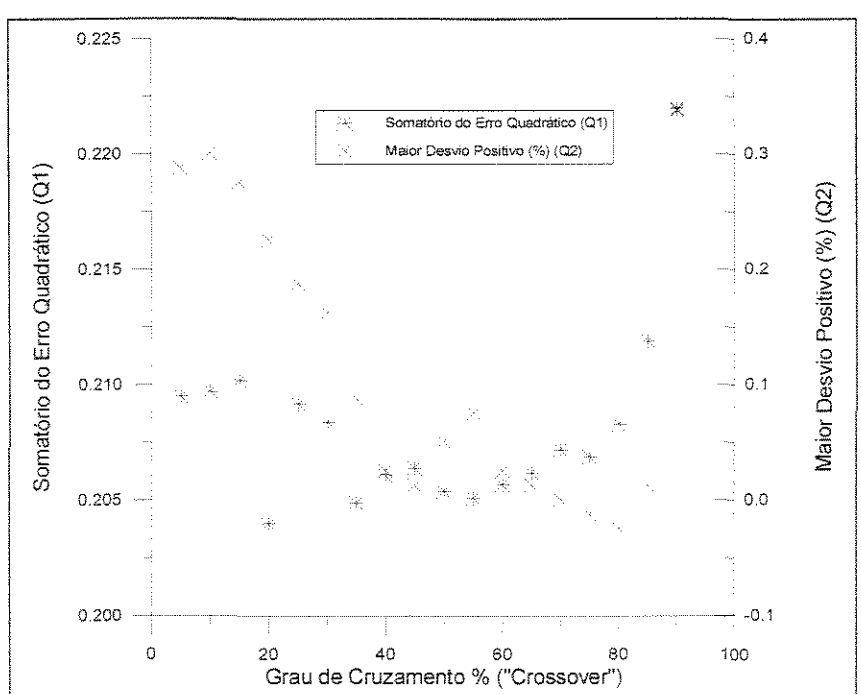


Gráfico 5.10: Somatório do Erro Quadrático e Maior Desvio Positivo Relativo x Grau de Cruzamento.

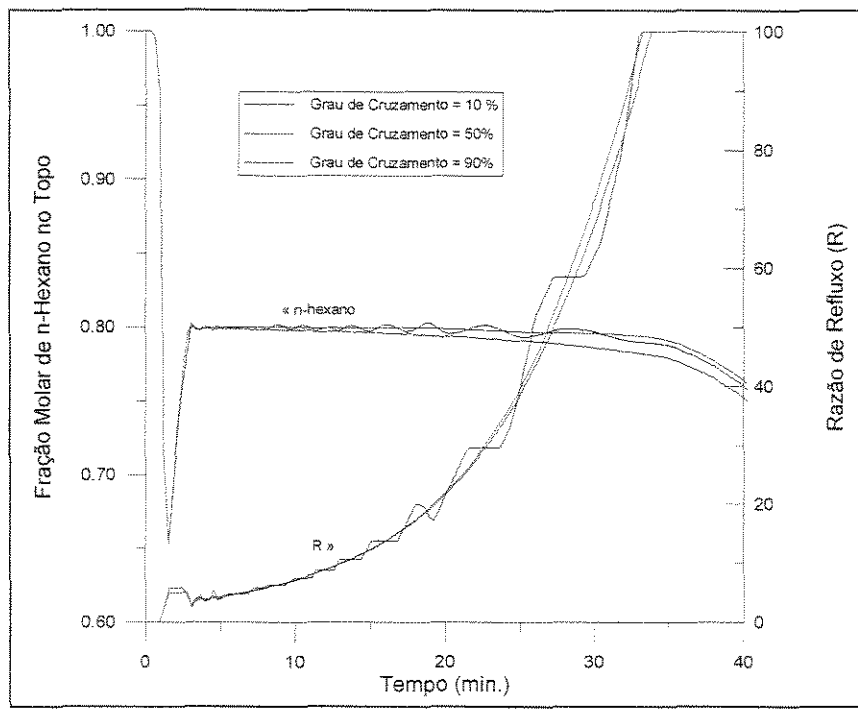


Gráfico 5.11: Composição no Topo e Razão de Refluxo para "Crossover" de 10%, 50% e 90%.

Para graus de cruzamento maiores do que 75% ocorre o rápido crescimento tanto do maior desvio positivo, quanto do somatório do erro quadrático (gráfico 5.10). O desempenho ruim do controlador para graus de cruzamento próximos a 1.0 fica clara no gráfico 5.11. A resposta do sistema usando um controlador nebuloso de "crossover" igual a 90% é muito ruim, inclusive o mesmo não consegue manter o valor de referência.

5.6 Tipos de Funções de Pertinência e o Desempenho do Controlador Nebuloso PI

Conforme já mencionado no transcorrer desta dissertação, foram testados dois tipos de funções de pertinência: a triangular (linear) e a gaussiana (não linear). É útil determinar qual dos dois tipos é mais apropriado para o uso no controlador "on-line".

A literatura (Driankov, 1996) cita que a grande maioria dos trabalhos utiliza funções de pertinência lineares, sejam elas triangulares, trapezoidais ou monotônicas. Isso pode ser justificado devido a facilidade de manipular tais funções em especial quando da defuzificação. Por outro lado, usar funções de pertinência não lineares pode acrescentar características não lineares ao controlador nebuloso, de acordo com as necessidades do projetista.

Uma característica que difere em muito quando do uso das funções de pertinência triangulares ou das funções de pertinência gaussianas é o tempo computacional gasto durante a defuzificação. Esse fator é de pouca importância durante as simulações, mas fundamental quando do teste experimental de um controlador nebuloso.

No gráfico 5.12 são apresentados os perfis de fração molar de n-hexano no topo e razão de refluxo para duas simulações utilizando todos os parâmetros do controlador idênticos (limite do erro, limite da variação do erro, cardinalidade e "crossover"), mas funções de pertinência de tipos diferentes. Pode-se observar que o desempenho do controlador usando funções de pertinência triangulares é muito parecida a do controlador utilizando funções de pertinência gaussianas, mas nesse segundo caso há a presença de uma pequena oscilação na resposta do sistema. Isto deixa claro que a sintonia dos outros parâmetros de projeto deveria ser refeita no caso de utilização do controlador usando funções de pertinência gaussianas.

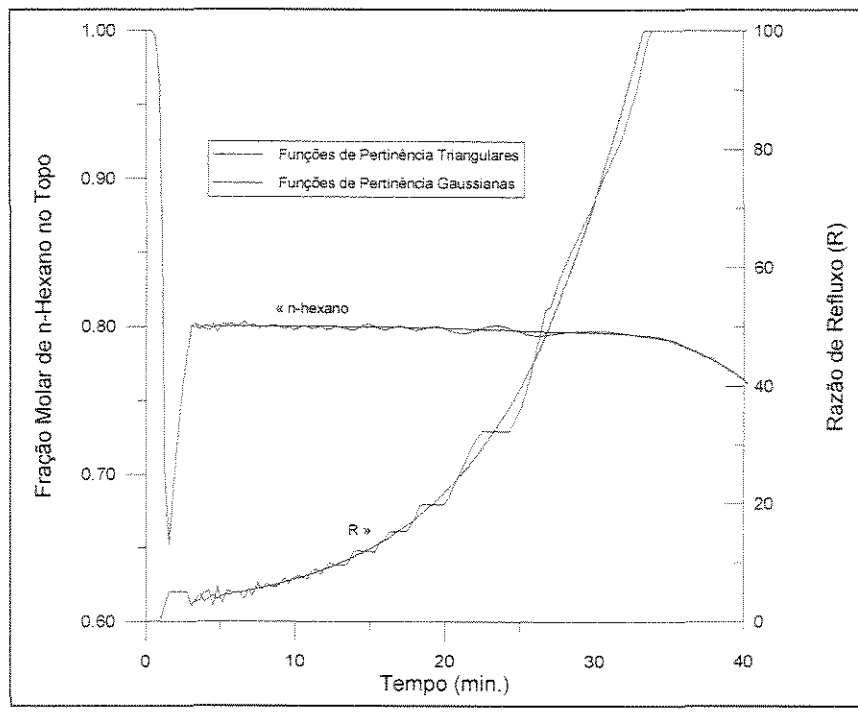


Gráfico 5.12: Composição e Razão de Refluxo x Funções de Pertinência Triangulares ou Gaussianas.

5.7 Conclusões

A etapa de simulação permitiu avaliar alguns parâmetros de projeto dos controladores nebulosos desenvolvidos, tais como os parâmetros de escala (limite do erro, limite da variação do erro, e limite da ação do controlador), cardinalidade, tipos de funções de pertinência, regras usadas no controlador, a fim de permitir um melhor domínio das influências e interações destes parâmetros com o desempenho dos controladores nebulosos.

Os resultados em simulação demonstraram que os parâmetros de escala têm uma grande influência na resposta dinâmica do controlador, no que tange a presença ou não de comportamento oscilatório, manutenção do valor de referência, etc. O uso de limites do erro

muito estreitos favorecem a presença do comportamento oscilatório, enquanto que valores muito amplos determinam valores de erro quadrático muito grandes, pois não é mantido o "setpoint". Para o limite da variação do erro, o uso de valores muito amplos ou muito estreitos implicam na presença de oscilações.

A análise do comportamento da coluna de destilação em batelada, em simulação, em função da cardinalidade dos controladores nebulosos testados, comprovou a dificuldade em determinar qual a cardinalidade que deveria ser utilizada nas corridas experimentais; ao mesmo tempo as simulações determinaram qual o grau de cruzamento ("crossover") a ser utilizado na prática, 50%, tanto para evitar comportamento oscilatório, como para sustentar o valor da concentração de n-hexano no "setpoint".

Os estudos em simulação também comprovaram as vantagens do uso das funções de pertinência lineares sobre as funções de pertinência não lineares, bem como demonstraram ser a base de regras de Mamdani (1975) a melhor opção para ser utilizada na prática, para a forma como os algoritmos de controle nebulosos foram implementados.

6. Resultados Experimentais

6.1 Introdução

Após a implementação e teste dos algoritmos de controle nebulosos em simulação, estes algoritmos foram usados em corridas experimentais realizadas na coluna piloto de destilação em batelada descrita no terceiro capítulo. A listagem do programa em linguagem FORTRAN 90 encontra-se no anexo D.

Os experimentos tiveram como objetivo destilar misturas binárias compostas de n-hexano e n-heptano, usando os controladores nebulosos a fim de determinar em condições experimentais o seu desempenho , avaliar a influência de alguns parâmetros de projetos dos controladores nebulosos, mantendo outros constantes (todos os testes experimentais foram feitos usando funções de pertinência triangulares igualmente espaçadas) , inclusive comparando-os com implementações digitais de controladores PI e PID (forma velocidade e posição).

Para determinar a concentração no topo e no fundo da coluna de destilação em batelada, foram utilizadas as seguintes equações de inferenciação, obtidas a partir de regressão não-linear, utilizando dados de equilíbrio líquido-vapor para o sistema n-hexano e n-heptano para a pressão de 1 atm:

$$x_{fundo} = 5{,}19986 - 8{,}12379 \cdot 10^{2} \cdot T_{fundo} + 2{,}89109 \cdot 10^{4} \cdot T_{fundo}^{2} \qquad \text{(eq. 6.1)}$$

$$y_{topo} = -1{,}10148 + 7{,}484458 \cdot 10^{2} \cdot T_{topo} - 6{,}45643 \cdot 10^{4} \cdot T_{topo}^{2} \qquad \text{(eq. 6.2)}$$

No gráfico 6.1 abaixo encontra-se o perfil de concentração para uma corrida experimental preliminar efetuada usando razão de refluxo constante (R=0,01). Como se pode observar, a concentração no topo de n-hexano cai continuamente, de acordo com o comportamento transitório esperado da destilação em batelada.

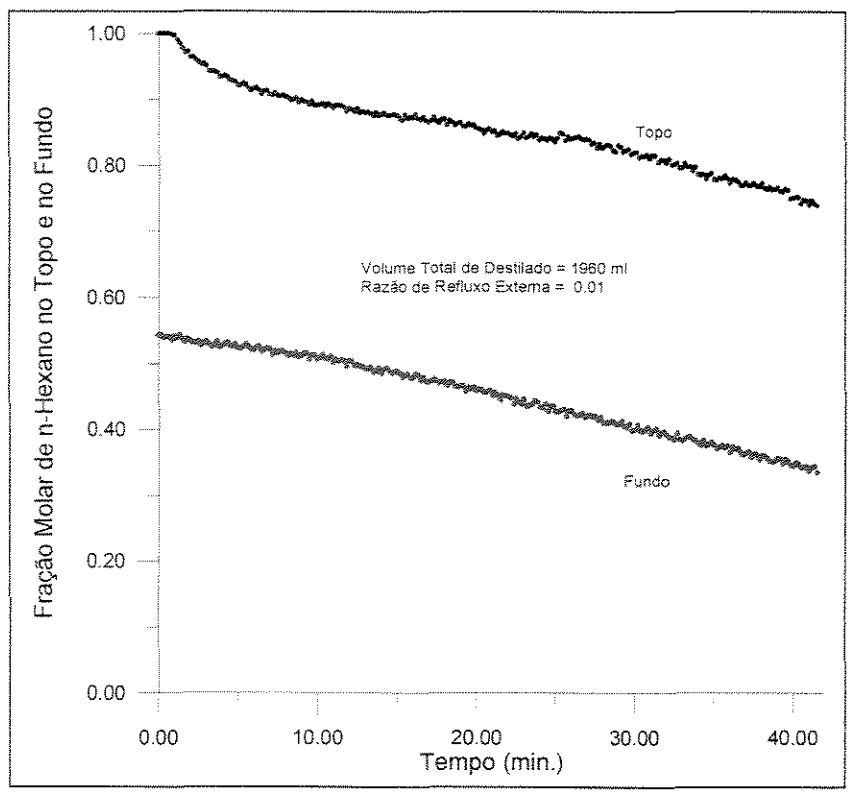


Gráfico 6.1: Perfil de Composição Para Operação à Refluxo Constante.

A operação utilizando uma razão de refluxo constante pequena prioriza a produção de grande volume de destilado, pois a quantidade de líquido que retorna à coluna é pequena, mas isso ocorre em detrimento da composição do destilado que varia ao longo do tempo.

A estratégia de operação básica utilizada durante a execução das corridas experimentais do presente trabalho foi manter a concentração em um dado valor de referência, exceto quando se optou por estudar a possibilidade de trabalhar com trajetórias de valores de referência durante o transcorrer da destilação.

6.2 Comparação de Desempenho entre Controladores Nebulosos e Controladores Tradicionais

6.2.1 Controladores PI Digital e PI Nebuloso

Os parâmetros de sintonia utilizados nos controladores proporcional integral tradicional, em sua versão digital, e no proporcional integral nebuloso estão na tabela 6.1 abaixo:

Parâmetro	Controlador PI	Controlador PI Nebuloso
Ganho Proporcional	250	-
Constante de Tempo Integral	0,5	-
Cardinalidade	-	15
Limite do Erro	-	0,005
Limite da Variação do Erro	-	0,01
Limite da Ação	-	2,5

Tabela 6.1: Parâmetros dos Controladores PI Digital e PI Nebuloso Testados.

O controlador proporcional integral tradicional, na forma velocidade, apresentou o comportamento esperado, ou seja, pequena oscilação ao redor do valor de referência e variações bastante bruscas na variável manipulada, a razão de refluxo. A partir da análise do gráfico 6.2, se pode inferir que o controlador PI tradicional seria uma ótima opção para o problema, mas cabe salientar que os valores de razão de refluxo que o controlador determina são muito altos, o que torna o tempo total de destilação muito grande.

Índice de Desempenho	Controlador PI Tradicional	Controlador PI Nebuloso
Concentração Média	0,913	0.907
Desvio Padrão	$2,42 \times 10^{-2}$	$2,14 \times 10^{-2}$
Erro Quadrático Médio	$7,61 \times 10^{-4}$	$5,08 \times 10^{-4}$
Razão de Refluxo Média	13,80	3,19
Vazão Média de Destilado	36,6 ml/min	42,8 ml/min

Tabela 6.2: Desempenho do Controlador PI Tradicional e do Controlador PI Nebuloso.

Além disso, as ações tomadas pelo controlador PI velocidade tem variações muito bruscas, o que pode trazer um maior desgaste à válvula de controle.

Conforme a tabela 6.2 , a vazão média de destilado é maior para o controlador nebuloso, o que determina para a mesma quantidade de destilado, um gasto inferior com energia de aquecimento.

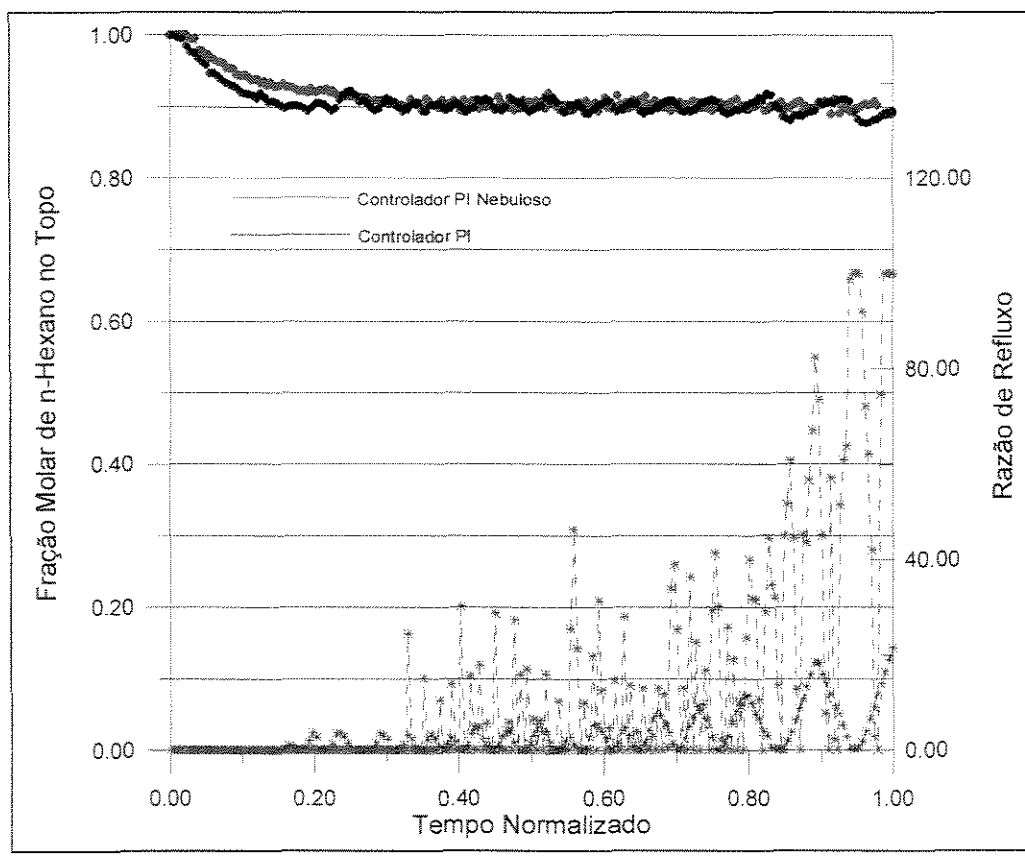


Gráfico 6.2: Perfis de Composição e Razão de Refluxo para Experimentos Usando Controlador PI Tradicional e Controlador PI Nebuloso.

Para facilitar a análise, diferentemente do capítulo anterior a variável tempo foi normalizada nos gráficos deste capítulo.

6.2.2 Controladores PID (Velocidade) e PID Nebuloso.

Os parâmetros de sintonia utilizados nos controladores proporcional integral derivativo, na sua forma velocidade, e no proporcional integral derivativo nebuloso encontram-se na tabela 6.3 abaixo:

Parâmetro	Controlador PID Velocidade	Controlador PID Nebuloso
Ganho Proporcional	250	-
Constante de Tempo Integral	0,5	-
Constante Derivativa	0,4	-
Cardinalidade	-	15
Limite do Erro	-	0,01
Limite da Variação do Erro	-	0,025
Limite da 2ª Variação do Erro	-	0,05
Limite da Ação	-	2,5

Tabela 6.3: Parâmetros dos Controladores PID Velocidade e PID Nebuloso.

Nota-se no gráfico 6.3 que o controlador PID velocidade manteve a concentração de n-hexano no valor de referência, mas isso ocorreu às custas de alterações bruscas na razão de refluxo. Além disso o referido controlador apresentou valores de razão de refluxo, em média, bastante superiores aqueles produzidos pelo controlador PID nebuloso.

Avaliando somente o perfil da variável controlada, o controlador PID velocidade apresentou os valores de desvio padrão e do erro quadrático médio inferiores aos do controlador PID nebuloso, devido à suas ações mais incisivas (tabela 6.4).

Índice de Desempenho	Controlador PID Velocidade	Controlador PID Nebuloso
Concentração Média	0,91	0,91
Desvio Padrão	$2,4 \times 10^{-2}$	$2,5 \times 10^{-2}$
Erro Quadrático Médio	$7,2 \times 10^{-4}$	$7,6 \times 10^{-4}$
Razão de Refluxo Média	26,33	3,47
Vazão Média de Destilado	34,9 ml/min	39,3 ml/min

Tabela 6.4: Desempenho do Controlador PID Velocidade e do Controlador PID Nebuloso.

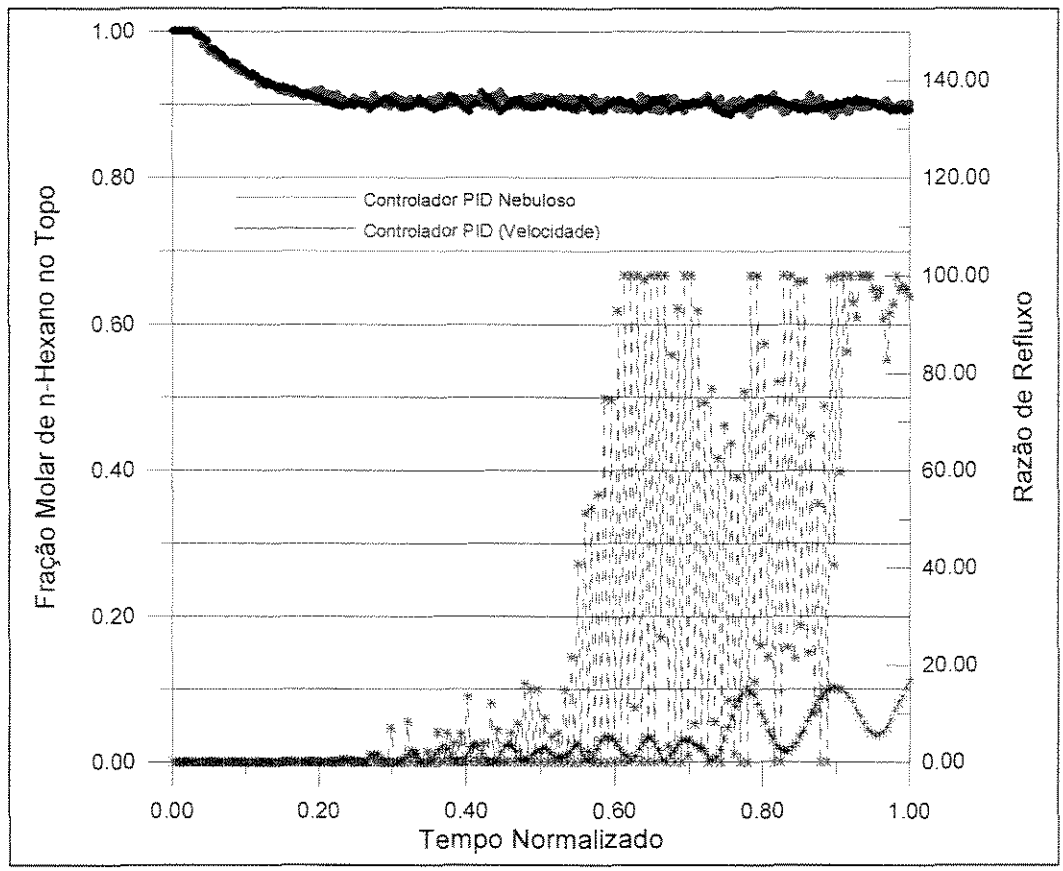


Gráfico 6.3: Perfis de Composição e Razão de Refluxo para Experimentos Usando Controlador PID Velocidade e Controlador PID Nebuloso.

6.2.3 Controladores PID (Posição) e PID Nebuloso.

Os parâmetros de sintonia utilizados nos controladores proporcional integral derivativo, na sua forma posição, e no proporcional integral derivativo nebuloso encontram-se na tabela 6.5 abaixo:

Parâmetro	Controlador PID Posição	Controlador PID Nebuloso
Ganho Proporcional	250	-
Constante de Tempo Integral	0,5	-
Constante Derivativa	0,4	-
Cardinalidade	-	15
Limite do Erro	-	0,01
Limite da Variação do Erro	-	0,025
Limite da 2ª Variação do Erro	-	0,05
Limite da Ação	-	2,5

Tabela 6.5: Parâmetros dos Controladores PID Posição e PID Nebuloso.

Através da análise do gráfico 6.4, verifica-se que o controlador PID posição apresenta um efeito de saturação bastante pronunciado, ocasionando oscilação na variável controlada. Isto sugere o uso do controlador PID, na forma velocidade, como uma opção melhor. O controlador PID nebuloso não apresentou a saturação, além disso, os valores de razão de refluxo especificados por esse controlador foram muito menos bruscos e menores (tabela 6.6).

Índice de Desempenho	Controlador PID Posição	Controlador PID Nebuloso
Concentração Média	0,91	0,91
Desvio Padrão	$2,77 \times 10^{-2}$	$2,5 \times 10^{-2}$
Erro Quadrático Médio	$8,66 \times 10^{-4}$	$7,82 \times 10^{-4}$
Razão de Refluxo Média	19,42	4,24
Vazão Média de Destilado	35,2	37,50

Tabela 6.6: Desempenho do Controlador PID Posição e do Controlador PID Nebuloso.

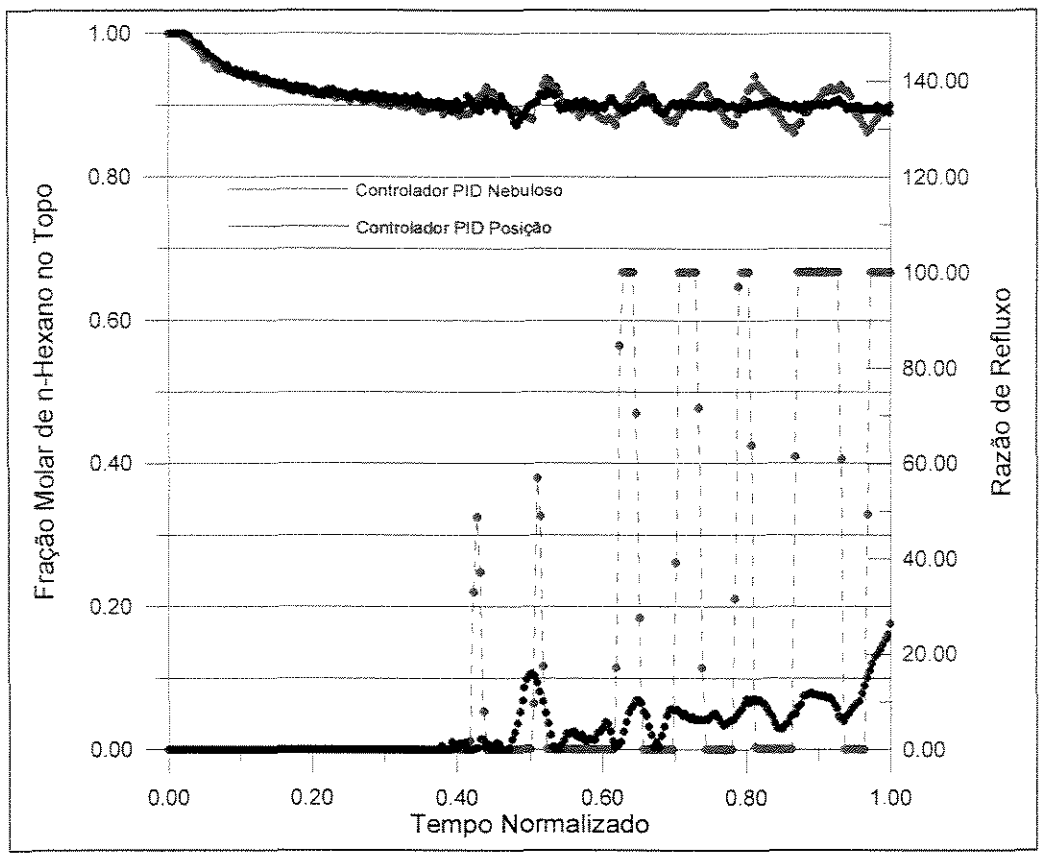


Gráfico 6.4: Perfis de Composição e Razão de Refluxo para Experimentos Usando Controlador PID Posição e Controlador PID Nebuloso.

6.3 Controladores PI Nebulosos

A seguir descrever-se-á o desempenho de controladores PI nebulosos, utilizados na destilação binária de misturas de n-hexano e n-heptano, avaliando a influência da cardinalidade. Foram testadas duas políticas para a especificação da pureza do destilado: a primeira usando um valor de referência fixo para a concentração de n-hexano no topo e a segunda usando uma trajetória de valores de referência, definindo três patamares de produção, de acordo com a concentração de n-hexano no fundo da coluna.

6.3.1 Operação com "Setpoint" Fixo

Conforme demonstrado nos estudos em simulação desenvolvidos no capítulo 5, a cardinalidade ,que é um parâmetro muito importante no projeto dos controladores nebulosos em geral, não guarda uma relação explícita com o desempenho, já que o erro quadrático em função da cardinalidade apresenta vários mínimos locais.

Parâmetro	Batelada 1	Batelada 2	Batelada 3
Cardinalidade	7	9	11
Limite do Erro	+/-0,005	+/-0,005	+/-0,005
Limite da Variação do Erro	+/-0,01	+/-0,01	+/-0,01
Limite da Ação	5,0	5,0	5,0
Valor de Referencia	0,80	0,80	0,80
Concentração Média	0,818	0,818	0,822
Desvio Padrão da Concentração	$4,3 \times 10^{-2}$	4,08E-2	$4,05 \times 10^{-2}$
Erro Quadrático Médio	$2,2 \times 10^{-3}$	$2,0 \times 10^{-3}$	$2,2 \times 10^{-3}$
Razão de Refluxo Média	2,97	3,81	3,96
Concentração Inicial de n-Hexano	0,4222	0,4255	0,4372
Tempo Batelada (min.)	30,66	27,50	28,16
Volume de Destilado (ml)	1100	1060	1090
Vazão Média de Destilado (ml/min)	35,87	38,54	38,71

Tabela 6.7: Comparação Entre Bateladas Usando Controladores de Cardinalidades Diferentes.

Na tabela 6.7 acima estão listados os parâmetros de projeto e são comparados os resultados de 3 bateladas utilizando controladores nebulosos PI de mesmo limite do erro e mesmo limite da variação do erro, mas com cardinalidades diferentes.

Observando-se os valores médios de concentração nota-se que todos os controladores mantiveram o valor da concentração de n-hexano no topo próximo ao valor de referência. Pelo

gráfico 6.5, nota-se que o perfil da variável manipulada, em todas as bateladas, assume valores pequenos em praticamente todo o transcorrer das três bateladas.

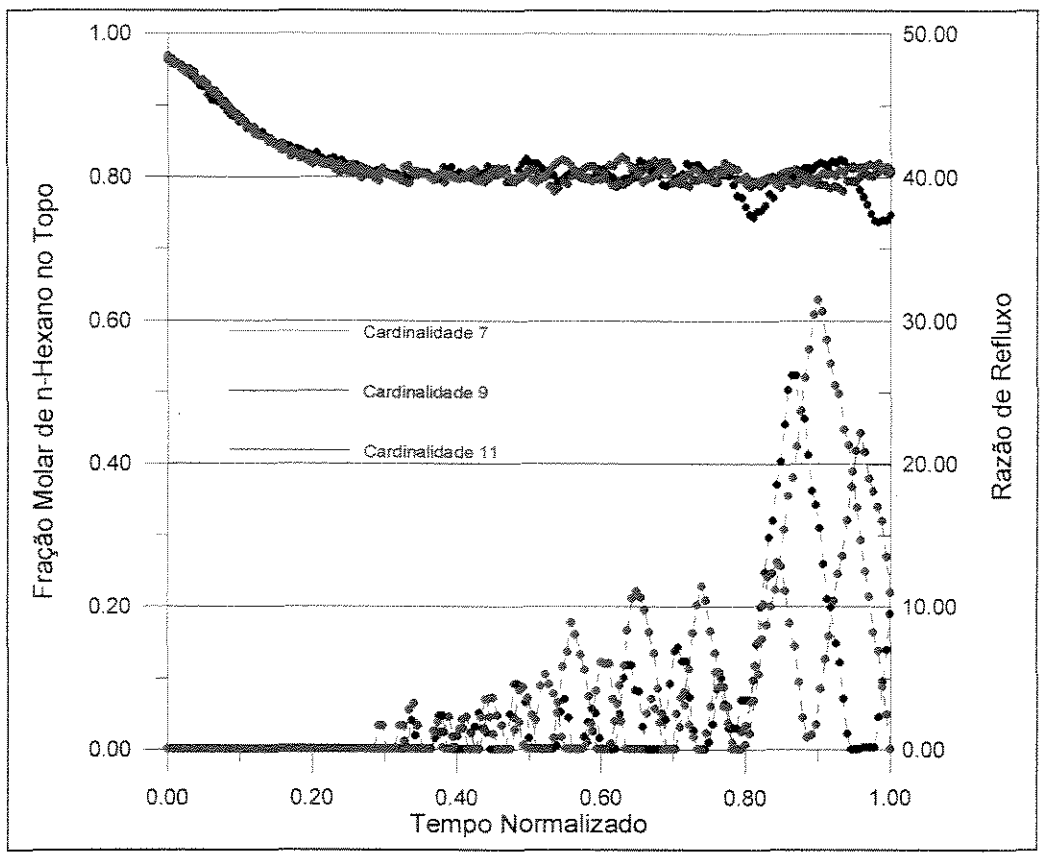


Gráfico 6.5: Controladores PI Nebulosos Com Cardinalidades Diferentes.

Parâmetro	Batelada 4	Batelada 5	Batelada 6
Cardinalidade	7	11	15
Limite do Erro	+/-0,005	+/-0,005	+/-0,005
Limite da Variação do Erro	+/-0,01	+/-0,01	+/-0,01
Limite da Ação	5,0	5,0	5,0
Valor de Referência	0,90	0,90	0,90
Concentração Média	0,908	0,913	0,912
Desvio Padrão da Concentração	$2,0 \times 10^{-3}$	$2,7 \times 10^{-3}$	$2,6 \times 10^{-2}$
Erro Quadrático Médio	$4,9 \times 10^{-4}$	$8,86 \times 10^{-4}$	$8,43 \times 10^{-4}$

(Continua...)

Razão de Refluxo Média	2,58	2,33	2,90
Concentração Inicial de n-Hexano	0,4855	0,4536	0,4645
Tempo Batelada (min.)	27,0	28,0	37,0
Volume de Destilado (ml)	920	740	910
Vazão Média (ml/min)	34,07	26,43	24,59

Tabela 6.8 Comparação entre Bateladas Usando Controladores de Cardinalidades Diferentes.

O comportamento descrito acima é corroborado por um outro conjunto de bateladas (tabela 6.8) , onde foi utilizado um "setpoint" diferente do usado nas bateladas da tabela 6.7.

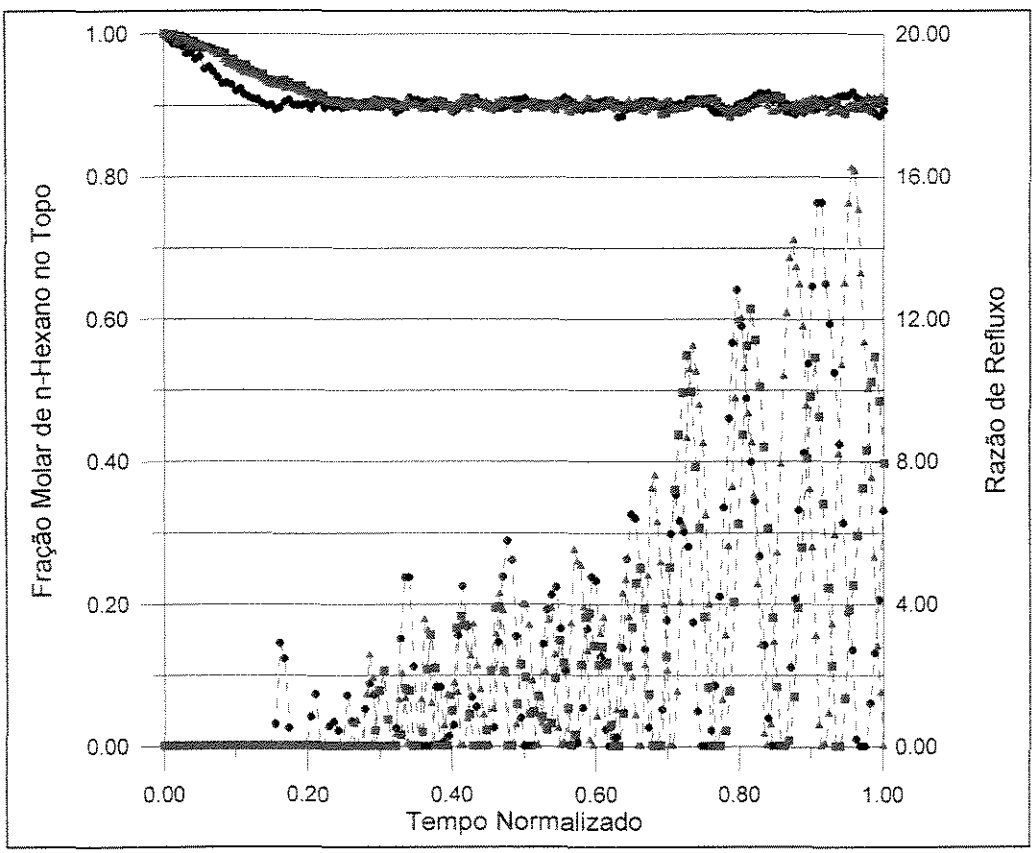


Gráfico 6.6: Perfis de Composição e Razão de Refluxo para 3 Bateladas Usando Controladores PI Nebulosos com Cardinalidades Diferentes

6.3.2 Operação com "Setpoint" Variável

Nas bateladas experimentais do item anterior utilizou-se um valor de referência para a concentração fixo. Uma outra alternativa é modificar o valor de referência ao longo da destilação,

conforme a mistura contida no refervedor da coluna fica mais pobre no componente mais volátil, obtendo-se assim vários "cortes" (destilados de composições diferentes).

Essa alternativa é muito utilizada na prática, pois ao permitir a obtenção de várias frações de destilado de purezas diferentes, dá maior flexibilidade a operação da coluna de destilação em batelada. Neste caso os diversos cortes podem compor produtos diferentes, ou alguns destes cortes podem ser destinados como produtos, enquanto os restantes são novamente destilados.

A mudança de valor de referência também é especialmente útil no caso da destilação em batelada multicomponente, quando o objetivo é separar os componentes intermediários da mistura.

A tabela 6.9 traz os dados de duas destilações bateladas, onde foram especificados 3 valores de "setpoint" para a concentração, da seguinte forma: enquanto a concentração de n-hexano no fundo da coluna fosse maior que 45% molar, o produto de topo deveria ter uma concentração de 90% molar em n-hexano; enquanto a concentração no fundo estivesse na faixa entre 45% e 35% molar em n-hexano, a concentração no topo deveria ser 80% molar em n-hexano; por fim, quando a concentração de n-hexano no fundo estivesse abaixo de 35% molar a concentração no produto de topo deveria ser 70% molar em n-hexano.

Em ambas as bateladas utilizou-se um controlador nebuloso de cardinalidade 17, usando o limite do erro na faixa de +/-0,005, o limite na variação do erro na faixa de +/-0,01 e funções de pertinência triangulares igualmente espaçadas, com grau de cruzamento de 50%.

Essas corridas experimentais também foram utilizadas para avaliar a influência do tipo de operação usada para o conectivo "e" da premissa das regras "se-então". Na batelada 1 foi usada a operação produto (o grau de pertinência da parte conseqüente da regra SE-ENTÃO é calculado como o produto entre os graus de pertinência da parte antecedente da regra SE-ENTÃO), enquanto na batelada 2 foi usada a operação de mínimo (o grau de pertinência da parte conseqüente é igual ao menor grau de pertinência na parte antecedente da regra).

Variável	Batelada 1	Batelada 2
Operação Entre os Conjuntos Nebulosos da Premissa	Produto	Mínimo
Limite do Erro	+/-0,005	0,005
Limite da Variação do Erro	+/-0,01	+/-0,01
Limite da Ação	5,0	5,0
Cardinalidade	17	17
Concentração média ("setpoint" 90%)	0,8987	0,9005
Desvio Padrão da Concentração ("setpoint" 90%)	$6,9495 \times 10^{-3}$	$6,752 \times 10^{-3}$
Concentração média ("setpoint" 80%)	0,8036	0,8027
Desvio Padrão da Concentração ("setpoint" 80%)	$7,7621 \times 10^{-3}$	$7,1159 \times 10^{-3}$
Concentração média ("setpoint" 70%)	0,6996	0,6977
Desvio Padrão da Concentração ("setpoint" 70%)	$1,3998 \times 10^{-2}$	$1,0421 \times 10^{-2}$
Razão de Refluxo Média ao Longo da Batelada	2,09	0,92

Tabela 6.9: Bateladas Usando Controladores PI Nebulosos Com Diferentes Operações para o Conectivo "e".

Comparando-se os resultados de concentração média e desvio padrão das duas bateladas, verifica-se que a batelada 2 apresenta valores de concentração mais próximos ao "setpoint" no primeiro e segundo patamares de produção, e menor desvio padrão em todos os patamares de produção.

Cabe salientar que a operação usada entre os conjuntos nebulosos da premissa é um dos fatores que determinam a não linearidade do controlador nebuloso. De acordo com a literatura (Jager, 1995) o uso da operação de mínimo torna o controlador não linear, diferentemente da operação produto.

O perfil da variável manipulada razão de refluxo na batelada 2, como pode-se notar no gráfico 6.7 , é melhor do que na batelada 1, pois são utilizados valores menores, o que como já discutido determina uma economia energética. Esse fato comprova a melhor opção é o uso da operação de mínimo como operação para o conectivo "e", não só neste caso de "setpoint" variável, mas também para os controladores operando a "setpoint" fixo.

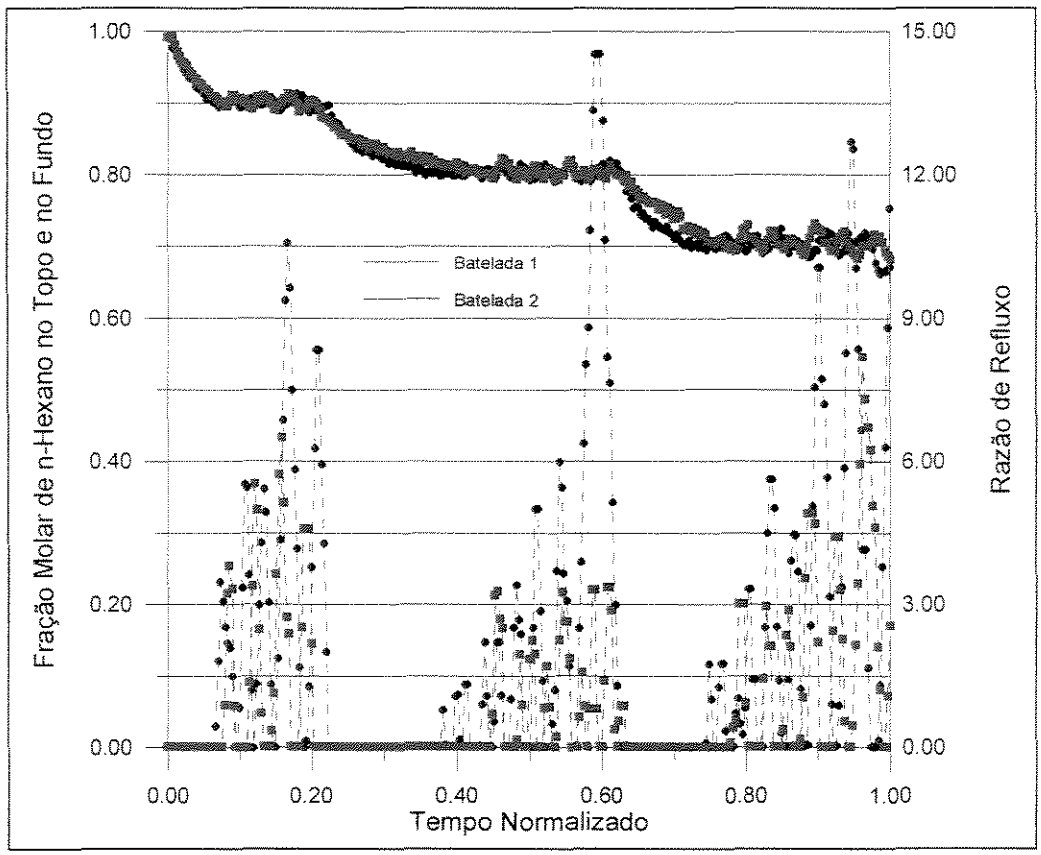


Gráfico 6.7: Perfis de Composição e Razão de Refluxo para Bateladas Com "Setpoints" Variáveis, Usando Controladores PI Nebulosos Com Operação "e" Diferentes.

6.4 Controladores PID Nebulosos

6.4.1 Operação em "Setpoint" Fixo

A avaliação do comportamento do controlador proporcional integral derivativo (PID) nebuloso, em bateladas experimentais, é importante por que o controlador PID tradicional é um dos mais amplamente utilizados na indústria.

Uma restrição muito importante vinculada a utilização do controlador PID nebuloso "on-line" é o fato da base de regras para esse controlador nebuloso crescer de forma geométrica,

conforme cardinalidades progressivamente maiores sejam utilizadas na definição dos conjuntos nebulosos das variáveis de entrada e saída do controlador (conforme discutido no capítulo 4).

Avaliar o controlador PID nebuloso somente a nível de simulação, não abrange essa restrição, que pode ser muito melhor estudada no controle "on-line", já que nesse caso o tempo de processamento do controlador deve ser pequeno o suficiente, para ocorrer dentro de um período de amostragem, e ainda permitir que a válvula magnética atue.

Os experimentos com esse tipo de controlador nebuloso demonstraram que o uso de funções de pertinência lineares abranda o problema do tempo de processamento, em especial na seção de defuzificação do controlador, pois o método do centro de área fica em muito simplificado mesmo para cardinalidades da ordem de 11.

A tabela 6.10 traz os dados para 3 destilações em batelada utilizando controladores PID nebulosos, que tiveram por objetivo, além de avaliar o desempenho do controlador PID nebuloso, determinar a influência da cardinalidade na resposta dinâmica do controlador.

Variável	Batelada 1	Batelada 2	Batelada 3
Operação de Inferência Nebulosa	Mínimo	Mínimo	Mínimo
Limite do Erro	0,01	0,01	0,01
Limite da Variação do Erro	0,025	0,025	0,025
Limite da Segunda Variação do Erro	0,05	0,05	0,05
Limite da Ação	5,0	5,0	5,0
Cardinalidade	3	5	7
Concentração média ("setpoint" 85%)	0,860	0,862	0,860
Desvio Padrão	$3,06 \times 10^{-2}$	$3,08 \times 10^{-2}$	$2,94 \times 10^{-2}$
Amplitude do Erro	0,0515	0,0481	0,0427
Erro Quadrático Médio	$1,056 \times 10^{-3}$	$1,085 \times 10^{-3}$	$0,976 \times 10^{-3}$
Razão de Refluxo Média	3,46	3,81	3,10

Tabela 6.10: Bateladas Usando Controladores PID Nebulosos com Cardinalidades Diferentes.

Analisando-se a tabela 6.10, verifica-se que conforme a cardinalidade foi aumentando, a amplitude do erro foi diminuindo, mas os valores médios de concentração são praticamente os mesmos, e o desvio padrão da concentração de n-hexano é aproximadamente também o mesmo nas três bateladas.

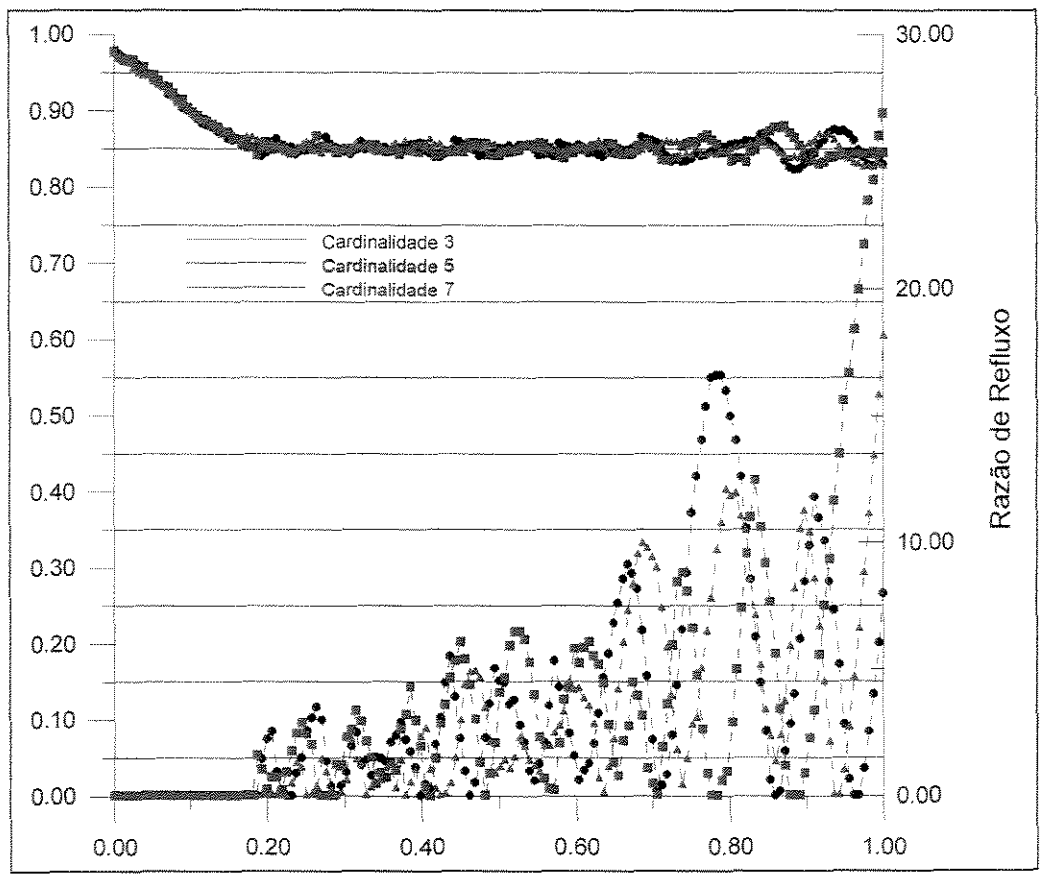


Gráfico 6.8: Perfis de Composição e Razão de Refluxo para Bateladas Usando Controlador PID Nebuloso com Cardinalidades Diferentes.

Nota-se também pelo gráfico 6.8, que a sintonia feita para estes controladores nebulosos teve melhor desempenho para a fase inicial da batelada. Isto se deve ao caráter transitório do processo em questão e pode ser eliminado através da adição de algum método de sintonia "on-line" (por exemplo o uso de algoritmos genéticos ou redes neurais).

6.4.2 Operação com "Setpoint" Variável

Para testar o funcionamento dos controladores proporcionais integrais derivativos nebulosos em condições de operação tais que o "setpoint" seja modificado ao longo da batelada, utilizou-se a seguinte programação para o "setpoint": enquanto a concentração no fundo da coluna era maior do que 45% molar em n-hexano, o setpoint era de 90%, quando a concentração no fundo ficou na faixa entre 45% e 35% o "setpoint" foi de 80% e, por fim quando a concentração no fundo caiu abaixo de 35% o setpoint foi de 70%. Os dados das três bateladas encontram-se na tabela 6.11 abaixo.

Variável	Batelada 1	Batelada 2	Batelada 3
Operação Entre os Conjuntos Nebulosos da Premissa	Mínimo	Mínimo	Mínimo
Limite do Erro	0,01	0,01	0,01
Limite da Variação do Erro	0,025	0,025	0,025
Limite da 2ª Variação do Erro	0,05	0,05	0,05
Limite da Ação	5,0	5,0	5,0
Cardinalidade	7	11	15
Concentração média ("setpoint" 90%)	0,901	0,900	0,900
Desvio Padrão da Concentração ("setpoint" 90%)	$4,9 \times 10^{-3}$	$4,9 \times 10^{-3}$	$4,6 \times 10^{-3}$
Concentração média ("setpoint" 80%)	0,801	0,800	0,800
Desvio Padrão da Concentração ("setpoint" 80%)	$4,5 \times 10^{-3}$	$5,6 \times 10^{-3}$	$6,4 \times 10^{-3}$
Concentração média ("setpoint" 70%)	0,700	0,698	0,701
Desvio Padrão da Concentração ("setpoint" 70%)	$7,8 \times 10^{-3}$	$6,7 \times 10^{-3}$	$8,5 \times 10^{-3}$
Tempo Médio de Processamento do Controlador	-	2,21 s	2,24s

Tabela 6.11: Bateladas Usando Controladores PID Nebulosos Com "Setpoint" Variável .

Da análise dos valores de concentração média e desvio padrão em cada um dos patamares de produção, para as três bateladas, deduz-se que a cardinalidade teve pouca influência no desempenho do controlador, já que os valores de concentração e desvio padrão são aproximadamente os mesmos em cada caso.

Quanto aos valores assumidos pela razão de refluxo, em todas as bateladas o perfil desta variável manipulada foi muito bom (vide gráfico 6.8 abaixo) , assumindo sempre valores menores que 10. Isso pode ser atribuído ao amplo limite do erro utilizado na sintonia do controlador, já que o erro podia variar entre +0,01 e –0,01.

Para elucidar a influência do aumento da base de regras no tempo computacional do algoritmo de controle "on-line", em duas das bateladas acima (batelada 2 e batelada 3, com 11^3 e 15^3 regras, respectivamente) o tempo médio de processamento do algoritmo de controle foi aproximadamente o mesmo. Isso comprovou o fato de que o uso de funções de pertinência lineares agiliza o processamento do controlador nebuloso, além de já se alcançar com o seu uso um desempenho muito bom.

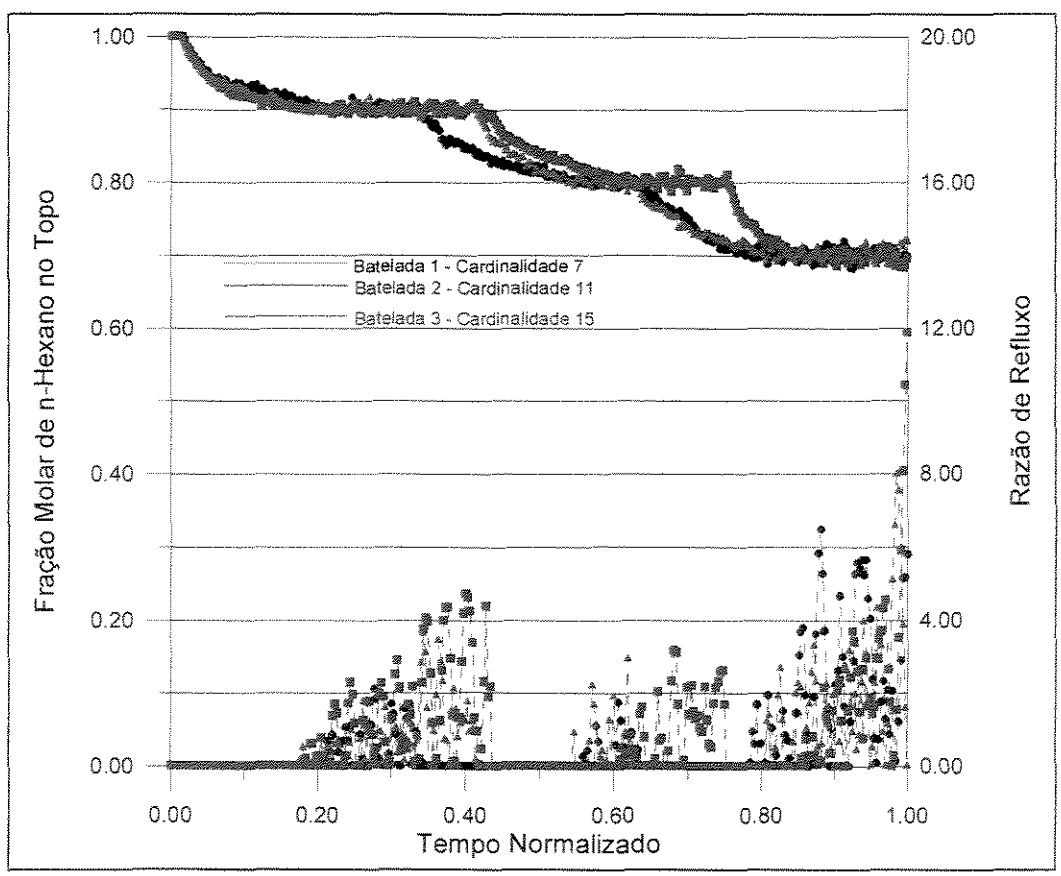


Gráfico 6.9: Perfis de Composição e Razão de Refluxo em Bateladas Usando Controlador PID Nebuloso com "Setpoint" Variável.

Nas três corridas é possível notar a facilidade com que o controlador absorve a mudança de "setpoint", não havendo prejuízo do desempenho do mesmo em nenhum momento.

6.5 Controladores Proporcionais Nebulosos

6.5.1 Operação com "Setpoint" Fixo

O controlador proporcional nebuloso é o controlador mais simples de ser implementado e o de pior desempenho na destilação em batelada, pois apresenta um comportamento "liga-desliga" bastante pronunciado.

A vantagem mais evidente do controlador proporcional nebuloso sobre o controlador proporcional tradicional é o fato de que devido a forma de implementação, as ações implementadas pelo controlador proporcional nebuloso são restritas a uma determinada faixa, o que não ocorre no controlador proporcional tradicional.

Foram realizadas duas bateladas para avaliar o desempenho do controlador nebuloso proporcional; os dados para fins de comparação estão na tabela 6.12 e o gráfico 6.10 correspondente encontra-se abaixo.

O parâmetro de projeto dos controladores que foi diferente entre as duas bateladas foi o limite do erro, mantidos todos os outros parâmetros iguais.

Verifica-se que o controlador proporcional nebuloso de limite do erro mais estreito apresenta uma exatidão maior, o que fica evidenciado pelo menor desvio padrão. Cabe salientar que na batelada 1 o perfil da variável controlada demonstra a presença de um "off– set", denotado pela maneira decrescente ao longo da batelada (o que ressalta o baixo desempenho deste controlador). A utilização de um limite do erro menos rígido fica mais clara ainda pelo comportamento da variável manipulada, que em momento nenhum apresentou o efeito "liga-desliga" comum em controladores proporcionais de ganho alto.

Parâmetro	Batelada 1	Batelada 2
Limite do Erro	0,010	0,005
Limite da Ação	10,01	10,01
Concentração Média	0,828	0,825
Desvio Padrão	$4,56 \times 10^{-2}$	$4,2 \times 10^{-2}$
Erro Quadrático Médio	$2,84 \times 10^{-3}$	$2,39 \times 10^{-3}$
Razão de Refluxo Média	2,56	2,59
Concentração Inicial de n-Hexano	0,4418	0,4350
Tempo Final da Batelada (min.)	33,50	35,33
Volume Total de Destilado (ml)	1260	1220
Vazão Média de Destilado (ml/min.)	37,61	34,53

Tabela 6.12: Dados de Bateladas Usando Controladores P Nebulosos (Cardinalidade=15).

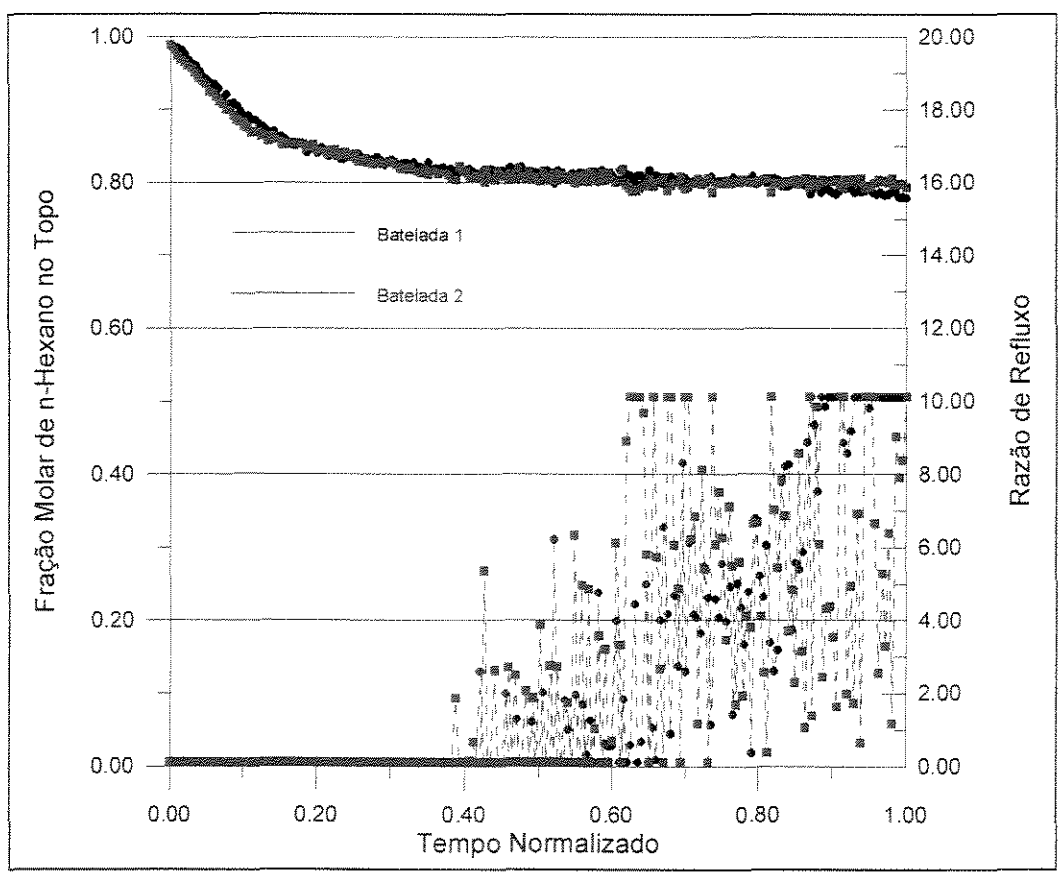


Gráfico 6.10: Composição e Razão de Refluxo em Bateladas Usando Controlador Proporcional Nebuloso com "Setpoint" Fixo.

6.5.2. Operação com "Setpoint" Variável

Na tabela 6.13, abaixo, são apresentados os dados de duas bateladas utilizando um controlador proporcional nebuloso (com limites do erro iguais a +/-0.0075 e limites da variável manipulada, R, de 0,01 a 10,01, em todas as 2 bateladas) onde são especificados três concentrações diferentes de produto (90%, 80% e 70% molar de n-hexano), de acordo com a concentração de n-hexano no fundo da coluna .

Parâmetro	Batelada 1	Batelada 2
Cardinalidade	7	15
Faixa do Limite do Erro	0,0075	0,0075
Limite da Ação	10,01	10,01
Concentração Média (90%)	0,894	0,897
Desvio Padrão	$4,3 \times 10^{-3}$	$3,8 \times 10^{-3}$
Concentração Média (80%)	0,799	0,804
Desvio Padrão	$6,0 \times 10^{-3}$	$7,7 \times 10^{-3}$
Concentração. Média (70%)	0,698	0,699
Desvio Padrão	$5,7 \times 10^{-3}$	$8,6 \times 10^{-3}$
Concentração. Inicial de n-Hexano	0,4302	0,4278
Tempo Batelada (min.)	43,50	41,83
Volume de Destilado (ml)	1580	1520
Vazão Média de Destilado (ml/min)	36,32	36,33

Tabela 6.13: Comparação Entre Bateladas Utilizando Cardinalidades Diferentes.

Quanto ao comportamento da variável manipulada, razão de refluxo, é possível afirmar que o mesmo não difere muito entre as duas bateladas, tanto pela análise do gráfico , quanto através da comparação das vazões médias de destilado ao longo da batelada.

Como foi utilizado um controlador proporcional nebuloso, o comportamento da variável manipulada é oscilatório, como já esperado, o que, dependendo do valor do ganho utilizado no

controlador pode torná-lo com características "liga-desliga". O algoritmo de controle nebuloso determina os limites inferior e superior para a variável manipulada, o que tornou possível ajustar o mesmo para varrer a faixa de valores de razão de refluxo de 0.01 a 10.01, o que maximiza a produção de destilado, apesar do tipo de controlador utilizado.

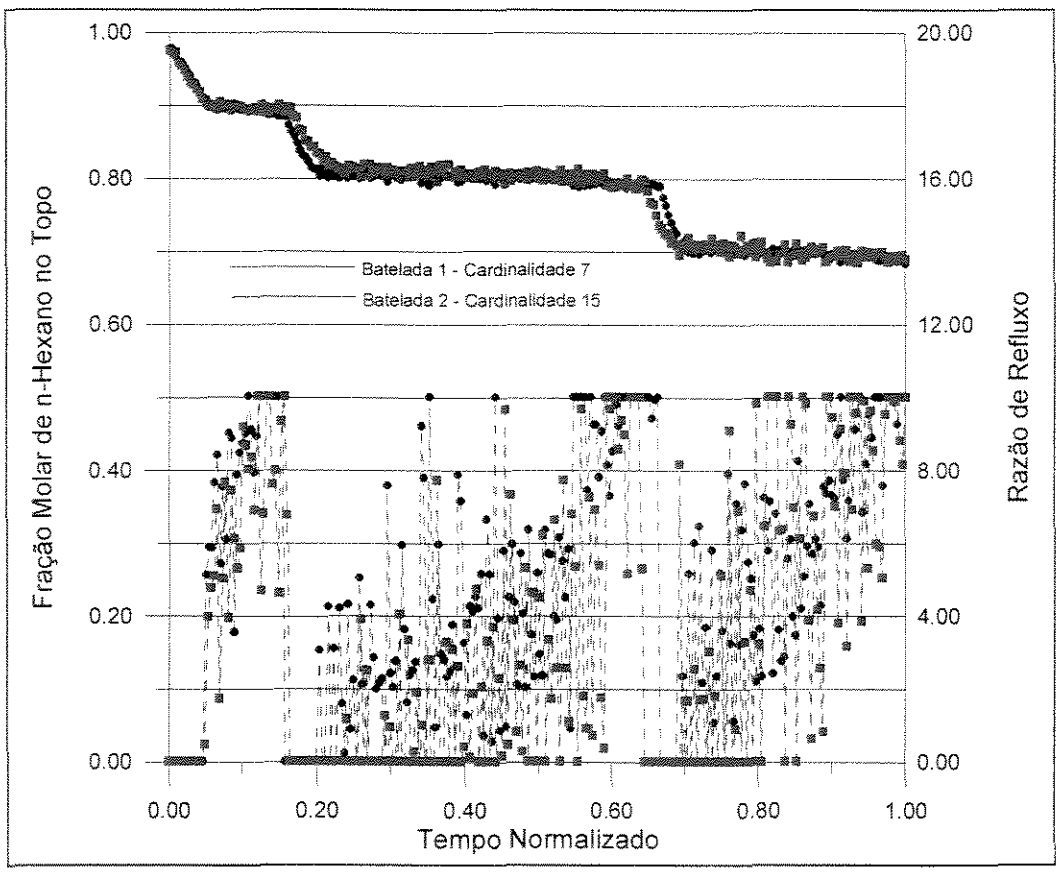


Gráfico 6.11: Perfis de Composição e Razão de Refluxo em Bateladas Usando Controlador Proporcional Nebuloso com "Setpoint" Variável.

6.6 Conclusões

Durante o trabalho experimental foram destiladas misturas binárias de n-hexano e n-heptano, testando os controladores nebulosos proporcional, proporcional integral e proporcional integral derivativo. Tanto o controlador PI nebuloso quanto o PID nebuloso alcançaram resultados muito bons, tanto em termos do comportamento da variável controlada (manutenção do valor de referência), como em termos da variável manipulada, razão de refluxo, onde se

alcançaram perfis mais suaves sem ações bruscas, com valores em média menores do que aqueles obtidos por outras espécies de controladores (tradicionais ou não).

O controle nebuloso mostrou-se capaz de operar sob condições de "setpoint" variável, facilitando a retirada de vários cortes, apesar dos seus parâmetros de projeto permanecerem constantes ao longo do transcorrer da batelada.

Os controladores nebulosos mostraram-se como uma alternativa válida para a automação de colunas de destilação em batelada, em especial pela facilidade com que é possível implementar o conhecimento de um especialista humano no controle do processo em questão. Isso permite um grande grau de especificidade do controlador projetado ao problema que se deseja resolver.

7. Conclusões e Sugestões

A partir do estudo da aplicação da lógica nebulosa no controle de processos, em especial no controle de processos químicos, foram desenvolvidos algoritmos de controle nebulosos, com o intuito de explorar e desenvolver o controle digital "on-line" da destilação em batelada de misturas binárias.

A grande limitação dos controladores tradicionais, quando usados em processos de natureza transiente, está vinculada ao fato dos mesmos usarem parâmetros fixos, ajustados de acordo com a média do comportamento ao longo da duração do processo. Uma alternativa, já utilizada, é o uso de controladores adaptativos, tais como o PAC (Controlador Adaptativo Programável) ou o STR (Controlador Auto Ajustável).

A abordagem neste trabalho foi explorar os controladores nebulosos, que a literatura menciona como mais robustos que os controladores tradicionais, como mais uma alternativa para o controle digital "on-line".

Inicialmente a partir do trabalho desenvolvido por Fileti (1992) foi usado um simulador de destilação em batelada multicomponente, para testar os algoritmos de controle nebuloso desenvolvidos, antes da implementação dos mesmos no equipamento disponível em laboratório.

A etapa de simulação permitiu avaliar os parâmetros de projeto dos controladores nebulosos desenvolvidos, tais como os parâmetros de escala, cardinalidade, tipos de funções de pertinência, regras usadas no controlador, a fim de permitir um melhor domínio das influências e interações destes parâmetros com o desempenho dos controladores nebulosos.

Como já discutido nos capítulos 4 e 5, desde as primeiras simulações, ficou nítida uma dicotomia no que se refere a sintonia dos controladores nebulosos: de um lado a possibilidade de desenvolver um algoritmo de controle específico para um dado processo, com uma grande flexibilidade na estrutura do controlador, facilitando a inserção do conhecimento de um

especialista humano (especialmente pelo fato do controlador nebuloso ser baseado em regras e manipular conceitos mal definidos), de outro lado a dificuldade em ajustar os parâmetros de sintonia do controlador nebuloso.

Como ponto de partida para o desenvolvimento dos algoritmos de controle, optou-se por usar os controladores nebulosos análogos aos controladores P, PI e PID tradicionais, pelo fato destes serem amplamente discutidos na literatura, manipularem variáveis facilmente conhecidas (erro, variação do erro e segunda variação do erro) e pela disponibilidade da base de regras definida por Mamdani (1975).

Os resultados em simulação demonstraram que os parâmetros de escala têm uma grande influência na resposta dinâmica do controlador, no que tange a presença ou não de comportamento oscilatório, manutenção do valor de referência, etc. O uso de limites do erro muito estreitos favorecem a presença do comportamento oscilatório, enquanto valores muito amplos determinam valores de erro quadrático muito grandes, pois não é mantido o "setpoint". Para o limite da variação do erro, o uso de valores muito largos ou muito estreitos permitem a presença de oscilações.

A análise do comportamento da coluna de destilação em batelada, em simulação, em função da cardinalidade dos controladores nebulosos testados, comprovou a dificuldade em determinar qual a cardinalidade que deveria ser utilizada nas corridas experimentais; ao mesmo tempo as simulações determinaram qual o grau de cruzamento ("crossover") a ser utilizado na prática, 50%, tanto para evitar comportamento oscilatório, como para sustentar o valor da concentração de n-hexano no "setpoint".

Os estudos em simulação também comprovaram as vantagens do uso das funções de pertinência lineares sobre as funções de pertinência não lineares, bem como demonstraram ser a base de regras de Mamdani (1975) a melhor opção para ser utilizada na prática, para a forma como os algoritmos de controle nebulosos foram implementados.

Durante o trabalho experimental foram destiladas misturas binárias de n-hexano e n-heptano, testando os controladores nebulosos proporcional, proporcional integral e proporcional integral derivativo. Tanto o controlador PI nebuloso quanto o PID nebuloso alcançaram resultados muito bons, tanto em termos do comportamento da variável controlada (manutenção do valor de referência), como em termos da variável manipulada, razão de refluxo, onde se alcançaram perfis mais suaves sem ações bruscas, com valores em média menores do que aqueles obtidos por outras espécies de controladores (tradicionais ou não).

O controle nebuloso mostrou-se capaz de operar sob condições de "setpoint" variável, facilitando a retirada de vários cortes, apesar dos seus parâmetros de projeto permanecerem constantes ao longo do transcorrer da batelada.

A grande dificuldade observada na implementação dos controladores nebulosos é a sintonia, e para resolver esse problema são sugeridas algumas alternativas, tais como:

- utilizar os chamados controladores nebulosos baseados em modelo. Isso poderia ser feito usando um modelo discretizado do sistema (uma função de primeira ordem com atraso de tempo), de forma que os parâmetros desta fossem identificados "on-line" durante o transcorrer da destilação e servissem como base para o ajuste dos fatores de escala do controlador nebuloso;

- embutir no algoritmo de controle nebuloso uma rede neural artificial capaz de reconhecer os padrões de comportamento dinâmico da coluna de destilação em batelada, o que permitiria a modificação dos parâmetros do controlador nebuloso ao longo da batelada, implantando assim um controle nebuloso preditivo baseado em redes neurais;

- agregar métodos de otimização numérica tradicionais ou não (algoritmos genéticos) ao controlador nebuloso. Isso poderia ser feito novamente usando um modelo discretizado que serviria como base para a otimização de determinada função objetivo (por exemplo custo).

Para finalizar, este trabalho acrescenta-se aos poucos trabalhos relatados na literatura de aplicação da lógica nebulosa na engenharia química. Confirmou-se por simulações e

experimentalmente que essa ferramenta mostra-se bastante promissora na área de controle de processos químicos transientes e não lineares. Especialmente em processos de difícil modelagem fenomenológica e/ou onde há desconhecimento das propriedades físico-químicas dos fluidos processados. Nestes casos, os controladores nebulosos apresentam-se como excelente alternativa à modelagem empírica tão utilizada industrialmente no projeto de controladores.

8. Bibliografia

BAROLO, M., GUARISE, G. B., Batch Distillation of Multicomponent Systems With Constant Relative Volatilities. Trans I ChemE, vol. 74, Part A, p. 863-871, 1996.

BAUGHMAN, D. R., LIU, Y. A., Neural Networks in Bioprocessing and Chemical Engineering. 1st Ed. , Academic Press Inc., 487 p. 1995.

BIASIZZO, K. K., SKRJANC, I., MATKO, D., Fuzzy Predictive Control of Higly Nonlinear pH Process. Computers Chem. Engng., Vol 21, Suppl., pp. S613-S618, 1997.

CONVERSE, A. O., GROSS, G.D., Optimal Distillate – Rate Policy in Batch Distillation. Ind. Engng. Chem. Fundam., V. 2, p. 217-221, 1963.

COWARD, I., The Time Optimal Problem in Binary Batch Distillation. Chem. Engng. Sci., V. 22, pp. 503-516, 1967.

CUNHA, A. P., Automação e Controle "On-line" de uma Coluna de Destilação em Batelada. Campinas: Faculdade de Engenharia Química, Universidade Estadual de Campinas, 1996, 115p. Tese (Mestrado).

DIWEKAR, U. M., MALIK, R.K., MADHAVAN, K. P., Optimal Reflux Rate Policy Determination for Multicomponent Batch Distillation Columns. Computers. Chem. Engng., V. 11, n. 6, pp. 626-637, 1987.

DIWEKAR, U. M., MADHAVAN, K. P., Batch-Dist: A Comprehensive Pakcage for Simulation, Design, Optimization and Optimal Control os Multicomponente, Multifraction Batch Distillation Columns. Computers Chem. Engng., Vol. 15, 12, pp. 833-842, 1991.

DOMENECH, S., ENJALBERT, M. Program for Simulating Batch Rectification as a Unit Operation. Computers and Chemical Engineering, v. 5, n. 3, pp. 181-184, 1981.

DRIANKOV, D., HELLENDOORN, H., REINFRANK, M., An Introduction to Fuzzy Control. 2. Ed. , Springer-Verlag, New York, 316 p, 1996.

FARHAT, S., PIPOULEAU, L., DOMENECH, S., CZERNICK, M., Optimal Control of Batch Distillation via Nonlinear Programming. Chem. Eng. Process., 29, pp. 33-38, 1991.

FILETI, A. M. F. , Controle Adaptativo e Preditivo com Redes Neurais de uma Coluna de Destilação em batelada. Campinas: Faculdade de Engenharia Química, Universidade Estadual de Campinas, 1996, 201 p. Tese (Doutorado).

FILETI, A. M. F., Estratégias de Controle em Destilação Batelada. Campinas: Faculdade de Engenharia Química, Universidade Estadual de Campinas, 1992, 115p. Tese (Mestrado).

GALINDEZ, H., FREDENSLUND, A., Simulation of Multicomponent Batch Distillation Processes. Computers Chem. Engng, Vol. 12, 4, pp. 281-288, 1988.

GALICHET, S., FOULLOY, L. CHEBRE, M., BEAUCHENE, J. P., Fuzzy Logic Control of a Floating Level in a Refinery Tank. Proc. of 3rd IEEE Int. Conf. on Fuzzy Systems, Orlando, US, Jun. 1994, pp. 1538-1542.

GODJEVAC, J. Comparison Between PID and Fuzzy Control. Internal Report R95.361. Ecole Polytechnique Féderale, Lausanne, 1995.

GOMIDE, F. A. C., GUDWIN, R. R., Modelagem, Controle, Sistemas e Lógica Fuzzy. 1° Simpósio Brasil-Japão de Lógica Nebulosa. UNICAMP, Campinas, 1992.

GOMIDE, F. A. C., PEDRICZ, W., An Introduction to Fuzzy Sets – Analysis and Design. 1st Ed., MIT Press, 465 p., 1998.

JAGER, R., Fuzzy Logic in Control. Thesis Technische, Universiteit Delft, 1995, p. 234.

KARR, C., GENTRY, E., Fuzzy Control of pH Using Genetic Algorithms. IEEE Transactions on Fuzzy Systems, v. 1, 1, Feb. 1993, pp 46-53.

KETONEN, M., Fuzzy Logic Application in Phosphoric Acid Production. Control Engineering Practice, v. 1, 2, Apr. 1993, pp. 273-281.

LI., X., RUAN, D., The Current Development of FLC Project at BR1 and New Algorithms Design. Internal Report, SCK.CEN, Belgium, Nov., 1997.

LU, Y. Z., HE, M., WEI, C., Fuzzy Modeling and Expert Optimization Control for Industrial Processes. IEEE Transactions on Control Systems Technology, v. 5, 1, January, 1997.

LUYBEN, W. L., Some Practical Aspects of Optimal Batch Distillation Design. Ind. Eng. Chem. Prac. Des. Dev., v. 10, 1, pp. 54-59, 1971.

LUYBEN, W. L., Multicomponent Batch Distillation. 1. Ternary Systems with Slop Recycle. Ind. Eng. Chem. Res., v. 27, pp. 642-647, 1988.

MACCHIETTO, S., BAROLO, M. GUARISE, G. B., RIBON, N., RIENZI, S., TROTTA, A., Some Issues in the Design and Operation of a Batch Distillation Column with a Middle Vessel. Computers Chem. Engng., v. 20, suppl, pp. S37-S42, 1996.

McCULLOUGH, M. K., Fuzzy Logic Control of Resin Curing. in: Fuzzy Logic and Control: Software and Hardware Applications. Ed. M. Jamshidi, N. Vadiee, T. J. Ross, v.2, Prentice-Hall, 1993.

MAMDANI, E. H., Applications of Fuzzy Algorithm for Simple Dynamic Plant. Proceedings IEEE, 121(12), pp. 1585-1588, 1974.

MAMDANI, E. H., ASSILIAN, S., An Experiment in Linguistic Synthesis With a Fuzzy Logic Controller. International Journal of Man-Machine Studies, 7, pp. 1-13, 1975.

MARMOL, E. Q., LUYBEN, W.L., Multicomponente Batch Distillation. 2. Comparison of Alternative Slop Handling and Operating Strategies. Ind. Eng. Chem. Res., 29, pp. 1915-1921, 1990.

MAHESH, M.M., MADHAVAN, K. P., Fuzzy Control of a Semi-Batch Copolymerization Reaction. Chemical Engineering and Processing, 32, pp. 327-331, 1993.

MESKI, O. A., MORARI, M., Design and Opeation of a Batch Distillation Column with a Middle Vessel, Computers Chem. Engng., v. 19, sup., pp. S597-S602, 1995.

PONTON, J. W., KLEMES, J., Alternatives to Neural Networks for Inferential Measurement. Computers Chem. Engng., v. 17, 10, pp. 991-1000, 1993.

OISOVICI, R. M., Operação e Controle por Computador "On-line" de uma Coluna de Destilação em Batelada. Campinas: Faculdade de Engenharia Química, Universidade Estadual de Campinas, 1998, 140p. Tese (Mestrado) .

PENG, X., LIU, S., YAMAKAWA, T., WANG, P., LIU, X., Self-Regulating PID Controllers and ITS Applications to a Temperature Controlling Process. Fuzzy Computing, Elsevier Science Publishers, North-Holland, 1988.

PEDRYCZ, W., Fuzzy Control and Fuzzy Systems. 2nd Ed., John Wiley and Sons, 1993.

PEDROSA, L. S., Controle Auto-Ajustável de uma Coluna Piloto de destilação em Batelada com Inferenciação de Composição Através de Redes Neurais Artificiais. Campinas: Faculdade de Engenharia Química, Universidade Estadual de Campinas, 1998. Tese (Mestrado).

PROCYK, T. J., MAMDANI, E. H., A Linguistic Self-Organising Process Controller. Automatica, 15, pp. 15-30, 1979.

ROBINSON, E.R., The Optimal Control of na Industrial Batch Distillation Column. Chem Eng. Sci., 25, pp. 921-928, 1970.

SORENSEN, E., SKOGESTAD, S., Comparison of Regular and Inverted Batch Distillation. Chem. Engng. Sci., v. 51, 22, pp. 4949-4962, 1996.

SHINSKEY, F. G., Process Control Systems – Application Design and Tuning. New York, McGraw-Hill, 1988.

STENZ, R., KUHN, U., Vergleich: Fuzzy-Automatisierung und Konventionelle Automatisierung einer Batch-Destillationskolonne. Automatisierungstechnische Praxis - atp, v. 35, 5, pp. 288-295, 1993.

TAKAGI, T., SUGENO, M. , Derivation of Fuzzy Control Rules from Human Operator's Control Actions. Sanchez and Gupta, pp. 55-60, 1983.

TANAKA, K., SANO, M., WATANABE, K., Modeling and Control of Carbon Monoxide Concentration Using a Neuro-Fuzzy Techinique. IEEE Transactions on Fuzzy Systems,v. 3, 3, Aug., 1995.

ROFFEL, B., CHIN, P.A., Application of Fuzzy Logic in the Control of Polymerization Reactors. Control Engineering Practice, v. 1, 2, pp. 267-272, Apr. 1993.

WILSON, J. A., MARTINEZ, E.C., Neurofuzzy Modeling and Control of a Batch Process Involving Simultaneous Reaction and Distillation. Computers Chem. Engng, v. 21, Suppl., pp. S1233-S1238, 1997.

YAGER, R. R., FILEV, D. P., Essentials of Fuzzy Modeling and Control. 1st Ed., John Wiley & Sons Inc., 388 p., 1994.

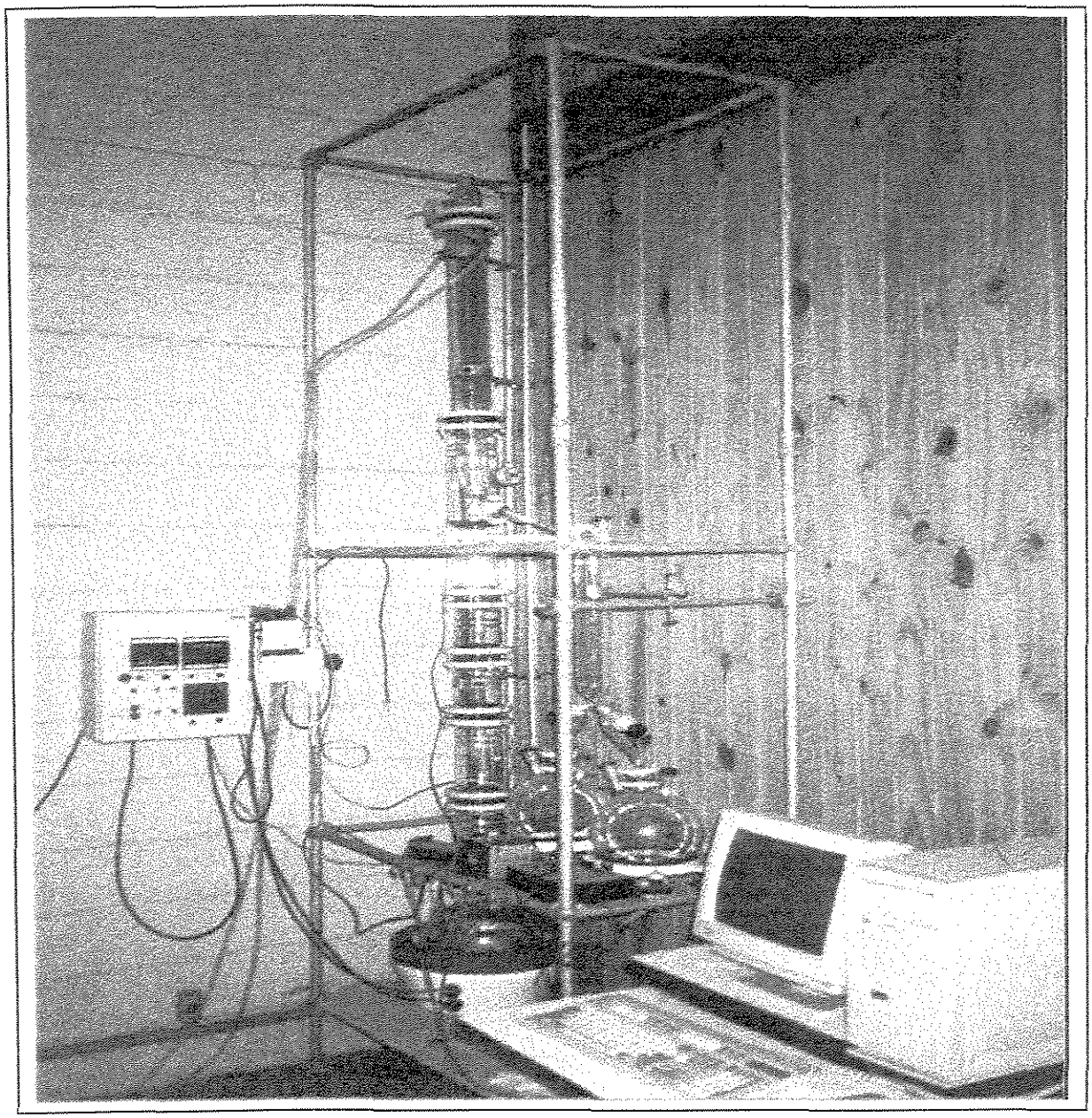
YAGISHITA, O., ITOH, O. , SUGENO, M., Application of Fuzzy Reasoning to the Water Purification Process. Industrial Applications of Fuzzy Control. M. Sugeno (ed.), Elsevier Science Publishers, North-Holland, 1985.

ZADEH, L. A., Fuzzy Sets. Information and Control, 8, pp 338-353, 1965.

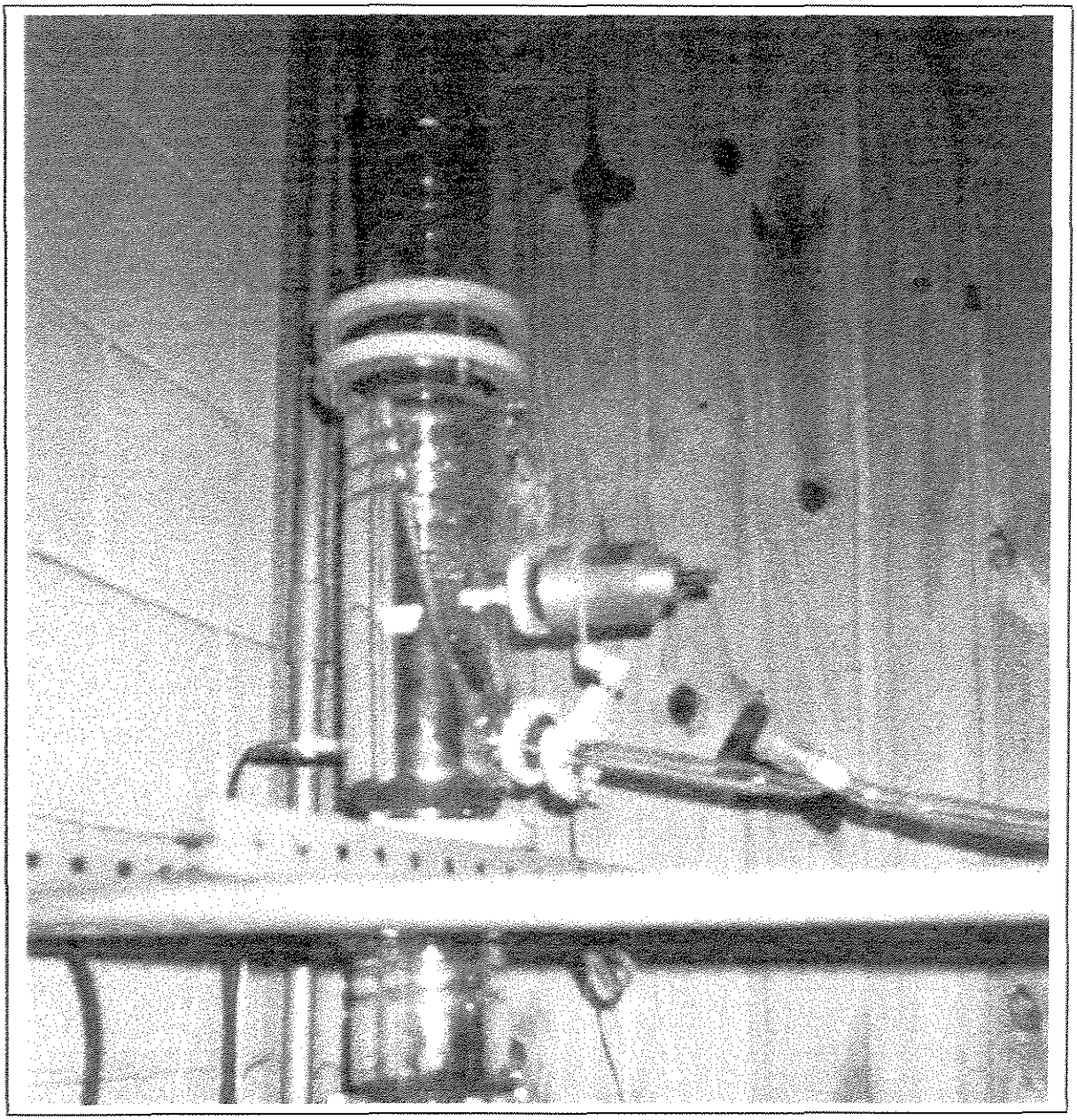
ZADEH, L. A., Outline of a New Approach to the Analysis of Complex Systems and Decision Processes. IEEE Transactions on Systems, Man and Cybernetics, p 29-44, 1973.

ZADEH, L. A., A theory of aproximate reasoning. Machine Intelligence, v. 9, pp. 149-194. New York, 1979.

ANEXO A:



Coluna de Destilação em Batelada do Laboratório de Controle da Área de Concentração de Sistemas de Processos Químicos e Informática, FEQ, UNICAMP.



Detalhe da Válvula Magnética e do Divisor de Corrente Usados no Refluxo de Líquido à Coluna de Destilação em Batelada.

ANEXO B:

Algoritmo do Método de Integração Numérica de Simpson. (aproxima a função a ser integrada por sucessivas parábolas).

inicio=limite_inferior ;limite inferior da f. de pertinência Gaussiana a ser integrada
fim=limite_superior ;limite superior da f. de pertinência Gaussiana a ser integrada

n=21 ;intervalos de integração
n1=n-1

h=(fim-inicio)/n1 ;passo na abcissa

X=inicio
coef=2
aux=-2

$t = e^{\frac{-(X-a)^2}{2\cdot\sigma^2}}$;cálculo da função gaussiana para x=início

X=fim

$t = e^{\frac{-(X-a)^2}{2\cdot\sigma^2}}$;cálculo da função gaussiana para x=fim

t=t+yy

faça i=1 até n1 ;laço de integração numérica
  x=x+h
    aux=-aux
    coef=coef+aux
    $t = e^{\frac{-(X-a)^2}{2\cdot\sigma^2}}$
    t=t+coef*yy

fim faça
AREA=t*h/3

ANEXO C:

Listagem do Programa Simulador da Coluna de Destilação em Batelada e do Módulo de Controle Nebuloso.

```
/* Programa:    NEWCONT.C
Escrito Por:  Ana Maria Frattini Fileti & Renato Dutra Pereira

/* ****************************************************
  * Simulacao de uma Coluna de Destilacao Batelada              *
  * controlada atraves de controle fuzzy                        *
  *                                                             *
  **************************************************** */
/* Observacoes :
1) Fluxos molares "nao" sao constantes;
2) Somente o refervedor e considerado estagio ideal;
3) Usa Refluxo total para estimativa do perfil de conc. inicial;
4) Hold-up constante (coluna) considerado na estim. inicial;
5) Hold-up de vapor desprezivel;
6) Hold-up de liquido variavel nos pratos;
7) Hold-up constante e suficientemente pequeno no condensador para que se possa considerar a temp. do recipiente
igual a do prim. prato;

8) Condensador total;
9) Eficiencia constante nos pratos;
10) Queda de pressao desprezivel;
11) Calor fornecido ao refervedor varia para manter V[n+1] constante; (ou seja, deve haver controle no sistema de
aquecimento)

12) Pressao deve ser baixa ou moderada devido ao uso da eq. virial truncada no segundo termo e eq. de antoine.

/*************************************************************/
#include <math.h>
#include <stdlib.h>
#include <stdio.h>
/*************************************************************/

/* Prototipo de Funcao(habilita checagem de tipos de argumentos das funcoes): */

extern void cfuzzy_(float*,float*,float*,int*);


/*************************************************************/


void somatx();
void somaty();
void ental();
void entvap();
void bolha();
void orvalho();
void uniq();
void viri();
void fugpad();
void CONTROLE();
```

```
int i,j,n,c,bo=1,it,polar[9],PH[9];
float x[9][18],y[9][18],E,gam[9],fi[9],fug0[9],K[9],t=0.0;
float Pc[9],Tc[9],cpl[9],cpv[9],Teb[9],To,P,alfa;
float h[18],H[18],he[18],Hvapr,Hvapc,Hvpteb[9],Hvap[9],Qc,Qr,
    T[18], hant[18];
float mdp[9],at[9],bt[9],omega[9],B[9][9]; /* virial (viri) */
float Vc[9],Aanto[9],Banto[9],Canto[9]; /* Antoine (fugpad) */
float unr[9],unq[9],una[9][9]; /* Parametros Uniquac (uniq) */
int comeca_erro=0, comeca_variacao_erro=0,FINO=0,muda=0,vezez=1;
double erro_anterior=0, variacao_erro=0, acao, incremento;
double RRanterior=0.5, RRanteriormaximo=10.5;
float erro_fortran, var_erro_fortran, acao_fortran,
    corrigir_fortran, variacao_anterior=0;
int segunda_variacao;
int rep,passou;

/*************************************************************/

void main(void)

{

char arq[30];
float bas,taux;
int iter,ii,ced=0,ent,aux1=0,aux2=1,ctem=0,ctem0=4,cont=0,
   contr=1, Vfix,na=1,pert;
float fun[9][18],k1[9][18],k2[9][18],k3[9][18],k4[9][18];
float xant[9][18],xsp[9],xmed[10],xc[18],xan[18],corr;
float Mc,So,Mrt,ro[18],d[18],fe,Ap,Lw,hv,rom; /* Holdup's */
float Raz,Lo,D,Dacum=0.0,Sini=0.0,pda=0.0,pb=-2.0;
float pas,tf,timp1=0.0,timp2=0.0,tx=1.e-20,soma,kc,Ti,Iant=0.0,I=0.0;
float desv,dif,fun1,V1,erant=0.0;
float xo[9],totcl[18],carg[18],somcl[18],som[18],Q[18],delta[18];
float km1[18],km2[18],km3[18],km4[18],func[18],L[18],M[18],
    Mant[18],V[18];
float somcom[9],somdes[9],compmed[9];
float TAUI=0.1,KC=200.0,Ranterior=0.0,somaerro=0.0,erro=0.0;
float XDSETPOINT=0.80,acao_anterior,amplitude,leitura;
int ON=0, chamadas=0,vezes=10,sinal;
double nn,iii;

FILE *a1;
FILE *fp1;
FILE *fp2;
FILE *fp3;
FILE *fp4;
if((fp1=fopen("topo.dat","w+b"))==NULL) {
   printf ("\n   Cannot open file\n\n");
   goto fim; }
if((fp2=fopen("percom.dat","w+b"))==NULL) {
   printf ("\n   Cannot open file\n\n");
   goto fim; }
if((fp3=fopen("pertem.dat","w+b"))==NULL) {
   printf ("\n   Cannot open file\n\n");
   goto fim; }
if((fp4=fopen("parproc.dat","w+b"))==NULL) {
```

```
   printf ("\n    Cannot open file PARPROC.DAT \n\n");
   goto fim; }
printf ("\n\n                COLUNA DE DESTILACAO BATELADA \n\n\n");
printf (" Entrada de dados via teclado (1) ou por arquivo (2) ?  ");
/*(scanf ("%d",&ent);*/
ent=2;
printf ("\nEnt= ",ent);

if (ent==1) {
printf ("\n Digite o num. de compon. e o num. de pratos:\n");
scanf ("%d,%d",&c,&n);
printf ("\n Digite os valores do passo e tf:\n");
scanf ("%f,%f",&pas,&tf);
printf ("\n Digite os valores da vazao de vapor(molar) e carga:\n");
printf ("   (Vazao inicial evaporada do refervedor)\n");
scanf ("%f,%f",&V[n+1],&So);
printf ("\n Digite efic.de estagio(%),ganho do control.e cte.tempo integral:\n");
scanf ("%f,%f,%f",&E,&kc,&Ti);
printf ("\n Digite o holdup no cond.:\n");
printf ("\n CUIDADO ! O hold-up do cond. deve ser pequeno em relacao aos pratos!!\n");
scanf ("%f",&Mc);
printf ("\n Digite fracao mol.carga e set-point para cada componente:\n");
for (i=0;i<c;i++) {
  scanf ("%f,%f",&xo[i],&xsp[i]); }
printf ("\n Digite o desvio permitido para o set-point :  ");
scanf ("%f",&desv);
printf ("\n");
}
else
{
printf ("\n Digite o num. de compon. e o num. de pratos:\n");
/*scanf ("%d,%d",&c,&n);*/
c=2;
n=12;

printf ("\n De o nome do arquivo:  ");
printf ("\n\n CUIDADO COM AS CONSTANTES DE ANTOINE - LOG,mmHg,Celsius ! \n");
scanf ("%s",arq);

if((a1=fopen(arq,"r")) == NULL) {
   printf ("\n\n       Nao existe o arquivo %-s !!!!!\n\n",arq);
   goto fim;          }

fscanf(a1,"%f %f %f %f",&Ap,&Lw,&hv,&E);
fscanf(a1,"%f %f %f",&kc,&Ti,&desv);
fscanf(a1,"%f %f %f %d %f %f",&So,&Mc,&P,&Vfix,&V[n+1],&Qr);
fscanf(a1,"%f %f %f",&pas,&tf,&To);
for (i=0;i<c;i++)       /* le primeiro todos Pc,Vc,Tc,Teb p/ cada comp. */
  { fscanf(a1,"%f %f %f %f",&Pc[i],&Vc[i],&Tc[i],&Teb[i]);
  }
for (i=0;i<c;i++)
  { fscanf(a1,"%f %f %f %f",&cpl[i],&cpv[i],&d[i],&Hvap[i]);
  }
for (i=0;i<c;i++)
  { fscanf(a1,"%d %d %f %f %f",&polar[i],&PH[i],&mdp[i],&at[i],&bt[i]);
  }
```

```
for (i=0;i<c;i++)
  { fscanf(a1,"%f %f %f %f",&omega[i],&Aanto[i],&Banto[i],&Canto[i]);
  }
for (i=0;i<c;i++)
  { fscanf(a1,"%f %f %f %f",&xo[i],&xsp[i],&unr[i],&unq[i]);
  }
for (i=0;i<c;i++)
  { for (ii=0;ii<c;ii++)
     { fscanf(a1,"%f ",&una[i][ii]);
     }
  }
fclose(a1);
}
printf("%f %f %f %f\n",Ap,Lw,hv,E);
printf("%f %f %f\n",kc,Ti,desv);
printf("%f %f %f %d %f %f\n",So,Mc,P,Vfix,V[n+1],Qr);
printf("%f %f %f\n",pas,tf,To);
for (i=0;i<c;i++)       /* le primeiro todos Pc,Vc,Tc,Teb p/ cada comp. */
  { printf("%f %f %f %f\n",Pc[i],Vc[i],Tc[i],Teb[i]);
  }
for (i=0;i<c;i++)
  { printf("%f %f %f %f\n",cpl[i],cpv[i],d[i],Hvap[i]);
  }
for (i=0;i<c;i++)
  { printf("%d %d %f %f %f\n",polar[i],PH[i],mdp[i],at[i],bt[i]);
  }
for (i=0;i<c;i++)
  { printf("%f %f %f %f\n",omega[i],Aanto[i],Banto[i],Canto[i]);
  }
  for (i=0;i<c;i++)
  { printf("%f %f %f %f\n",xo[i],xsp[i],unr[i],unq[i]);
  }
for (i=0;i<c;i++)
  { for (ii=0;ii<c;ii++)
     { printf("%f \n",una[i][ii]);
     }
  }

/* Calculo do hold-up maximo de cada prato (refluxo total - L[j]=V[n+1]) */
fe = 9.345e-3;  /* constante para eq. Francis */
printf ("%f",fe);
rom = 0;
for (i=0;i<c;i++)
  { rom = rom + (d[i] * xo[i]);  /* mas.esp.mol.da mistura inicial */
  }
printf ("\n Vn+1 = %f\n",V[n+1]);
Mrt = Ap * rom * (hv + (fe * pow((V[n+1]/(rom * Lw)),(2./3.))));  /* Eq. Francis */
printf ("\n Mrt = %f \n",Mrt);
Sini = So - (Mrt * n) - Mc;
M[n+1] = Sini;

/* iniciacao */

for (j=1;j<=n;j++) {
  M[j] = Mrt;     /* Considera-se hold-up inicial igual p/ todos pratos */
  }
```

```
To = 273.15 + To;
P = P * 101.325;
for (i=0;i<c;i++) {
  Tc[i] = 273.17 + Tc[i];      /* Transformacao das T em C para K */
  Teb[i] = 273.15 + Teb[i];
  Pc[i] = Pc[i] * 101.325;     /* Conversao da Pc de atm para kPa */
  Q[i] = 0.0;

  somcom[i] = 1.e-5;
  somdes[i] = 1.e-5;
   }

/* Estimativa da concentracao inicial no condensador */
printf("\n Estimativa inicial da concentracao no condensador: \n");

for (i=0;i<c;i++) {
/*   x[i][0] = 1.0 / c; */
  printf("\n x[i][0] = ");
  scanf ("%f",&x[i][0]);
  y[i][1] = x[i][0];
  xc[i] = x[i][0];
  xan[i] = 0.0;
  }

/* calculo das conc.iniciais prato a prato a partir da estimativa anterior */

calc:;
printf("\n xcond[0,1] = %f %f, k[0,1] referv.= %f %f",xc[0],xc[1],K[0],K[1]);
for (j=1;j<=(n+1);j++)
   { orvalho();   /* Devolvera os verdadeiros K[i]'s e x[i][j]'s */
     if (j!=(n+1))
       { for (i=0;i<c;i++)
           { y[i][j+1] = x[i][j];
/*           printf("\n K[%d] = %f e x[%d][%d] = %f",i,K[i],i,j,x[i][j]); */
           }
          }
   }

for (i=1;i<c;i++) {
  soma = 0;
  somcl[i] = 0;
  for (j=1;j<=n;j++) {
    somcl[i] = somcl[i] + x[i][j]; }
  fun1 = (x[i][n+1]*Sini)/(xo[i]*So - x[i][0]*Mc - somcl[i]*Mrt) - 1.0;
  printf ("\n fun1 = %f",fun1);
  if (fabs(fun1)<0.1)  continue;  /* Convergiu, vai para outro compon. */
  xan[i] = x[i][0];        /*   (fabs(xan[i]-x[i][0]) */
  if ((So*xo[i]-Mc*x[i][0]-Mrt*somcl[i])>0.000000)
    { corr = (So*xo[i] - Mc*x[i][0] - Mrt*somcl[i])/(Sini*x[i][n+1]);
      printf ("\n coef. de correcao de x[%d][0] = %f",i,corr);
      x[i][0] = corr*x[i][0]; }
  else {  dif = (x[i][n+1]*Sini - xo[i]*So + Mc*x[i][0] + Mrt*somcl[i]);
            if (dif < 0)  x[i][0] = 1.10 * x[i][0];
                 else x[i][0] = x[i][0]/1.10; }
  for (i=0;i<c;i++) {
    soma = soma + x[i][0]; }
```

```
  for (i=0;i<c;i++) {
    x[i][0] = x[i][0] / soma;
    xc[i] = x[i][0];
    y[i][1] = x[i][0]; }
  goto calc;
  }

/* Calculo do calor lat.mol. de vap. de cada comp. a temp. normal de ebulicao */
for (i=0;i<c;i++)
   { Hvpteb[i] = (Teb[i] * 2.17 * (log(Pc[i]/101.325) - 1))/(0.930 - Teb[i]/Tc[i]);
   }       /* Teb em Kelvin, Pc em atm, Hvpteb em cal/mol */

/* Estimativa inicial dos fluxos molares de liq. e vapor */

for (j=1;j<=(n+1);j++)
   { bolha();
     ental();
   }
entvap();
if (Vfix==1) {
Qr = V[n+1] * Hvapr; /* calor fornecido ao ref. para manter fluxo V[n+1] cte. */
        }
else V[n+1] = Qr/Hvapr; /* vazao de vapor do refervedor se Qr for cte. e nao V[n+1] */
L[n] = V[n+1];
for (j=n;j>0;j--)  {
   V[j] = (V[j+1] * (h[j] - H[j+1]))/(h[j-1] - H[j]);
   L[j-1] = V[j];  }
Qc = V[1] * Hvapc;  /* calor retirado no cond. p/ vapor se tornar liq.sat. */
printf ("\n Perfil de conc. inicial dos comp. 1,2,3 e 4 e de temp.:\n");
for (j=0;j<=(n+1);j++) {
  printf ("%f  %f  %f  %f %f\n",x[0][j],x[1][j],x[2][j],x[3][j],T[j]);
   }
bo = 2;
L[n+1] = 0.0;
V[0] = 0.0;
passou=0;
Raz = 0.01; /* valor inicial da razao de refluxo */
segunda_variacao=1;
acao_anterior=0;
do {
  V1 = 0.97 * V[1];
  taux = TAUX;


  RRanterior=0.01;
  ON=1;

  if ((x[0][0] < (XDSETPOINT) ) || (ON==1))
    {
      erro_anterior=erro;
      erro=XDSETPOINT-x[0][0];
      variacao_anterior=variacao_erro;
      variacao_erro=erro-erro_anterior;
      RRanterior=Raz;
      erro_fortran=erro;
      var_erro_fortran=variacao_erro ;
```

```
        cfuzzy_(&erro_fortran,&var_erro_fortran,&acao_fortran,
            &segunda_variacao);
        acao=acao_fortran;

    rep=rep+1;
    if (rep>=6)
      {
       rep=1;
       Raz=RRanterior+acao;
      }
    else Raz=RRanterior;
    acao_anterior=acao;
  }
   else Raz=0.01;


if (Raz>100) Raz=100;
if (Raz<0.01) Raz=0.01;
L[0] = (Raz/(Raz+1)) * V[1];
if (L[0]<1.e-20) {
                        Raz = 1.e-20;
                  L[0] = 1.e-20;
                  }

if (L[0]>(0.99*V[1])) L[0] = 0.99*V[1];
D = V[1] - L[0];
somcom[ced] = somcom[ced] + x[ced][0]*D;
somdes[ced] = somdes[ced] + D;
compmed[ced] = somcom[ced]/somdes[ced];


if (x[ced][0]>(xsp[ced]-desv))
   Q[ced] = Q[ced] + (x[ced][0] * D * pas);


// if (t>=taux) fprintf (fp4,"  %f %f %f %f %f %f %f\r\n",t,L[0],Raz,x[ced][n+1],kc,Ti,compmed[ced]);


/* gravacao em arquivo */
if (t>=timp1) {     /* grava comp.de todos comp. no cond. */
fprintf (fp1,"%f %f %f %f %f %f ",t,L[0],pda,Qr,Qc,(Q[ced]*100.0/(So*xo[ced])));
for (i=0;i<c;i++)
  { if (i==(c-1)) fprintf (fp1,"%-5.4f %-5.4f\r\n",x[c-1][n+1],x[c-1][0]);
    else fprintf (fp1,"%-5.4f %-5.4f ",x[i][n+1],x[i][0]);
  }
printf("\n Composicao media no destilado = %f",compmed[ced]);
printf("\n Erro = %f",erro);
printf("\n Variacao do Erro = %f",variacao_erro);
printf("\n Razao de Refluxo Atual = %f\n",Raz);
printf("\n Leitura de Composicao no Topo = %f\n",leitura);


timp1 = timp1 + (10. * pas);
                  }

if (t>=timp2) {
```

```
  fprintf (fp2,"%f %f ",t,Raz);  /* grava o tempo e o perfil de todos os compon. */
  for (i=0;i<c;i++)
    { for (j=0;j<=(n+1);j++)
            { if (j==(n+1) && i==(c-1)) fprintf (fp2,"%-5.4f\r\n",x[i][n+1]);
              else fprintf (fp2,"%-5.4f ",x[i][j]);
        }
    }
   fprintf (fp3,"%f ",t);  /* grava o perfil de temper. e vaz.liq. p/ varios t */
   for (j=0;j<=(n+1);j++)
     { if (j==(n+1)) fprintf (fp3,"%-6.3f %-6.4f %-6.4f\r\n",(T[n+1]-273.15));
       else fprintf (fp3,"%-6.3f %-6.4f %-6.4f ",(T[j]-273.15));
     }
      /* impressao na tela do perfil da coluna */
   printf ("\nTempo   Conc.0   Conc.1   Conc.2   Temper.   Vaz.vap. Vaz.liq.  Hold-up \n");
   M[0] = Mc;
   for (j=0;j<=(n+1);j++)
     { printf ("%6.2f   %5.4f   %5.4f   %5.4f   %6.3f   %4.3f   %4.3f
%6.3f\n",t,x[0][j],x[1][j],x[2][j],T[j],V[j],L[j],M[j]);
     }
    timp2 = timp2 + (10.0 * pas);
                    }
   t = t + pas;
   Dacum = Dacum + D*pas;
   pda = (Dacum/Sini) * 100.0;

/* Integracao por Runge-Kutta de ordem 4  -  BALANCOS MASSICOS */

   iter = 1;
   do {
      for (i=0;i<c;i++) {
            /* Condensador - Composicao */
       fun[i][0] = V[1] * (y[i][1] - x[i][0])/Mc;
            /* Refervedor - Composicao */
       fun[i][n+1]=(L[n]*(x[i][n]-x[i][n+1])+V[n+1]*(x[i][n+1]-y[i][n+1]))/M[n+1];
                  }
       /* Refervedor - Hold-up */
      func[n+1] = L[n] - V[n+1];
       /* Pratos */
      for (j=1;j<=n;j++) {
            /* Hold-up */
       func[j] = L[j-1] + V[j+1] - L[j] - V[j];
       for (i=0;i<c;i++)  {
            /* Composicao */
        fun[i][j]=(V[j+1]*(y[i][j+1]-x[i][j])+V[j]*(x[i][j]-y[i][j])+L[j-1]*(x[i][j-1]-x[i][j]))/M[j];
                  }
                              }

      switch(iter) {

          case 1:
        for (j=0;j<=(n+1);j++) {
            ro[j] = 0.0;
         for (i=0;i<c;i++) {
          k1[i][j] = fun[i][j];
          xant[i][j] = x[i][j];
          x[i][j] = x[i][j] + (0.5 * pas * k1[i][j]);
```

```
    ro[j] = ro[j] + (x[i][j] * d[i]); /*
                                    mas.esp.mol.media no prato j */
                                          }
                                      if (j!=0)
                                    { Mant[j] = M[j];
                                     km1[j] = func[j];
                                     M[j] = M[j] + (0.5 * pas * km1[j]);
                                           if (j!=(n+1))

                  { bas = (1./fe) * (M[j]/(Ap*ro[j]) - hv);
           if (bas<1.e-10) L[j] = 0.0;
                    else L[j] = ro[j] * Lw * pow(bas,1.5);
          }
         bolha();                  /* Eq. Francis */
         ental();
       }
      somatx();
      somaty();
                    }
   entvap();   /* refervedor */
   if (Vfix==1) {
       Qr = V[n+1] * Hvapr;  /*Consid.V[n+1] cte, Qr tem que
                  variar */
             }
   else V[n+1] = Qr/Hvapr; /* Com Qr cte. quem varia e
                     V[n+1] */
       /* Balancos de Energia */
   for (j=n;j>=1;j--)      /* pratos */
    { delta[j] = (h[j] - hant[j])/pas; /* aprox.por
                      dif.finitas de dh/dt */
      V[j] = ((M[j] * delta[j]) + L[j-1] * (h[j] - h[j-1]) + V[j+1] * (h[j] - H[j+1]))/(h[j] - H[j]);
    }
   Qc = V[1] * Hvapc;   /* Condensador */
   break;

  case 2:
   for (j=0;j<=(n+1);j++) {
        ro[j] = 0.0;
    for (i=0;i<c;i++) {
      k2[i][j] = fun[i][j];
      x[i][j] = x[i][j] + (0.5 * pas * k2[i][j]);
      ro[j] = ro[j] + (x[i][j] * d[i]);  /*
                  mas.esp.mol.media no prato j */
                  }
     if (j!=0)
       { km2[j] = func[j];
        M[j] = M[j] + (0.5 * pas * km2[j]);
        if (j!=(n+1))
                  { bas = (1./fe) * (M[j]/(Ap*ro[j]) - hv);
          if (bas<1.e-10) L[j] = 0.0;
          else L[j] = ro[j] * Lw * pow(bas,1.5);
                  }
        bolha();
        ental();
       }
         somatx();
```

```
 somaty();
                }
entvap();   /* refervedor */
if (Vfix==1) {
Qr = V[n+1] * Hvapr;  /* Consid.V[n+1] cte, Qr tem que
                variar */
        }
else V[n+1] = Qr/Hvapr; /* Com Qr cte. quem varia e
                V[n+1] */
/* Balancos de Energia */
    for (j=n;j>=1;j--)     /* pratos */
 { delta[j] = (h[j] - hant[j])/pas; /* aprox.por
                dif.finitas de dh/dt */
   V[j] = ((M[j] * delta[j]) + L[j-1] * (h[j] - h[j-1]) + V[j+1] * (h[j] - H[j+1]))/(h[j] - H[j]);
 }
    Qc = V[1] * Hvapc; /* Condensador */
break;

case 3:
 for (j=0;j<=(n+1);j++) {
 ro[j] = 0.0;
 for (i=0;i<c;i++) {
       k3[i][j] = fun[i][j];
   x[i][j] = x[i][j] + (pas * k3[i][j]);
       ro[j] = ro[j] + (x[i][j] * d[i]);  /*
         mas.esp.mol.media no prato j */
             }
 if (j!=0)
    { km3[j] = func[j];
     M[j] = M[j] + (pas * km3[j]);
     if (j!=(n+1))
             { bas = (1./fe)*(M[j]/(Ap*ro[j])-hv);
       if (bas<1.e-10) L[j] = 0.0;
       else L[j] = ro[j] * Lw * pow(bas,1.5);
      }
     bolha();
           ental();
    }
      somatx();
 somaty();
                }
entvap();   /* refervedor */
if (Vfix==1) {
    Qr = V[n+1] * Hvapr;  /* Consid.V[n+1] cte, Qr tem que
                variar */
        }
else V[n+1] = Qr/Hvapr; /* Com Qr cte. quem varia e
                V[n+1] */
/* Balancos de Energia */
for (j=n;j>=1;j--)     /* pratos */
 { delta[j] = (h[j] - hant[j])/pas; /* aprox.por
                dif.finitas de dh/dt */
   V[j] = ((M[j] * delta[j]) + L[j-1] * (h[j] - h[j-1]) + V[j+1] * (h[j] - H[j+1]))/(h[j] - H[j]);
 }
    Qc = V[1] * Hvapc; /* Condensador */
break;
```

```
    case 4:
     for (j=0;j<=(n+1);j++) {
          ro[j] = 0.0;
      for (i=0;i<c;i++) {
       k4[i][j] = fun[i][j];
       x[i][j]=xant[i][j]+pas/6.0*(k1[i][j]+2.0*k2[i][j]+2.0*k3[i][j]+k4[i][j]);
       ro[j] = ro[j] + (x[i][j] * d[i]); /* mas.esp.mol.media no prato j */
                  }
      if (j!=0)
                { km4[j] = func[j];
                  M[j] = Mant[j] + pas/6.0 * (km1[j]+2.0*km2[j]+2.0*km3[j]+km4[j]);
         if (j!=(n+1))
                   { bas = (1./fe) * (M[j]/(Ap*ro[j]) - hv);
           if (bas<1.e-10) L[j] = 0.0;
           else L[j] = ro[j] * Lw * pow(bas,1.5);
          }
         bolha();
       /*  printf("\n gam[i]={%f,%f,%f}",gam[0],gam[1],gam[2]);
         printf("\n Bii={%f,%f,%f} na Temp=%f",B[0][0],B[1][1],B[2][2],T[j]);
         printf("\n Ki={%f,%f,%f}",K[0],K[1],K[2]); */
         ental();
        }
          somatx();
          somaty();
                 }
      entvap();   /* refervedor */
      if (Vfix==1) {
      Qr = V[n+1] * Hvapr;  /* Consid.V[n+1] cte, Qr tem que variar */
            }
         else V[n+1] = Qr/Hvapr; /* Com Qr cte. quem varia e V[n+1] */
      /* Balancos de Energia */
      for (j=n;j>=1;j--)     /* pratos */
       { delta[j] = (h[j] - hant[j])/pas; /* aprox.por dif.finitas de dh/dt */
        V[j] = ((M[j] * delta[j]) + L[j-1] * (h[j] - h[j-1]) + V[j+1] * (h[j] - H[j+1]))/(h[j] - H[j]);
       }
      Qc = V[1] * Hvapc; /* Condensador */
         break;
           }
   iter = iter + 1;
  } while (iter<=4);

if (pb<0.00001)
  { if (x[c-1][0]>=xsp[c-1])  { pb = t;
                    printf ("\n VAI SAIR !!!\n");
                    goto fim;
                   }
  }
  /* Poderia parar a batelada e recolher c-1 no fundo da coluna  */

} while (t<=tf && M[n+1]>V[n+1] && V[n+1]>0.000000000000001);
fim:
printf ("\nxo={%-3.2f,%-3.2f,%-3.2f,%-3.2f}, ced=%d, desv=%-5.4f\n",
xo[0],xo[1],xo[2],xo[3],ced+1,desv);
printf ("passo=%-5.4f\n",pas);
printf ("xsp={%-3.2f,%-3.2f,%-3.2f,%-3.2f}, aux1=%d, aux2=%d\n",
```

```
xsp[0],xsp[1],xsp[2],xsp[3],aux1,aux2);
printf ("\n %% moles acima espec.no dest.={%-4.2f,%-4.2f,%-4.2f,%-4.2f}, tempo total=%-6.3f\n",
(Q[0]*100.0/(So*xo[0])),(Q[1]*100.0/(So*xo[1])),(Q[2]*100.0/(So*xo[2])),(Q[3]*100.0/(So*xo[3])),t);
printf ("\n\n Batelada poderia ter sido finalizada no tempo %f !!!",pb);
printf ("\n (O componente c-1 seria recolhido no fundo da coluna)");
fclose(fp1);
fclose(fp2);
fclose(fp3);
fclose(fp4);
}

void somatx()
{
float somx=0.0;
for (i=0;i<c;i++) {
  somx = somx + x[i][j]; }
if (fabs(1.0-somx)>0.01) {
   printf ("\n somx = %f no prato %d no tempo %f\n",somx,j,t);
   printf ("    CUIDADO  !!!!!!!!!!!! \n");
   return; }
}

void somaty()
{
float somy=0.0;
if (j!=0) {
for (i=0;i<c;i++) {
  somy = somy + y[i][j]; }
if (fabs(1.0-somy)>0.01) {
   printf ("\nsomy = %f no prato %d no tempo %f\n",somy,j,t);
   printf ("    CUIDADO  !!!!!!!!!!!! \n");
   return;      }
   }
}

void ental()
{
float hid[18],hcom[9][18],Hcom[9][18];
 /* Calculo das entalpias necessarias para balanco de energia -
   Entalpias do liquido e do vapor nos pratos                  */

 hid[j] = 0;
 H[j] = 0;
 for (i=0;i<c;i++)          /* cp em cal/mol.K - To em K */
  {  hcom[i][j] = cpl[i] * (To - T[j]);
        /* ent.mol.de i puro liq. a temp.T do pr. j (cal/mol) */
    Hcom[i][j] = Hvap[i] + cpv[i] * (T[j] - To);
           /* ent.mol.de i puro gas. a temp.T do pr. j (cal/mol) */
    hid[j] = hid[j] + (x[i][j] * hcom[i][j]);
        /* ent.mol.de solucao liq.ideal no pr. j (cal/mol) */
    H[j] = H[j] + (y[i][j] * Hcom[i][j]);
        /* ent.total molar da fase gasosa no pr.j (cal/mol) */
  }
 hant[j] = h[j];
 h[j] = hid[j] + he[j];  /* ent. excesso he calculada em uniq() */
           /* entalpia total molar da fase liq. no prato j (cal/mol) */
```

```
}


void entvap()
{
 float Hvpc[9],Hvpr[9];
 /* Calculo das entalpias de vaporizacao para referv. e condens. */

 Hvapc = 0;
 Hvapr = 0;
 T[0] = T[1]; /* temp. no cond. = temp. prato 1 */
 for (i=0;i<c;i++)
    { Hvpc[i] = Hvpteb[i] * pow(((1 - T[0]/Tc[i])/(1 - Teb[i]/Tc[i])),0.38);
                    /* cal.lat.mol.vap.de i puro a temp.do condensador */
      Hvpr[i] = Hvpteb[i] * pow(((1 - T[n+1]/Tc[i])/(1 - Teb[i]/Tc[i])),0.38);
                    /* cal.lat.mol.vap.de i puro a temp. do refervedor */
      Hvapc = Hvapc + (y[i][1] * Hvpc[i]);
                  /* cal.lat.mol.vap.da solucao liq. no cond. */
      Hvapr = Hvapr + (x[i][n+1] * Hvpr[i]);
                  /* cal.lat.mol.vap.da solucao liq. no referv. */
    }
}


void orvalho()
{
/* Calculo da temperatura de orvalho e da razao de equilibrio p/ estimativa
do perfil inicial de temperatura e composicao da coluna              */
float T1=0,S,Sant,SS,fun1,fun2,dfdt;
int orv;

/* Estimativa inicial */
for (i=0;i<c;i++)
   { T1 = T1 + ((Teb[i]+10.0) * y[i][j]);
   }
T[j] = T1;

vir:
it = 2;
orv = 1;
viri();
fugpad();

quac:
if (orv==1)
  { Sant = 0.0;
    for (i=0;i<c;i++) gam[i] = 1.0;  /* fase liq. ideal na prim. iteracao */
  }
 else { uniq();
    /*   Sant = S; */
     }
 S = 0.0;
 for (i=0;i<c;i++)
    { K[i] = (gam[i] * fug0[i])/(fi[i] * P);
  /*  printf("\n gam[%d] = %f, fug0[%d]=%f, fi[%d]=%f,P=%f",i,gam[i],i,fug0[i],i,fi[i],P); */
      x[i][j] = y[i][j]/K[i];
```

```
    S = S + x[i][j];
   }
if (fabs(Sant - S) > 0.001)   /* as somatorias de x ainda nao se tornaram ctes. */
  { for (i=0;i<c;i++)
          { x[i][j] = x[i][j]/S;   /* normalizacao dos x */
          }
    Sant = S;
    orv = orv + 1;
    goto quac;
   }
fun1 = log(S);
if (fabs(fun1) > 0.001)   /* somatoria de x ainda nao e = 1 */
  { T1 = T1 + 1.0;    /* correcao da temp. por Newton-Raphson */
    T[j] = T1;
    uniq();
    viri();
    fugpad();
    SS = 0;
    for (i=0;i<c;i++)
          { K[i] = (gam[i] * fug0[i])/(fi[i] * P);
            x[i][j] = y[i][j]/K[i];
            SS = SS + x[i][j];
          }
    fun2 = log(SS);
    T1 = T1 - 1.0;

    dfdt = (fun2 - fun1)/1.0;    /* aprox. por dif. finitas */

    T[j] = T1 - (fun1/dfdt);
    T1 = T[j];
    goto vir;
   }
}


void bolha()
{
/* Calculo da temperatura de cada estagio
  - Metodo iterativo de Newton-Raphson -     */
/* Calculo da composicao de equilibrio do vapor */

float S,SS,Sant,T1=0,fun1,fun2,dfdt,yeq[9][18];

/* Estimativa inicial */
for (i=0;i<c;i++)
   {  T1 = T1 + (x[i][j] * Teb[i]);        /* T EM KELVIN !!!!!  */
   }       /* chute bom para pressoes proximas a atmosferica */
T[j] = T1;

uniquac:
it = 1;
uniq();

virial:
if (it==1)
  { Sant = 0.0;
```

```
    viri(); /* nao serao calculados os fi,mas sim os Bik */
    fugpad();
    for (i=0;i<c;i++)  fi[i] = 1.0; /* fase gasosa ideal na primeira iteracao */
   }
else { viri();
      fugpad();
   /*   Sant = S;  */
    }
S = 0.0;
for (i=0;i<c;i++)
   { K[i] = (gam[i] * fug0[i])/(fi[i] * P);
     y[i][j] = K[i] * x[i][j];
     S = S + y[i][j];
   }
if (fabs(Sant - S) > 0.001)  /* as somatorias de y ainda nao se tornaram ctes. */
   { for (i=0;i<c;i++)
           { y[i][j] = y[i][j]/S;    /* normalizacao dos y  */
           }
     Sant = S;
     it = it + 1;
     goto virial;
   }

fun1 = log(S);
if (fabs(fun1) > 0.001)  /* somatoria de y ainda nao e = 1 */
   { T1 = T1 + 1.0;   /* correcao da temperatura por Newton-Raphson */
     T[j] = T1;
     uniq();
     viri(); /* calcula fi considerando a mesma composicao da temp.anterior */
     fugpad();
     SS = 0;
     for (i=0;i<c;i++)
           { K[i] = (gam[i] * fug0[i])/(fi[i] * P);
        y[i][j] = K[i] * x[i][j];
        SS = SS + y[i][j];
       }
     fun2 = log(SS);
     T1 = T1 - 1.0;

     dfdt = (fun2 - fun1)/1.0;    /* aproximacao por dif. finitas */

     T[j] = T1 - (fun1/dfdt);
     T1 = T[j];
     goto uniquac;
    }

 for (i=0;i<c;i++)
   { yeq[i][j] = y[i][j];
     if (j!=(n+1) && bo!=1) { /* refervedor e estagio ideal e bo=1 tambem */
           y[i][j] = E/100.0 * (yeq[i][j] - y[i][j+1]) + y[i][j+1];
                   }
   }
 }


 void viri()
```

```
{
/* Calculo do coeficiente de fugacidade fi[i] pela Equacao Virial; */

float mdpr[9],Tr,term,Tcm,Pcm,omegam,f0,f1,f2,Bm,som;
int k;

/* Estimativa dos seg. coef. vir. puros e cruzados da mistura gasosa pelas
correlacoes de Tsonopoulos */

for (i=0;i<c;i++)
  { if (polar[i]==1)  /* gas polar */
    {  if (PH[i]==0)
          { /* nao forma ponte de hidrogenio */
            mdpr[i] = (986.4 * pow(mdp[i],2.0) * Pc[i])/pow(Tc[i],2.0);
        /* momento dipolar reduzido - Pc em kPa - Tc em K */
            at[i] = (-2.14e-4 * mdpr[i]) - (4.308e-21 * pow(mdpr[i],8.0));
            bt[i] = 0.0;
          }
     /* Se for polar e formar pte. de H, at e bt devem ter sido lidos no inicio */
     }
   /* Se nao for polar, estes calculos nao sao necessarios */
  }
 for (i=0;i<c;i++)
  { for (k=0;k<c;k++)       /* k tambem e componente */
     { if (i==k)  Tr = T[j]/Tc[i];
          else { term = pow(Vc[i],(1./3.)) + pow(Vc[k],(1./3.));
                 Tcm = sqrt(Tc[i]*Tc[k]) * (Vc[i] * Vc[k])/pow((0.5*term),6.0);
                 Pcm = 4.0*Tcm * (Pc[i]*Vc[i]/Tc[i] + Pc[k]*Vc[k]/Tc[k])/pow(term,3.0);
           /* Volume critico em m3/kmol */
           omegam = 0.5 * (omega[i] * omega[k]);
                 Tr = T[j]/Tcm;
          }
           f0 = 0.1445 - 0.33/Tr - 0.1385/pow(Tr,2.0) - 0.0121/pow(Tr,3.0) - 6.07e-4/pow(Tr,8.0);
           f1 = 0.0637 + 0.331/pow(Tr,2.0) - 0.423/pow(Tr,3.0) - 0.008/pow(Tr,8.0);

           if (polar[i]==1)
             { if (i==k)  f2 = at[i]/pow(Tr,6.0) - bt[i]/pow(Tr,8.0);
         else if (polar[k]==1) /* componentes diferentes mas ambos polares */
                    { f2 = 0.5*(at[i]+at[k])/pow(Tr,6.0) - 0.5*(bt[i]+bt[k])/pow(Tr,8.0);
                    }
             }
        else f2 = 0.0;
        if (i==k) B[i][i] = (8.31439*Tc[i]/Pc[i]) * (f0 + omega[i]*f1 + f2);
        else B[i][k] = (8.31439*Tcm/Pcm) * (f0 + omegam*f1 + f2);
            B[k][i] = B[i][k];
      }
  }
 if (it!=1)
  { Bm = 0.0;
    for (i=0;i<c;i++)
      { for (k=0;k<c;k++)  /* Segundo coeficiente virial da mistura gasosa */
            { Bm = Bm + (y[i][j] * y[k][j]) * B[i][k];
            }                              /* Bm em m3/kmol */
      }
    for (i=0;i<c;i++)
      { som = 0;
```

```
          for (k=0;k<c;k++)
            { som = som + y[k][j]*B[i][k];
            }
          fi[i] = P * ((2.0 * som) - Bm)/(8.31439 * T[j]);
          fi[i] = exp(fi[i]);  /* coef. de fugacidade do comp. i */
         }
  }
}


void fugpad()
{
float Pv[9],fisat[9];
/* Calculo da fugacidade no estado padrao fug0[i] */

/* Calculo das pressoes de saturacao pela eq. empirica de Antoine */
for (i=0;i<c;i++)
  { T[j] = T[j] - 273.15;  /* Pv em mmHg - T em Celsius - CUIDADO */
    Pv[i] = Aanto[i] - (Banto[i]/(T[j] + Canto[i]));
          /* Pv em mmHg - T em K - Cuidado com as ctes.!!!! */
 /*  Pv[i] = exp(Pv[i]);
   CUIDADO LOGARITMO NEBERIANO */
    Pv[i] = pow(10.0,Pv[i]);  /* CUIDADO LOG NA BASE 10 */
    T[j] = T[j] + 273.15;  /* volta T[j] para Kelvin */
    Pv[i] = Pv[i] * 0.13332237;   /* Pv de mmHg para kPa */
                                  /* o calculo dos coef. vir. nao muda */
    fisat[i] = (Pv[i] * B[i][i])/(8.31439 * T[j]);
    fisat[i] = exp(fisat[i]);  /* Virial para comp i puro */
/*  printf("\n B[%d][%d] = %f, fisat[%d] = %f, T[%d]=%f",i,i,B[i][i],i,fisat[i],j,T[j]); */
    fug0[i] = fisat[i] * Pv[i];  /* fugac. do liq. no est. padrao */
  }
}


void uniq()
{
/* Calculo do coeficiente de atividade pelo modelo de UNIQUAC
   com parametros UNIFAC  */
float som,somqx,somrx,somki,somsk,somdiv,somln,somtt,soma;
float gamC[9],gamR[9],l[9],theta[9],unfi[9],tal[9][9];
int k,s;

/* Contribuicao combinatorial (gamC e na verdade ln gamC) */
somqx = 0.0;
somrx = 0.0;
for (k=0;k<c;k++)
  { somqx = somqx + (unq[k] * x[k][j]);
    somrx = somrx + (unr[k] * x[k][j]);
    l[k] = 5.0 * (unr[k] - unq[k]) - (unr[k] - 1.0);
    for (s=0;s<c;s++)
          { tal[k][s] = exp(-una[k][s]/(T[j]*1.98721));
          }      /* una[i][k] em cal/mol, R=1.98721 cal/mol.K */
  }
som = 0.0;
for (k=0;k<c;k++)
  { som = som + (x[k][j] * l[k]);
```

```
  }
for (i=0;i<c;i++)
  { theta[i] = (unq[i] * x[i][j])/somqx;
    unfi[i] = (unr[i] * x[i][j])/somrx;
    gamC[i] = log(unfi[i]/x[i][j]) + 5.0*unq[i]*log(theta[i]/unfi[i]) + l[i] - (unfi[i]/x[i][j])*som;
  }

 /* Contribuicao residual (gamR e na verdade ln gamR) */

for (i=0;i<c;i++)
  { somki = 0;
    somdiv = 0;
    for (k=0;k<c;k++)
      { somki = somki + (theta[k] * tal[k][i]);
        somsk = 0;
        for (s=0;s<c;s++)
              { somsk = somsk + (theta[s] * tal[s][k]);
              }
        somdiv = somdiv + (theta[k]*tal[i][k])/somsk;
          }
    gamR[i] = unq[i] * (1.0 - log(somki) - somdiv);
  }

 for (i=0;i<c;i++)    /* Coef. de ativ. do comp. i no prato j */
  { gam[i] = exp(gamC[i]+gamR[i]);
  }

 /* Calculo da entalpia de excesso do prato j */
 soma = 0;
 for (i=0;i<c;i++)
  { somln = 0;
    somtt = 0;
    for (k=0;k<c;k++)
      { somln = somln + (theta[k]*tal[k][i]*log(tal[k][i]));
          somtt = somtt + (theta[k] * tal[k][i]);
        }
    soma = soma + unq[i]*x[i][j]*(somln/somtt);
  }
 he[j] = -1.9872 * T[j] * soma;
                    /* entalpia em cal/mol, temp em K, R em cal/mol.K */
 }



 C    MÓDULO DO CONTROLADOR FUZZY EM FORTRAN
 C    RENATO DUTRA PEREIRA FILHO - 3/06/1998
 C    DIMENSIONAMENTO DAS MATRIZES


      SUBROUTINE CFUZZY(ERRO,VARERRO,ACAO,VAR2ERRO)


      REAL LIE,LSE,LSVE,LIVE,LIA,LSA,RE,RVE,RA,SOMAE,SOMAVE
      REAL ATIVAR,LE,LA,LVE,TESTE
      REAL RESQUERDA,RDIREITA,TRIANGULO,L2SVE
      REAL PE(7,3),PA(7,3),PVE(7,3),BR(7,7)
```

```
      REAL FPE(7,3),FPVE(7,3),FPA(7,3)
      REAL NOVOFPE(7,3),NOVOFPVE(7,3),NOVOFPA(7,3)

      REAL WATOT, ATOT,VA,BASE,PESO,AREA,VALORR,MENOR,CROSSOVER,ELE,erre
      INTEGER XX,YY,P,CC,L,I, FUNCOES,FIM,ESPERA,C,J
      REAL KP,KI,KD,CROSS
      REAL L2VE, LI2VE, LS2VE, R2VE
      INTEGER COMECAERRO,ACABAERRO,COMECAVARERRO,ACABAVARERRO,SACO
      INTEGER COMECA2VARERRO, ACABA2VARERRO,CONTADOR
      INTEGER ANTAGORDA1,ANTAGORDA2

C     INICIALIZACAO DOS PARAMETROS DO CONTROLADOR
      FUNCOES=7.0
      CROSSOVER=0.50

      DO XX=1,FUNCOES
      DO YY=1,FUNCOES
      VALORR=XX+YY-((FUNCOES+1.0)/2)
      IF (VALORR .LT. 1.0) VALORR=1.0
      IF (VALORR .GT. FUNCOES) VALORR=FUNCOES
      BR(YY,XX)=VALORR
      END DO
      END DO

      LIE=-0.01
      LSE=+0.01

      LSVE=+1e-1
      LIVE=-1e-1

      LIA=-2.5
      LSA=+2.5


      RE=LSE-LIE
      RVE=LSVE-LIVE
      RA=LSA-LIA

C     ATRIBUICAO DO CENTRO E LARGURA DAS FUNCOES DE PERTINENCIA
C     PARA AS FPS TRIANGULARES IGUALMENTE ESPACADAS
C     USADAS PARA TESTE DAS FPS TRIANGULARES QUAISQUER
C     LARGURAS

      DO I=1,FUNCOES
      PE(I,1)=RE/((FUNCOES-1)/2)
      PVE(I,1)=RVE/((FUNCOES-1)/2)
      PA(I,1)=RA/((FUNCOES-1)/2)
      END DO

C     CENTROS
      DO I=2,FUNCOES-1
      PE(I,2)=(RE/(FUNCOES-1.0))*(I-1) + LIE
      PVE(I,2)=(RVE/(FUNCOES-1.0))*(I-1) + LIVE
      PA(I,2)=(PA(I,1)/2)*(I-1)+LIA
      END DO
```

```
      PE(1,2)=LIE
      PVE(1,2)=LIVE
      PA(1,2)=LIA
      PE(FUNCOES,2)=LSE
      PVE(FUNCOES,2)=LSVE
      PA(FUNCOES,2)=LSA


C     ATRIBUICAO DOS VALORES DA FP TRIANGULAR COM CROSSOVER DE 50%
C     PARA FPE, FPVE  E FPA

      DO I=2, FUNCOES-1
      FPE(I,1)=PE(I-1,2)
      FPVE(I,1)=PVE(I-1,2)
      FPA(I,1)=PA(I-1,2)

      FPE(I,3)=PE(I+1,2)
      FPVE(I,3)=PVE(I+1,2)
      FPA(I,3)=PA(I+1,2)

      FPE(I,2)=PE(I,2)
      FPVE(I,2)=PVE(I,2)
      FPA(I,2)=PA(I,2)
      END DO


      FPE(1,1)=0.0
      FPE(1,2)=PE(1,2)
      FPE(1,3)=PE(2,2)

      FPVE(1,1)=0.0
      FPVE(1,2)=PVE(1,2)
      FPVE(1,3)=PVE(2,2)

      FPA(1,1)=0.0
      FPA(1,2)=PA(1,2)
      FPA(1,3)=PA(2,2)

      I=FUNCOES
      FPE(I,1)=PE(I-1,2)
      FPE(I,2)=PE(I,2)
      FPE(I,3)=0.0

      FPVE(I,1)=PVE(I-1,2)
      FPVE(I,2)=PVE(I,2)
      FPVE(I,3)=0.0

      FPA(I,1)=PA(I-1,2)
      FPA(I,2)=PA(I,2)
      FPA(I,3)=0.0


C     CALCULO DAS NOVAS FP USANDO O CROSSOVER DIFERENTE DE 50% - PE
```

```
      ELE=(FPE(2,3)-FPE(2,1))/4
      ERRE=CROSSOVER*ELE/(1-CROSSOVER)

      NOVOFPE(1,1)=0.0
      NOVOFPE(1,2)=FPE(1,2)
      NOVOFPE(1,3)=FPE(1,2)+ELE+ERRE

      NOVOFPE(FUNCOES,1)=FPE(FUNCOES,2)-ELE-ERRE
      NOVOFPE(FUNCOES,2)=FPE(FUNCOES,2)
      NOVOFPE(FUNCOES,3)=0

      DO I=2, FUNCOES-1
      NOVOFPE(I,1)=FPE(I,2)-ELE-ERRE
      NOVOFPE(I,2)=FPE(I,2)
      NOVOFPE(I,3)=FPE(I,2)+ELE+ERRE
      END DO

      DO I=1,FUNCOES
      FPE(I,1)=NOVOFPE(I,1)
      FPE(I,2)=NOVOFPE(I,2)
      FPE(I,3)=NOVOFPE(I,3)
      END DO

C FIM DE FPE
C     CALCULO DAS NOVAS FP USANDO O CROSSOVER DIFERENTE DE 50% - PVE

      ELE=(FPVE(2,3)-FPVE(2,1))/4
      ERRE=CROSSOVER*ELE/(1-CROSSOVER)

      NOVOFPVE(1,1)=0.0
      NOVOFPVE(1,2)=FPVE(1,2)
      NOVOFPVE(1,3)=FPVE(1,2)+ELE+ERRE

      NOVOFPVE(FUNCOES,1)=FPVE(FUNCOES,2)-ELE-ERRE
      NOVOFPVE(FUNCOES,2)=FPVE(FUNCOES,2)
      NOVOFPVE(FUNCOES,3)=0.0


      DO I=2, FUNCOES-1
      NOVOFPVE(I,1)=FPVE(I,2)-ELE-ERRE
      NOVOFPVE(I,2)=FPVE(I,2)
      NOVOFPVE(I,3)=FPVE(I,2)+ELE+ERRE
      END DO



      DO I=1,FUNCOES
      FPVE(I,1)=NOVOFPVE(I,1)
      FPVE(I,2)=NOVOFPVE(I,2)
      FPVE(I,3)=NOVOFPVE(I,3)
      END DO

C FIM DE FPVE

C     CALCULO DAS NOVAS FP USANDO O CROSSOVER DIFERENTE DE 50% - FPA
```

```
      ELE=(FPA(2,3)-FPA(2,1))/4
      ERRE=CROSSOVER*ELE/(1-CROSSOVER)
      NOVOFPA(1,1)=0.0
      NOVOFPA(1,2)=FPA(1,2)
      NOVOFPA(1,3)=FPA(1,2)+ELE+ERRE

      NOVOFPA(FUNCOES,1)=FPA(FUNCOES,2)-ELE-ERRE
      NOVOFPA(FUNCOES,2)=FPA(FUNCOES,2)
      NOVOFPA(FUNCOES,3)=0.0


      DO I=2, FUNCOES-1
      NOVOFPA(I,1)=FPA(I,2)-ELE-ERRE
      NOVOFPA(I,2)=FPA(I,2)
      NOVOFPA(I,3)=FPA(I,2)+ELE+ERRE
      END DO



      DO I=1,FUNCOES
      FPA(I,1)=NOVOFPA(I,1)
      FPA(I,2)=NOVOFPA(I,2)
      FPA(I,3)=NOVOFPA(I,3)
      END DO

C FIM DE FPA


      DO I=1, FUNCOES
      DO J=1, 3
      IF ((FPE(I,J).LT.1E-5) .AND. (FPE(I,J).GT.-1E-5)) THEN
      NOVOFPE(I,J)=0.0
      END IF
      END DO
      END DO

      DO I=1, FUNCOES
      DO J=1, 3
      IF ((FPVE(I,J).LT.1E-5) .AND. (FPVE(I,J).GT.-1E-5)) THEN
      NOVOFPVE(I,J)=0.0
      ENDIF
      END DO
      END DO

      DO I=1, FUNCOES
      DO J=1, 3
      IF ((FPA(I,J).LT.1E-5) .AND. (FPA(I,J).GT.-1E-5)) THEN
      NOVOFPA(I,J)=0.0
      ENDIF
      END DO
      END DO

C     FUZZIFICACAO DA RAMPA ESQUERDA PARA O ERRO

      IF (ERRO .LT. FPE(1,2)) PE(1,3)=1.0
      IF (ERRO .GT. FPE(1,3)) PE(1,3)=0.0
```

```
      IF ((ERRO .GE. FPE(1,2)) .AND. (ERRO .LE. FPE(1,3))) THEN
      PE(1,3)=1-((ERRO-FPE(1,2))/(FPE(1,3)-FPE(1,2)))
      END IF

C     FUZZIFICACAO DAS FUNCOES DE PERTINENCIA DO ERRO CENTRAIS

      DO I=2,FUNCOES-1
      IF ((ERRO.GE.FPE(I,1)).AND.(ERRO .LE. FPE(I,2))) THEN
      PE(I,3)=1-((FPE(I,2)-ERRO)/(FPE(I,2)-FPE(I,1)))
      ENDIF

      IF ((ERRO .GE. FPE(I,2)) .AND. (ERRO .LE. FPE(I,3))) THEN
      PE(I,3)=1-((ERRO-FPE(I,2))/(FPE(I,3)-FPE(I,2)))
      ENDIF

      IF (ERRO .LT. FPE(I,1)) PE(I,3)=0.0
      IF (ERRO .GT. FPE(I,3)) PE(I,3)=0.0
      END DO

C     FUZZIFICACAO DA RAMPA DIREITA

      IF (ERRO .GT. FPE(FUNCOES,2)) PE(FUNCOES,3)=1.0
      IF (ERRO .LT. FPE(FUNCOES,1)) PE(FUNCOES,3)=0.0

      IF ((ERRO.GE.FPE(FUNCOES,1)).AND.(ERRO.LE.FPE(FUNCOES,2))) THEN
      AUX=((FPE(FUNCOES,2)-ERRO)/(FPE(FUNCOES,2)-FPE(FUNCOES,1)))
      PE(FUNCOES,3)=1-AUX
      END IF



C FUZZIFICACAO DA VARIACAO DO ERRO

C FUZZIFICACAO DA RAMPA ESQUERDA

      IF (VARERRO .LT. FPVE(1,2)) PVE(1,3)=1.0
      IF (VARERRO .GT. FPVE(1,3)) PVE(1,3)=0.0

      IF ((VARERRO .GE. FPVE(1,2)) .AND. (VARERRO .LE. FPVE(1,3))) THEN
      PVE(1,3)=1-((VARERRO-FPVE(1,2))/(FPVE(1,3)-FPVE(1,2)))
      END IF

C     FUZZIFICACAO DAS FUNCOES DE PERTINENCIA DA VARIACAO DO ERRO



      DO I=2,FUNCOES-1
      IF ((VARERRO.GE.FPVE(I,1)).AND.(VARERRO .LE. FPVE(I,2))) THEN
      PVE(I,3)=1-((FPVE(I,2)-VARERRO)/(FPVE(I,2)-FPVE(I,1)))
      ENDIF

      IF ((VARERRO .GE. FPVE(I,2)) .AND. (VARERRO .LE. FPVE(I,3))) THEN
      PVE(I,3)=1-((VARERRO-FPVE(I,2))/(FPVE(I,3)-FPVE(I,2)))
      ENDIF

      IF (VARERRO .LT. FPVE(I,1)) PVE(I,3)=0.0
```

```
      IF (VARERRO .GT. FPVE(I,3)) PVE(I,3)=0.0
      END DO


C  FUZZIFICACAO DA RAMPA DIREITA
      IF (VARERRO .GT. FPVE(FUNCOES,2)) PVE(FUNCOES,3)=1.0
      IF (VARERRO .LT. FPVE(FUNCOES,1)) PVE(FUNCOES,3)=0.0

      IF (VARERRO.GE.FPVE(FUNCOES,1)) THEN
      IF (VARERRO.LE.FPVE(FUNCOES,2)) THEN
      AUX=((FPVE(FUNCOES,2)-VARERRO)/(FPVE(FUNCOES,2)-FPVE(FUNCOES,1)))
      PVE(FUNCOES,3)=1-AUX
      END IF
      END IF

C     INICIO DO CODIGO DO MOTOR DE INFERENCIA FUZZY
      AREA=0
      ATOT=0
      WATOT=0
      SOMA=0
      DO L=1, FUNCOES
      DO CC=1, FUNCOES
      IF (PE(L,3) .NE. 0.0) THEN
      IF (PVE(CC,3) .NE. 0.0) THEN

      MENOR=PE(L,3)
      IF (MENOR .GT. PVE(CC,3)) MENOR=PVE(CC,3)
      VA=BR(CC,L)
      GRAU=MENOR

      IF ((L .NE. FUNCOES).AND. (L.NE.1.0)) BASEMAIOR=FPA(L,3)-FPA(L,1)
      IF (L .EQ. FUNCOES) BASEMAIOR=FPA(L,2)-FPA(L,1)
      IF (L .EQ. 1.0) BASEMAIOR=FPA(L,3)-FPA(L,2)

      IF (L .NE. FUNCOES) THEN
      IF (L .NE.1.0) THEN
      BASEMENOR=(1-GRAU)*(FPA(L,3)-FPA(L,1))
      ENDIF
      ENDIF
      IF (L .EQ. FUNCOES) BASEMENOR=(1-GRAU)*(FPA(L,2)-FPA(L,1))
      IF (L .EQ. 1.0) BASEMENOR=(1-GRAU)*(FPA(L,3)-FPA(L,2))
      PESO=FPA(VA,2)
      AREA=(BASEMAIOR+BASEMENOR)*(GRAU/2)
      ATOT=ATOT+AREA
      WATOT=WATOT+AREA*PESO

      END IF
      END IF

      END DO
      END DO

      ACAO=WATOT/ATOT

      RETURN
      END
```

ANEXO D:

Listagem do Programa Controlador Nebuloso Utilizado Experimentalmente

```
!
program BATCHDIS
!
 use msflib
 use portlib
!
 implicit none
!
 include 'batchdis.fi'
!
 character(50)      fparam(5)
 logical(4)         corte,test
 integer        i,na,nat,cna,ced,nc,nt,nv, &
           & ia,np,iparam(9)
 parameter      (nc=8,nt=8,nv=8,ia=20)
 real(8)            t,dt,tD,tL,Ta,rparam(9),R, &
                                & yss,uss,Te(nt), &
                                & cp(nc,nt),sp(nc),y(ia), &
                                & u(ia),erro(ia),par(nc,nv)
!
 real (8) fuzzyparam(10),louco,parou,Texp(2),filtro
 real (8) Rreal
 real(8)        Ranterior
 integer tipo,funcoes,binario,fps
 real (8) la, le, lve, l2ve, ysp, py(20)
 integer faixa

 call mkwindows()
 call inputdata(fuzzyparam,iparam,rparam,fparam,nc,nv,par, &
        & sp,test)

 np=iparam(4)
 Ta=rparam(1)
 na=1
 nat=0
 cna=1
 ced=1
 dt=0.d0
 tD=0.d0
 tL=0.d0
 R=0.01
 Ranterior=R
 y=0.d0
 u=0.d0
 uss=0.d0
 yss=0.d0
 erro=0.d0
 faixa=0.0
 call initgraph(1)
 call selectcanal(128)
```

```
write(10,*) 't   ','na   ','Tt   ','Tf   ','yt1   ','yt2   ', 'yt3   ','R   '
write(90,*) 't   ','Ta   ','tD   ','tL   ','dt   ','R   '
i=0
do while(.true.)
! ativar linha abaixo qdo em teste
! exit
  call tread(i,nt,iparam(2),Te,test)
        write (10,*) (Te(1))
        write (10,*) (Te(2))
        call graphic(1,nat,ced,nc,nt,nv,np,iparam, &
             & par,cp,sp,Te,R)
        if(i.le.10) then
          call selectcanal(0)
        end if
        if(.not.test) then
          call sleepqq(int(Ta*500.))
        end if
  call selectcanal(128)
        if(.not.test) then
          call sleepqq(int(Ta*500.))
        end if
        write(10,'(I4,2X,2(F12.4,2X))') i,Te(1),Te(2)
        write(*,'(I4,2X,2(F12.4,2X))') i,Te(1),Te(2)
        if(Te(1).gt.64.d0) exit
        i=i+1
 enddo
 call clearscreen($GCLEARSCREEN)
 call initgraph(2)
 t=timef()
 do while(.not.corte)
        call selectcanal(128)
        do while(.true.)
          if((t.ge.(nat*Ta)).or.test) then
            Ranterior=R
                binario=fuzzyparam(1)
                fps=fuzzyparam(3)    ! FPS 1=T, 2=G,
     filtro=fuzzyparam(9)
! Parametros do Controlador Fuzzy
                la=fuzzyparam(4)
                le=fuzzyparam(5)
                lve=fuzzyparam(6)
                l2ve=fuzzyparam(7)
                funcoes=fuzzyparam(2)
                ysp=fuzzyparam(8)
                sp(1)=ysp
                if (iparam(5) .eq. 1) tipo=4
                if (iparam(5) .eq. 2) tipo=3
                if (iparam(5) .eq. 3) tipo=1
                if (iparam(5) .eq. 4) tipo=2

 ! Chamada da rotina de controle usando FPS TRIANGULARES igualmente espacadas
                if (FPS .eq. 1) call
confuzzy1(fps,binario,tipo,funcoes,la,le,lve,l2ve,Te,py,erro,u,ysp,nt,iparam,test,ced,nc,cp,R,louco,filtro)
! Chamada da rotina de controle usando FPS GAUSSIANAS igualmente espacadas
                if (FPS .eq. 2) call
confuzzy2(fps,binario,tipo,funcoes,la,le,lve,l2ve,Te,py,erro,u,ysp,nt,iparam,test,ced,nc,cp,R,louco,filtro)
```

```
                    if (binario.eq. 1) then
                    cp(ced,1)=py(1)
                         cp(ced,2)=py(2)
                    end if
                    call sleepqq(550)
! Defuzzificacao da Acao
! Defuzzificacao P e PD
 if (fps.eq.2) then
                    if ((tipo.eq.1) .or. (tipo.eq.2))  then
                      if (u(1) .lt. 0.0) u(1)=0.0
                      if (u(1) .gt. 0.0) u(1)=u(1)*LA
                    end if
 endif
! Defuzzificacao PI e PID
  if (fps.eq.2) then
                    if ((tipo.eq.3) .or. (tipo.eq.4))  u(1)=u(1)*LA
  endif
                    if (tipo .eq. 1) R=u(1)
                    if (tipo .eq. 3) R=Ranterior+u(1)
                    if (tipo .eq. 2) R=u(1)
                    if (tipo .eq. 4) R=Ranterior+u(1)
                    if(R.le.0.01) R=0.01
      if(R.gt.100.0) R=100.0
      exit
           else
            t=timef()
           end if
          enddo
          call graphic(2,nat,ced,nc,nt,nv,np,iparam, &
                & par,cp,sp,Te,R)
          nat=nat+1
          call valve(t,Ta,tD,tL,dt,R,test)

          if (binario .eq. 1) then
   write(10,'(F14.4,2X,I4,2X,F14.4,2X,F14.4,2X,F14.4,2X,F14.4,2X,F14.4)') t,nat,Te(1),
 Te(2),cp(ced,1),cp(ced,2),R
          end if

          write(90,'(6(F14.4,2X))') t,Ta,tD,tL,dt,R
          !if((dabs(sp(ced)-cp(ced,1)).gt.0.1d0).and. &
          !& (sp(ced).gt.cp(ced,1))) corte=.true.
          t=timef()

  enddo
  write(10,'(27H Tempo total da batelada = , &
        & F12.4,3H s.)') t
  write(10,'(19H Arquivo de dados: ,A80)') fparam(1)
  close(10)
  close(90)
  if(test) close(15)
  stop 'Fim do Programa'
 end program

 /*************************************************************************\
```

```
!
subroutine inputdata(fuzzyparam,iparam,rparam,fparam,nc,nv, &
                     & par,sp,test)
!
 use msflib
!
 implicit none
!
 character(50)    fparam(5)
 integer          nc,nv,iparam(9)
 real (8)         fuzzyparam(10)
 real(8)          rparam(9),par(nc,nv),sp(nc)
 logical(4)       test
!
 character(1)     key
 integer(2)       i,mode,opcao
 integer(4)       stt
 type(windowconfig) screen
!
 stt=focusqq(13)
 write(13,'(56H INÍCIO DE OPERAÇÃO DA COLUNA DE DESTILAÇÃO EM BATELADA.)')
 write(13,'(35H NOME COMPLETO DO ARQUIVO DE SAIDA:)')
 read(13,'(A50)') fparam(2)
 open(UNIT=10,FILE=fparam(2),STATUS='UNKNOWN')
 write(13,*)('NOME COMPLETO DO ARQUIVO DOS TEMPOS:')
 read(13,*) fparam(1)
 open(UNIT=90,FILE=fparam(1),STATUS='UNKNOWN')
          write(13,*)('DIGITE EM ORDEM (NC NT ):')
          read(13,*) (iparam(i),i=1,2)
          write(13,*)('TIPO DE CONTROLADOR FUZZY (PID=1,PI=2,P=3,PD=4):')
          read(13,*) iparam(5)
          fuzzyparam(1)=1
          write(13,*)('Cardinalidade (3 a 15 ) :')
          read(13,*) fuzzyparam(2)
          write(13,*)('Tipo de Funcoes de Pertinencia (1=Triangular, 2=Gaussiana:')
          read(13,*) fuzzyparam(3)
          write(13,*)('Limite da Acao, valor de 2 a 10 (P e PD - 0 a LA, PI,PID - -LA a +La')
          read(13,*) fuzzyparam(4)
          write(13,*)('Limite do Erro , valor de 0.001 a 0.1')
          read(13,*) fuzzyparam(5)
          if ((iparam(5).eq.1) .or. (iparam(5).eq.2) .or.(iparam(5).eq.4)) then
          write(13,*)('Limite da Variacao do Erro , valor de 0.002 a 0.2')
          read(13,*) fuzzyparam(6)
          endif
          if (iparam(5).eq.1) then
          write(13,*)('Limite da 2a. Variacao do Erro , valor de 0.004 a 0.4')
          read(13,*) fuzzyparam(7)
          endif
          write(13,*)('Concentracao no setpoint do componente mais volatil ')
          read(13,*) fuzzyparam(8)
          write(13,*)('Qual tipo de filtro de ruido ? (1=Media, 2=Exponencial Duplo')
          read(13,*) fuzzyparam(9)
          iparam(4)=40
          write(13,* )('Tempo de amostragem (TA):')
          read(13,*) rparam(1)
```

```
 call CLEARSCREEN($GCLEARSCREEN)
 write(13,'(A18)') 'DADOS FORNECIDOS:'
 write(13,'(29H NÚMERO DE TERMOPARES     = ,I2)') iparam(2)
 write(13,'(29H NÚMERO DE PONTOS NA TELA  = ,I2)') iparam(4)
 write(13,'(29H TIPO DE CONTROLADOR FUZZY = ,I2)')iparam(5)
 write(13,'(29H INTERVALO DE AMOSTRAGEM(s)= ,F7.3)') rparam(1)
 write(13,'(19H ARQUIVO DE SAÍDA: ,A50)') fparam(2)
 write(13,*) 'Arquivo dos tempos:'
 write(13,*) fparam(1)
 mode=getwindowconfig(screen)
 stt=focusqq(12)
 return
end subroutine

/*****************************************************************************\
!    CONTROLADOR FUZZY - MAMDANI
!    PROGRAMA DO CONTROLADOR FUZZY EM FORTRAN
!    RENATO DUTRA PEREIRA FILHO - 19/11/1998
!    CALCULO DA AREA ABAIXO DE GRAU P/ FPS GAUSSIANA

subroutine areagauss(tipo,area,grau,peso,va,funcoes,lia,lsa)

implicit none
real (8) area,grau
integer funcoes,j,n1,n,tipo,i
real (8) va,peso
real (8) u,ureal,x1,x2,x1real,x2real,areadireita,t,fim,yy,teta
real (8) tempo,inicio, h,direita,esquerda,areatotal,areaesquerda
real (8) areadoretangulo,xx,lia,lsa,parou
real (8) xi,f,derivada,xi2,x,coef,aux
u=0

if (grau.ne.0.0) then
ureal=peso
if ((va.eq.1) .or. (va.eq.funcoes)) then
   teta=0.14
        else
        teta=0.14
end if


if (va.eq.1) then
    x1=0
 else
    x1=((2*u)-sqrt((4*u**2)-(4*2*teta**2*log(grau))))/2
endif

if (va.eq.funcoes) then
    x2=0
 else
    x2=((2*u)+sqrt((4*u**2)-(4*2*teta**2*log(grau))))/2
endif

x1real=x1+ureal
```

```
x2real=x2+ureal
! Busca da Raiz Esquerda nos tipos 3 e 4 Usando Metodo de Busca
if ((tipo.eq.3) .or. (tipo.eq.4)) then
esquerda=-20

do xx=x1real,-1,-0.01
 call fgauss(xx,ureal,teta,yy)
 if (yy .le. 0.01) then
   esquerda=xx
          exit
 endif
end do
if (esquerda .eq.-20.0) esquerda=-1.0
endif

! Busca da Raiz Esquerda nos tipos 1 e 2 Usando Metodo de Busca
if ((tipo.eq.1) .or. (tipo.eq.2)) then
esquerda=-20
do xx=x1real,0,-0.01
 call fgauss(xx,ureal,teta,yy)
 if (yy .le. 0.01) then
   esquerda=xx
          exit
 endif
end do
if (esquerda .eq.-20.0) esquerda=0.0
endif

! Calculo da Area esquerda p/ controladores PI e PID usando Simpson
if ((tipo.eq.3) .or. (tipo.eq.4)) then
 t=0
 if (x1real.gt.-1) then
   inicio=esquerda
          fim=x1real
          n=21
          n1=n-1
          h=(fim-inicio)/n1
          x=inicio
          coef=2
          aux=-2
          call fgauss(x,ureal,teta,yy)
          t=yy
          x=fim
          call fgauss(x,ureal,teta,yy)
          t=t+yy
          do i=1, n1
            x=x+h
                  aux=-aux
                  coef=coef+aux
                  call fgauss(x,ureal,teta,yy)
                  t=t+coef*yy
          end do
          t=t*h/3
  endif
  areaesquerda=t
 endif
```

```
! Calculo da Area direita p/ controladores P e PD usando Simpson
if ((tipo.eq.1) .or. (tipo.eq.2)) then
 t=0
 if (x1real.gt.0.0) then
    inicio=esquerda
           fim=x1real
           n=21
           n1=n-1
           h=(fim-inicio)/n1
           x=inicio
           coef=2
           aux=-2
           call fgauss(x,ureal,teta,yy)
           t=yy
           x=fim
           call fgauss(x,ureal,teta,yy)
           t=t+yy
           do i=1, n1
             x=x+h
                   aux=-aux
                   coef=coef+aux
                   call fgauss(x,ureal,teta,yy)
                   t=t+coef*yy
           end do
           t=t*h/3
  endif
  areaesquerda=t
 endif

! Calculo da Raiz Direita Usando Metodo de Busca
direita=20
do xx=x2real,1,0.01
  call fgauss(xx,ureal,teta,yy)
  if (yy .le. 0.01) then
    direita=xx
           exit
  endif
end do
if (direita .eq.20.0) direita=1

!Calculo da Area Direita Usando Trapezoidal
!t=0
!if (x2real .lt. 1) then
!  inicio=x2real
!  fim=direita
!  n=100
!  call fgauss(inicio,ureal,teta,yy)
!  t=yy/2
!  n1=n-1
!  h=(fim-inicio)/n
!  do j=1,n1
!    tempo=inicio+j*h
!          call fgauss(tempo,ureal,teta,yy)
!          t=yy+t
```

```
! end do
! call fgauss(fim,ureal,teta,yy)
! t=(yy/2+t)*abs(h)
!endif
!areadireita=t

! Calculo da Area direita p/ controladores P, PD,PI e PID  usando Simpson

 t=0
 if (x2real.lt.1.0) then
    inicio=x2real
           fim=direita
           n=21
           n1=n-1
           h=(fim-inicio)/n1
           x=inicio
           coef=2
           aux=-2
           call fgauss(x,ureal,teta,yy)
           t=yy
           x=fim
           call fgauss(x,ureal,teta,yy)
           t=t+yy
           do i=1, n1
             x=x+h
                   aux=-aux
                   coef=coef+aux
                   call fgauss(x,ureal,teta,yy)
                   t=t+coef*yy
           enddo
           t=t*h/3
  endif
  areadireita=t

if ((tipo.eq.3).or.(tipo.eq.4)) then
    if (x1real.lt.-1) x1real=-1
endif
if ((tipo.eq.1).or.(tipo.eq.2)) then
    if (x1real.lt.0) x1real=0
endif


if (x2real.gt.1) x2real=1
if (grau.lt.1.0) then
     areadoretangulo=grau*abs(x2real-x1real)
           else
           areadoretangulo=0
end if

areatotal=areaesquerda+areadoretangulo+areadireita


!if (va.eq.funcoes) then
!write(*,*) areaesquerda
!write(*,*) areadoretangulo
!write(*,*) areadireita
```

```
!write(*,*) '-------------------------'
!write(*,*) x1real,x2real
!write(*,*) funcoes,va
!write(*,*) ureal,grau
!write(*,*) x1,x2
!read(*,*) parou
!endif
area=areatotal

else
  area=0.0

endif

!if (area.lt.0.0) then
!   write(*,*) '*******'
!          write(*,*) area
!          write(*,*) '*******'
!end if



end subroutine
/*****************************************************************************\

! RENATO DUTRA PEREIRA FILHO - 19/11/1998
! MODULO GERADOR DA MATRIZ BASE_DE_REGRAS
!
subroutine basederegras(tipo,br,funcoes)
implicit none
integer tipo,funcoes,br(15,15,15)
integer valor,x,y,z

if (tipo .eq. 1) then
  do x=1, funcoes
    br(x,1,1)=x
  end do
end if

if ((tipo .eq. 2) .or. (tipo .eq. 3)) then
   do x=1, funcoes
          do y=1, funcoes
            valor=x+y-((funcoes+1)/2)
            if (valor .lt. 1.0) valor=1.0
            if (valor .gt. funcoes) valor=funcoes
            br(y,x,1)=valor
          end do
          end do
end if

if (tipo .eq. 4) then
   do x=1, funcoes
          do y=1, funcoes
          do z=1, funcoes
            valor=x+y+z-(funcoes+1)
            if (valor .lt. 1.0) valor=1.0
```

```
            if (valor .gt. funcoes) valor=funcoes
            br(y,x,z)=valor

          end do
          end do
          end do
end if

end subroutine
/*****************************************************************************\

!    CONTROLADOR FUZZY - MAMDANI
!    RENATO DUTRA PEREIRA FILHO - 19/11/1998
!    DEFINICAO DOS CENTROS DAS FPS TRIANGULARES
!    IGUALMENTE ESPACADAS

subroutine centros(funcoes,pe,re,tipo,rve,pve,p2ve,r2ve,pa,lie,live,li2ve,lia,lse,lsve,ls2ve,lsa)
implicit none
real(8) pe(15,3),re,rve,pve(15,3),p2ve(15,3),r2ve,pa(15,3)
integer funcoes,tipo, i
real (8) lie,live,li2ve,lia,lse,lsve,ls2ve,lsa,parou


do i=2, funcoes-1
  pe(i,2)=re/((funcoes-1))*(i-1)+lie
  if ((tipo .eq.2) .or. (tipo.eq.3) .or. (tipo.eq.4)) pve(i,2)=rve/((funcoes-1))*(i-1)+live
  if (tipo .eq.4) p2ve(i,2)=r2ve/((funcoes-1))*(i-1)+li2ve
  pa(i,2)=(pa(i,1)/2)*(i-1)+lia

end do

pe(1,2)=lie
pve(1,2)=live
p2ve(1,2)=li2ve
pa(1,2)=lia
pe(funcoes,2)=lse
pve(funcoes,2)=lsve
p2ve(funcoes,2)=ls2ve
pa(funcoes,2)=lsa


!do i=1, funcoes
!  write(*,*) pe(i,2),pve(i,2),p2ve(i,2)
!end do
!write (*,*) '-------------------------------'
!read(*,*) parou



end subroutine

/*****************************************************************************\

!    CONTROLADOR FUZZY - MAMDANI
!    PROGRAMA DO CONTROLADOR FUZZY EM FORTRAN
```

```
!    RENATO DUTRA PEREIRA FILHO - 19/11/1998
!
!
subroutine confuzzy2(fps,binario,tipo,funcoes,la,le,lve,l2ve,Te,py,erro,u,ysp,nt,iparam,test,ced,nc,cp,R,louco,filtro)
implicit none
include 'cfiles.fi'
!ms$attributes C,EXTERN :: offset
character(1)  offset

logical(4) test
real(8) Te(8),py(20),erro(20),u(20),R

integer tipo,funcoes,nc,nt,ced,iparam(9),binario
real(8) la,le,lve,l2ve,ysp,live,lsve,li2ve,ls2ve
real(8) acao,lia,lsa,re,rve,r2ve,ra,lse,lie
real(8) pa(15,3),p2ve(15,3),pve(15,3),pe(15,3),cp(nc,nt),louco
integer br(15,15,15),fps
real(8) leitexp(2),a,vfa1,vfa2,texp(2),vfb1,vfb2
real(8) leitura(2),t(2),filtro
integer j
save vfa1,vfa2,vfb1,vfb2

a=0.33

call basederegras(tipo,br,funcoes)
if (binario.eq.1.0) call infecompo(Te,py,filtro)
if (binario.eq.0.0) then
   !louco=louco+1
        !te(1)=68+(louco/50)-(R*R/50)
        !te(2)=78+(louco/100)-(R*R/150)
        !louco=louco-(R/10)
   if (filtro.eq.2.0) then
          call tread(2,nt,iparam(2),Te,test)
   endif


   if (filtro.eq.1.0) then
        ! Colocar em comentario p/ testar
        !LEITURAS DOS TERMOPARES, MEDIAS, INFERENCIAS, ETC
        call selectcanal(128+2)
        call sleepqq(250)
        do j=1,30000
        leitura(1)=leitura(1)+read_anl('0',offset)
        enddo
        leitura(1)=leitura(1)/30000.0
        !Linha abaixo adicionada para comparar filtro exponencial duplo

!       leitexp(1)=leitura(1)*(a**2.d0)+(1.0d0-a)*2.d0*vfa1-((1.d0-a)**2.d0)*vfa2
!               vfa2=vfa1
!               vfa1=leitexp(1)

   !temperatura no topo calculada usando filtro de media
        T(1)=147.0*(leitura(1)-1050.0)/(4220.0-1050.0)
        Te(1)=T(1)
        !
        !temperatura no topo calculada usando filtro exponencial duplo
```

```
!  Texp(1)=147.0*(leitexp(1)-1050.0)/(4220.0-1050.0)

          call selectcanal(128+1)
          call sleepqq(250)
          do j=1,30000
          leitura(2)=leitura(2)+read_anl('0',offset)
          enddo
          leitura(2)=leitura(2)/30000.0

          !Linha abaixo adicionada para comparar filtro exponencial duplo

!         leitexp(2)=leitura(2)*(a**2.d0)+(1.0d0-a)*2.d0*vfb1-((1.d0-a)**2.d0)*vfb2
!         vfb2=vfb1
!         vfb1=leitexp(2)

          !temperatura no fundo calculada usando filtro de media
          T(2)=147.0*(leitura(2)-1050.0)/(4220.0-1050.0)
          Te(2)=T(2)
          !

          !temperatura no fundo calculada usando filtro exponencial duplo
 !  Texp(2)=147.0*(leitexp(2)-1050.0)/(4220.0-1050.0)

    endif



          call yinfer(ced,nc,nt,Te,cp,R)
          py(1)=cp(1,1)
          py(2)=cp(2,1)
          py(3)=cp(3,1)
end if
call limites(tipo,fps,ysp,py,erro,le,lve,l2ve,la,lie,lse,live,lsve,li2ve,ls2ve,lia,lsa,re,rve,r2ve,ra)
call largura(funcoes,pe,re,tipo,rve,pve,p2ve,r2ve,pa,ra)
call centros(funcoes,pe,re,tipo,rve,pve,p2ve,r2ve,pa,lie,live,li2ve,lia,lse,lsve,ls2ve,lsa)
call fuzzificagauss(tipo,erro,pe,funcoes,pve,p2ve)
call infefuzzygauss(funcoes,tipo,pe,pve,p2ve,br,acao,u,pa,lia,lsa)

u(1)=acao

end subroutine

/*************************************************************************\


!    CONTROLADOR FUZZY - MAMDANI
!    PROGRAMA DO CONTROLADOR FUZZY EM FORTRAN
!    RENATO DUTRA PEREIRA FILHO - 19/11/1998
!    DIMENSIONAMENTO DAS MATRIZES
!
subroutine confuzzy1(fps,binario,tipo,funcoes,la,le,lve,l2ve,Te,py,erro,u,ysp,nt,iparam,test,ced,nc,cp,R,louco,filtro)
implicit none
include 'cfiles.fi'
!ms$attributes C,EXTERN :: offset
character(1)  offset
logical (4) test
```

```
real(8) Te(8),py(20),erro(20),u(20),R
integer tipo,funcoes,nc,nt,ced,iparam(9),binario
real(8) la,le,lve,l2ve,ysp,live,lsve,li2ve,ls2ve
real(8) acao,lia,lsa,re,rve,r2ve,ra,lse,lie
real(8) pa(15,3),p2ve(15,3),pve(15,3),pe(15,3),cp(nc,nt)
real(8) leitexp(2),a,vfa1,vfa2,texp(2),vfb1,vfb2
integer br(15,15,15),fps
real (8) louco,parou,filtro
real(8) leitura(2),t(2)
integer j
save vfa1,vfa2,vfb1,vfb2
a=0.33
call basederegras(tipo,br,funcoes)
if (binario.eq.1.0) call infecompo(Te,py,filtro)
if (binario.eq.0.0) then
        !louco=louco+1
        !te(1)=68+(louco/50)-(R*R/50)
        !te(2)=78+(louco/100)-(R*R/150)
        !write(*,*) te(1),te(2),R
        !write(*,*) ced,nc,nt
        !write(*,*) '--------------------'

        if (filtro.eq.2.0) then
                call tread(2,nt,iparam(2),Te,test)
   endif


        if (filtro.eq.1.0) then
        ! Colocar em comentario p/ testar
        !LEITURAS DOS TERMOPARES, MEDIAS, INFERENCIAS, ETC
        call selectcanal(128+2)
        call sleepqq(250)
        do j=1,30000
        leitura(1)=leitura(1)+read_anl('0',offset)
        enddo
        leitura(1)=leitura(1)/30000.0
        !Linha abaixo adicionada para comparar filtro exponencial duplo
!       leitexp(1)=leitura(1)*(a**2.d0)+(1.0d0-a)*2.d0*vfa1-((1.d0-a)**2.d0)*vfa2
!               vfa2=vfa1
!               vfa1=leitexp(1)

   !temperatura no topo calculada usando filtro de media
        T(1)=150.0*(leitura(1)-1050.0)/(4220.0-1050.0)
        Te(1)=T(1)
        !
        !temperatura no topo calculada usando filtro exponencial duplo
!  Texp(1)=150.0*(leitexp(1)-1050.0)/(4220.0-1050.0)

        call selectcanal(128+1)
        call sleepqq(250)
        do j=1,30000
        leitura(2)=leitura(2)+read_anl('0',offset)
        enddo
        leitura(2)=leitura(2)/30000.0

        !Linha abaixo adicionada para comparar filtro exponencial duplo
```

```
!         leitexp(2)=leitura(2)*(a**2.d0)+(1.0d0-a)*2.d0*vfb1-((1.d0-a)**2.d0)*vfb2
!         vfb2=vfb1
!         vfb1=leitexp(2)

          !temperatura no fundo calculada usando filtro de media
          T(2)=150.0*(leitura(2)-1050.0)/(4220.0-1050.0)
          Te(2)=T(2)
          !

          !temperatura no fundo calculada usando filtro exponencial duplo
!   Texp(2)=150.0*(leitexp(2)-1050.0)/(4220.0-1050.0)

   endif

          call yinfer(ced,nc,nt,Te,cp,R)
          py(1)=cp(1,1)
          py(2)=cp(2,1)
          py(3)=cp(3,1)
          !write (*,*) py(1),py(2),py(3)
          !read(*,*) parou
end if




call limites(tipo,fps,ysp,py,erro,le,lve,l2ve,la,lie,lse,live,lsve,li2ve,ls2ve,lia,lsa,re,rve,r2ve,ra)
call largura(funcoes,pe,re,tipo,rve,pve,p2ve,r2ve,pa,ra)
call centros(funcoes,pe,re,tipo,rve,pve,p2ve,r2ve,pa,lie,live,li2ve,lia,lse,lsve,ls2ve,lsa)
call fuzzifica(tipo,erro,pe,funcoes,pve,p2ve)
call infefuzzy(funcoes,tipo,pe,pve,p2ve,br,acao,u,pa)
u(1)=acao

end subroutine

/*************************************************************************\
!   CONTROLADOR FUZZY - MAMDANI
!   PROGRAMA DO CONTROLADOR FUZZY EM FORTRAN
!   RENATO DUTRA PEREIRA FILHO - 19/11/1998
!   CALCULO DA Derivada da FP GAUSSIANA
subroutine dfgauss(x,u,teta,yy)
real (8) x,u,teta,yy,aux1,aux2
aux1=exp(-(x-u)**2/(2*teta**2))
aux2=(x-u)/(teta**2)
yy=aux2*aux1
end subroutine
/*************************************************************************\
!   CONTROLADOR FUZZY - MAMDANI
!   PROGRAMA DO CONTROLADOR FUZZY EM FORTRAN
!   RENATO DUTRA PEREIRA FILHO - 19/11/1998
!   CALCULO DA FP GAUSSIANA
subroutine fgauss(x,u,teta,yy)
real (8) x,u,teta,yy
yy=exp(-(x-u)**2/(2*teta**2))
end subroutine
```

```
/*************************************************************************\
!    CONTROLADOR FUZZY - MAMDANI
!    PROGRAMA DO CONTROLADOR FUZZY EM FORTRAN
!    RENATO DUTRA PEREIRA FILHO - 19/11/1998
!    FUZZIFICACAO

subroutine fuzzifica(tipo,erro,pe,funcoes,pve,p2ve)

real (8) erro(20),pe(15,3),pve(15,3),p2ve(15,3),parouu
integer funcoes,i,tipo

! fuzzificacao do erro

if (erro(1) .le. pe(1,2)) pe(1,3)=1.0
if (erro(1) .ge. pe(1,2)) pe(1,3)=1-(2*(erro(1)-pe(1,2)))/pe(1,1)
if (pe(1,3) .le. 0.0) pe(1,3)=0.0

do i=2, funcoes-1
  if (erro(1) .ge. pe(i,2)) pe(i,3)=1-(2*(erro(1)-pe(i,2)))/pe(i,1)
  if (erro(1) .le. pe(i,2)) pe(i,3)=1-(2*(pe(i,2)-erro(1)))/pe(i,1)

  if (pe(i,3) .le. 0.0) pe(i,3)=0.0

end do

if (erro(1) .ge. pe(funcoes,2)) pe(funcoes,3)=1.0
if (erro(1) .le. pe(funcoes,2)) pe(funcoes,3)=1-(2*(pe(funcoes,2)-erro(1)))/pe(funcoes,1)
if (pe(funcoes,3) .le. 0.0) pe(funcoes,3)=0.0



! fuzzificacao da variacao do erro
if ((tipo.eq.2) .or. (tipo.eq.3) .or. (tipo.eq.4)) then
  if (erro(3) .le. pve(1,2)) pve(1,3)=1.0
  if (erro(3) .ge. pve(1,2)) pve(1,3)=1-(2*(erro(3)-pve(1,2)))/pve(1,1)
  if (pve(1,3) .le. 0.0) pve(1,3)=0.0
  if (pve(1,3) .ge. 1.0) pve(1,3)=1.0


  do i=2, funcoes-1
    if (erro(3) .ge. pve(i,2)) pve(i,3)=1-(2*(erro(3)-pve(i,2)))/pve(i,1)
    if (erro(3) .le. pve(i,2)) pve(i,3)=1-(2*(pve(i,2)-erro(3)))/pve(i,1)
          if (pve(i,3) .le. 0.0) pve(i,3)=0.0
          if (pve(i,3) .ge. 1.0) pve(i,3)=1.0
  end do

  if (erro(3) .ge. pve(funcoes,2)) pve(funcoes,3)=1.0
  if (erro(3) .le. pve(funcoes,2)) pve(funcoes,3)=1-(2*(pve(funcoes,2)-erro(3)))/pve(funcoes,1)
  if (pve(funcoes,3) .lt. 0.0) pve(funcoes,3)=0.0

 end if
 ! fuzzificacao da 2a. variacao do erro
 if (tipo.eq.4) then
  if (erro(5) .le. p2ve(1,2)) p2ve(1,3)=1.0
  if (erro(5) .ge. p2ve(1,2)) p2ve(1,3)=1-(2*(erro(5)-p2ve(1,2)))/p2ve(1,1)
  if (p2ve(1,3) .le. 0.0) p2ve(1,3)=0.0
```

```
if (p2ve(1,3) .ge. 1.0) p2ve(1,3)=1.0


do i=2, funcoes-1
  if (erro(5) .ge. p2ve(i,2)) p2ve(i,3)=1-(2*(erro(5)-p2ve(i,2)))/p2ve(i,1)
  if (erro(5) .le. p2ve(i,2)) p2ve(i,3)=1-(2*(p2ve(i,2)-erro(5)))/p2ve(i,1)
        if (p2ve(i,3) .le. 0.0) p2ve(i,3)=0.0
        if (p2ve(i,3) .ge. 1.0) p2ve(i,3)=1.0
end do

if (erro(5) .ge. p2ve(funcoes,2)) p2ve(funcoes,3)=1.0
if (erro(5) .le. p2ve(funcoes,2)) p2ve(funcoes,3)=1-(2*(p2ve(funcoes,2)-erro(5)))/p2ve(funcoes,1)
if (p2ve(funcoes,3) .le. 0.0) p2ve(funcoes,3)=0.0
if (p2ve(funcoes,3) .ge. 1.0) p2ve(funcoes,3)=1.0

!do i=1,funcoes
! write(*,*) pe(i,3),pve(i,3),p2ve(i,3)
!end do
!write(*,*) '***-***'
!read(*,*) parouu

end if

end subroutine
/*****************************************************************************\

!   CONTROLADOR FUZZY - MAMDANI
!   PROGRAMA DO CONTROLADOR FUZZY EM FORTRAN
!   RENATO DUTRA PEREIRA FILHO - 19/11/1998
!   FUZZIFICACAO USANDO FUNCAO DE GAUSS

subroutine fuzzificagauss(tipo,erro,pe,funcoes,pve,p2ve)

real (8) erro(20),pe(15,3),pve(15,3),p2ve(15,3),sigma1,sigma2,parou
integer funcoes,i,tipo

! Esses sao os parametros das gaussianas, 1 p/ os extremos e
! 2 p/ os intermediarios

sigma1=0.14
sigma2=0.14

! fuzzificacao do erro
if (erro(1) .le. pe(1,2)) pe(1,3)=1.0
if (erro(1) .gt. pe(1,2)) pe(1,3)=exp(-(erro(1)-pe(1,2))**2/(2*sigma1**2))
if (pe(1,3) .lt. 0.0) pe(1,3)=0.0
if (pe(1,3) .gt. 1.0) pe(1,3)=1.0

do i=2, funcoes-1
  pe(i,3)=exp(-(erro(1)-pe(i,2))**2/(2*sigma2**2))
  if (pe(i,3) .lt. 0.0) pe(i,3)=0.0
  if (pe(i,3) .gt. 1.0) pe(i,3)=1.0
end do
if (erro(1) .ge. pe(funcoes,2)) pe(funcoes,3)=1.0
if (erro(1) .lt. pe(funcoes,2)) pe(funcoes,3)=exp(-(erro(1)-pe(funcoes,2))**2/(2*sigma1**2))
if (pe(funcoes,3) .lt. 0.0) pe(funcoes,3)=0.0
```

```
if (pe(funcoes,3) .gt. 1.0) pe(funcoes,3)=1.0

! fuzzificacao da variacao do erro
if ((tipo.eq.2) .or. (tipo.eq.3) .or. (tipo.eq.4)) then
 if (erro(3) .le. pve(1,2)) pve(1,3)=1.0
 if (erro(3) .gt. pve(1,2)) pve(1,3)=exp(-(erro(3)-pve(1,2))**2/(2*sigma1**2))
 if (pve(1,3) .lt. 0.0) pve(1,3)=0.0
 do i=2, funcoes-1
        pve(i,3)=exp(-(erro(3)-pve(i,2))**2/(2*sigma2**2))
        if (pve(i,3) .lt. 0.0) pve(i,3)=0.0
        if (pve(i,3) .gt. 1.0) pve(i,3)=1.0
 end do
if (erro(3) .ge. pve(funcoes,2)) pve(funcoes,3)=1.0
if (erro(3) .lt. pve(funcoes,2)) pve(funcoes,3)=exp(-(erro(3)-pve(funcoes,2))**2/(2*sigma1**2))
if (pve(funcoes,3) .lt. 0.0) pve(funcoes,3)=0.0
if (pve(funcoes,3) .gt. 1.0) pve(funcoes,3)=1.0
end if

! fuzzificacao da 2a. variacao do erro
if (tipo.eq.4) then
 if (erro(5) .le. p2ve(1,2)) p2ve(1,3)=1.0
 if (erro(5) .gt. p2ve(1,2)) p2ve(1,3)=exp(-(erro(5)-p2ve(1,2))**2/(2*sigma1**2))
 if (p2ve(1,3) .lt. 0.0) p2ve(1,3)=0.0
 !if (p2ve(1,3) .gt. 1.0) p2ve(1,3)=1.0

 do i=2, funcoes-1
        p2ve(i,3)=exp(-(erro(5)-p2ve(i,2))**2/(2*sigma2**2))
  if (p2ve(i,3) .lt. 0.0) p2ve(i,3)=0.0
        if (p2ve(i,3) .gt. 1.0) p2ve(i,3)=1.0
 end do

 if (erro(5) .ge. p2ve(funcoes,2)) p2ve(funcoes,3)=1.0
 if (erro(5) .lt. p2ve(funcoes,2)) p2ve(funcoes,3)=exp(-(erro(5)-p2ve(funcoes,2))**2/(2*sigma1**2))
 if (p2ve(funcoes,3) .lt. 0.0) p2ve(funcoes,3)=0.0
 if (p2ve(funcoes,3) .gt. 1.0) p2ve(funcoes,3)=1.0
end if

! limpa lixos nas pertinencias
!do i=1, funcoes
 if (pe(i,3).lt.0.01) pe(i,3)=0.0
 if (pve(i,3).lt.0.01) pve(i,3)=0.0
 if (p2ve(i,3).lt.0.01) p2ve(i,3)=0.0
!end do


end subroutine
/*************************************************************************\
!
subroutine graphic(tipo,na,ced,nc,nt,nv,np,iparam, &
        & par,cp,sp,Te,R)
!
 use msflib
!
 implicit none
!
 integer        tipo,na,ced,nc,nt,nv,np, &
```

```
            & iparam(9)
 real(8)        par(nc,nv),cp(nc,nt),Te(nt), &
            & sp(nc),R
!
 integer(4)      result
 integer(2)      i,j,ct,ix,ixa,iya(3),iy(3),stt
 character(15)    cy
 type(xycoord)    xy
 save                          ixa,iya,ct
!
 result=setactiveqq(12)
 call setgtextrotation(0)
 if(tipo.eq.1) then
  do i=1,iparam(2)
          iy(i)=430-int(Te(i)*4.0d0)
         enddo
 else
  if(na.eq.0) ct=0
  do i=1,iparam(2)
          iy(i)=430-int(cp(ced,i)*400.d0)
         enddo
         iy(3)=430-int(sp(ced)*400.d0)
 end if
 if(ct.eq.0) ix=75
 if(iy(1).le.30) iy(1)=31
 if(iy(2).le.30) iy(2)=31
 if(iy(1).ge.430) iy(1)=429
 if(iy(2).ge.430) iy(2)=429
 if((ct.le.np).and.(ix.lt.485)) then
  ix=ix+10
         if(ct.eq.np) ix=ix-1
         if((ix.eq.85).and.(ct.eq.0)) then
          ixa=ix-9
          iya=iy
         end if
         ct=ct+1
 else
         ct=1
         ix=85
         ixa=ix-9
   stt=setcolor(0)
         stt=rectangle($GFILLINTERIOR,76,31,484,429)
 end if
 stt=setcolor(10)
 call moveto(ixa,iya(1),xy)
 stt=lineto(ix,iy(1))
 stt=setcolor(12)
 call moveto(ixa,iya(2),xy)
 stt=lineto(ix,iy(2))
 ixa=ix
 iya(1:2)=iy(1:2)
!
 if(tipo.eq.2) then
!
! Set Point do comp. que esta sendo retirado.
!
```

```
  stt=setcolor(0)
        call moveto(76,iya(3),xy)
        stt=lineto(484,iya(3))
        if(iy(3).le.30) iy(3)=31
        if(iy(3).ge.430) iy(3)=429
        stt=setcolor(14)
        call moveto(76,iy(3),xy)
        stt=lineto(484,iy(3))
!
! Exibe as barras indicativas da razão de refluxo.
!
  stt=setcolor(0)
  stt=rectangle($GFILLINTERIOR,570,31,578,429)
  stt=rectangle($GFILLINTERIOR,587,31,595,429)
  if(R.lt.10.) then
        iy(4)=int(430-40.0d0*R)
  else
   iy(4)=31
  end if
  if(iy(4).eq.30) iy(4)=31
  if(iy(4).eq.430) iy(4)=429
  stt=setcolor(3)
  stt=rectangle($GFILLINTERIOR,570,iy(4),578,429)
  if(R.le.100.) then
        iy(4)=int(430-4.0d0*R)
  else
   iy(4)=31
  end if
  if(iy(4).eq.30) iy(4)=31
  if(iy(4).eq.430) iy(4)=429
  stt=rectangle($GFILLINTERIOR,587,iy(4),595,429)
        iya(3:4)=iy(3:4)
! Exibe os parametros do modelo (Kp,Tp,td).
!  stt=setcolor(0)
!  stt=rectangle($GFILLINTERIOR,76,405,484,428)
!  stt=setcolor(15)
!  call moveto(133,405,xy)
!  call outgtext('Kp')
!  call moveto(271,405,xy)
!  call outgtext('Tp')
!  call moveto(409,405,xy)
!  call outgtext('td')
!  j=1
!  do i=83,359,138
!   call moveto(i,420,xy)
!   write(cy,'(ES12.4)') par(ced,j)
!   call outgtext(cy)
!        j=j+1
!  enddo
! Exibe o numero de amostragem.
!
   if(na.ne.0) then
    stt=setcolor(0)
    call moveto(580,450,xy)
    write(cy,'(I4)') (na-1)
    call outgtext(cy)
```

```
   end if
   stt=setcolor(14)
   call moveto(580,450,xy)
   write(cy,'(I4)') na
   call outgtext(cy)
  end if
  return
end subroutine

/*****************************************************************************\

!    RENATO DUTRA PEREIRA FILHO - 19/11/1998
!    Modulo de inferencia termodinamica das composicoes
!

subroutine infecompo(Te,py,filtro)
implicit none
include 'cfiles.fi'
!ms$attributes C,EXTERN :: offset
character(1)  offset
real(8) Te(8),py(20),leitura(2),t(2),x,y,filtro
integer j

if (filtro.eq.1.0) then

! Colocar em comentario p/ testar
!LEITURAS DOS TERMOPARES, MEDIAS, INFERENCIAS, ETC
call selectcanal(128+2)
call sleepqq(250)
do j=1,30000
leitura(1)=leitura(1)+read_anl('0',offset)
enddo
leitura(1)=leitura(1)/30000.0
T(1)=150.0*(leitura(1)-1050.0)/(4220.0-1050.0)
Te(1)=T(1)
!
call selectcanal(128+1)
call sleepqq(250)
do j=1,30000
leitura(2)=leitura(2)+read_anl('0',offset)
enddo
leitura(2)=leitura(2)/30000.0
T(2)=150.0*(leitura(2)-1050.0)/(4220.0-1050.0)
Te(2)=T(2)


endif

if (filtro.eq.2.0) then
  call tread(2,2,Te,'false')
endif

y=-1.10142+(0.0748458*T(1))-(0.000645643*T(1)*T(1))
X= 5.22268-(0.0817958*T(2))+(0.000292439*T(2)*T(2))
!
y(1)=y
```

```
py(2)=x

end subroutine

/*************************************************************************\




!   CONTROLADOR FUZZY - MAMDANI
!   PROGRAMA DO CONTROLADOR FUZZY EM FORTRAN
!   RENATO DUTRA PEREIRA FILHO - 19/11/1998
!   INFERENCIA FUZZY USANDO FPS GAUSSIANAS

subroutine infefuzzygauss(funcoes,tipo,pe,pve,p2ve,br,acao,u,pa,lia,lsa)

implicit none
real (8) pe(15,3),pve(15,3),p2ve(15,3),acao,u(20),pa(15,3)
integer funcoes,tipo,l,c,p,br(15,15,15),i
real (8) area,atot,watot,soma,grau,va,base,peso,menor,lia,lsa,parou

area=0.0
atot=0.0
watot=0.0
soma=0.0


! inferencia fuzzy para o caso do controlador proporcional fuzzy
if (tipo .eq.1) then
  do l=1, funcoes
   grau=pe(l,3)
        va=br(l,1,1)
        base=pa(l,1)
        peso=pa(va,2)
        call areagauss(tipo,area,grau,peso,va,funcoes,lia,lsa)
        atot=atot+area
        watot=watot+area*peso
  end do
  acao=watot/atot
end if
! inferencia fuzzy para o caso dos controladores PI e PD fuzzy
if ((tipo .eq.2) .or. (tipo .eq. 3)) then
  do l=1, funcoes
  do c=1, funcoes
        if (pe(l,3).le.pve(c,3)) grau=pe(l,3)
        if (pe(l,3).gt.pve(c,3)) grau=pve(c,3)
        va=br(l,c,1)
        base=pa(l,1)
        peso=pa(va,2)
        call areagauss(tipo,area,grau,peso,va,funcoes,lia,lsa)
        atot=atot+area
        watot=watot+area*peso
  end do
```

```
  end do
  acao=watot/atot
  !write (*,*) acao

end if
! inferencia fuzzy para o caso do controlador PID
if ((tipo.eq.4)) then
  do l=1, funcoes
  do c=1, funcoes
  do p=1,funcoes
          menor=pe(l,3)
          if (menor.ge.pve(c,3)) menor=pve(c,3)
          if (menor.ge.p2ve(p,3)) menor=p2ve(p,3)
          grau=menor

          va=br(c,l,p)
          peso=pa(va,2)
          !write (*,*) br(c,l,p),peso
          !write(*,*) '=========================='

          call areagauss(tipo,area,grau,peso,va,funcoes,lia,lsa)
          atot=atot+area
          watot=watot+area*peso

          !if (pe(3,3).gt.0.95) then
          !  if (peso.gt.0.0) then
          !        write(*,*) area,peso
          !                                read(*,*) parou
          !  endif

          !endif


  end do
  end do
   end do
  acao=watot/atot
 end if

u(1)=acao

!write(*,*) acao,watot,atot
!read(*,*) parou

!write(*,*) watot,atot
!do i=1,funcoes
!write(*,*) pe(i,3),pve(i,3),p2ve(i,3)

!enddo
!read(*,*) parou

end subroutine
/*****************************************************************************\
```

```
!   CONTROLADOR FUZZY - MAMDANI
!   PROGRAMA DO CONTROLADOR FUZZY EM FORTRAN
!   RENATO DUTRA PEREIRA FILHO - 19/11/1998
!   INFERENCIA FUZZY

subroutine infefuzzy(funcoes,tipo,pe,pve,p2ve,br,acao,u,pa)

implicit none
real (8) pe(15,3),pve(15,3),p2ve(15,3),acao,u(20),pa(15,3)
integer funcoes,tipo,l,c,p,br(15,15,15),i
real (8) area,atot,watot,soma,grau,va,base,peso,menor,parou



area=0.0
atot=0.0
watot=0.0
soma=0.0

!do i=1,funcoes
!  write(*,*) pe(i,3),pve(i,3),p2ve(i,3)
!end do
!read(*,*) parou

! inferencia fuzzy para o caso do controlador proporcional fuzzy
if (tipo .eq.1) then
  do l=1, funcoes
   grau=pe(l,3)
          va=br(l,1,1)
          base=pa(l,1)
          peso=pa(va,2)
          area=((base*(1-grau))+base)*(grau/2)
          atot=atot+area
          watot=watot+area*peso
  end do
  acao=watot/atot
end if

! inferencia fuzzy para o caso dos controladores PI e PD fuzzy
if ((tipo .eq.2) .or. (tipo .eq. 3)) then
  do l=1, funcoes
  do c=1, funcoes
          if (pe(l,3).le.pve(c,3)) grau=pe(l,3)
          if (pe(l,3).gt.pve(c,3)) grau=pve(c,3)
          va=br(l,c,1)
          base=pa(l,1)
          peso=pa(va,2)
          area=((base*(1-grau))+base)*(grau/2)
          atot=atot+area
          watot=watot+area*peso
  end do
  end do
  acao=watot/atot
end if
! inferencia fuzzy para o caso do controlador PID
if (tipo.eq.4)  then
```

```
 do l=1, funcoes
 do c=1, funcoes
 do p=1,funcoes
        menor=pe(l,3)
        if (menor.ge.pve(c,3)) menor=pve(c,3)
        if (menor.ge.p2ve(p,3)) menor=p2ve(p,3)
        grau=menor
        va=br(c,l,p)

        base=pa(l,1)
        peso=pa(va,2)
        area=((base*(1-grau))+base)*(grau/2)
        atot=atot+area
        watot=watot+area*peso

 end do
 end do
 end do
 acao=watot/atot
 !write(*,*) acao
 !read(*,*) parou

end if

u(1)=acao


end subroutine


/*****************************************************************************\


!
subroutine initgraph(tipo)
!
 use msflib
 implicit none
!
 integer         tipo
 integer(2)      i,ix,iy,stt
 character(5)    cy
 real(8)         ry
 type(xycoord)   xy
 type(fontinfo)  fi
!
 stt=focusqq(12)
 stt=setcolor(15)
 stt=initializefonts()
 if(stt.le.0) write(*,*)'INITIALIZEFONTS ERROR.'
 if(grstatus().ne.$GROK) write(*,*)'INITIALIZEFONTS GRSTATUS ERROR.'
 stt=setfont('t"Arial"h12w8i')
 stt=getfontinfo(fi)
 stt=rectangle($GBORDER,75,30,485,430)
 call setgtextrotation(0)
 select case(tipo)
```

```
  case(1)
          call moveto(120,10,xy)
          call outgtext('TEMPERATURAS DE TOPO E FUNDO')
         case(2)
          call moveto(125,10,xy)
          call outgtext('COMPOSIÇÕES DE TOPO E FUNDO')
 end select
!
! Caracterização do eixo Y.
!
 ry=0.d0
 stt=setfont('t"Arial"h8w6')
 do i=0,400,40
  iy=430-i
          call moveto(70,iy,xy)
          stt=lineto(75,iy)
          call moveto(485,iy,xy)
          stt=lineto(491,iy)
          call moveto(41,iy-5,xy)
          write(cy,'(F5.1)') ry
          call outgtext(cy)
          call moveto(488,iy-5,xy)
          write(cy,'(F5.1)') ry
          call outgtext(cy)
          if(tipo.eq.1) then
           ry=ry+10.d0
          else
           ry=ry+1.d-1
          endif
 end do
!
! Caracterização do eixo X.
!
 do i=0,416,10
  ix=i+75
          call moveto(ix,430,xy)
          stt=lineto(ix,435)
 end do
!
! Título do eixo X.
!
 stt=setfont('t"Arial"h12w8i')
 call setgtextrotation(0)
 call moveto(250,456,xy)
 call outgtext('Tempo(s)')
!
! Título do eixo Y.
!
 call setgtextrotation(900)
 if(tipo.eq.1) then
  call moveto(15,260,xy)
  call outgtext('Temperatura(°C)')
 else
  call moveto(15,260,xy)
  call outgtext('Composição(%)')
 endif
```

```
! Construção do gráfico de R.
 if(tipo.eq.2) then
   stt=setcolor(15)
          stt=rectangle($GBORDER,565,30,600,430)
          call moveto(582,30,xy)
          stt=lineto(582,430)
   call setgtextrotation(0)
          stt=setfont('t"Arial"h8w6')
          ry=0.d0
          do i=0,400,40
            iy=430-i
            call moveto(560,iy,xy)
            stt=lineto(565,iy)
            call moveto(600,iy,xy)
     stt=lineto(605,iy)
            call moveto(535,iy-5,xy)
            write(cy,'(F4.1)') ry
            call outgtext(cy)
            call moveto(605,iy-5,xy)
            write(cy,'(F5.1)') 10.d0*ry
            call outgtext(cy)
            ry=ry+1.d0
          end do
          stt=setfont('t"Arial"h12w8i')
          call setgtextrotation(900)
          call moveto(525,260,xy)
          call outgtext('Razão de refluxo')
 endif
 stt=setfont('t"Courier"h10w6p')
 return
end subroutine

/*************************************************************************\


!    RENATO DUTRA PEREIRA FILHO - 19/11/1998
!    DEFINICAO DAS LARGURAS DAS FPS TRIANGULARES

!    IGUALMENTE ESPACADAS

subroutine largura(funcoes,pe,re,tipo,rve,pve,p2ve,r2ve,pa,ra)
implicit none
real(8) pe(15,3),re,rve,pve(15,3),p2ve(15,3),r2ve,pa(15,3),ra
integer funcoes,tipo, i
real(8) parou


do i=1, funcoes
  pe(i,1)=re/((funcoes-1)/2)
  if ((tipo .eq.2) .or. (tipo.eq.3) .or. (tipo.eq. 4)) pve(i,1)=rve/((funcoes-1)/2)
  if (tipo .eq.4) p2ve(i,1)=r2ve/((funcoes-1)/2)
  pa(i,1)=ra/((funcoes-1)/2)

end do

end subroutine
```

```
/*************************************************************************\
!   RENATO DUTRA PEREIRA FILHO - 19/11/1998
!   DEFINICAO DAS LARGURAS DAS FPS TRIANGULARES
!   IGUALMENTE ESPACADAS

subroutine largura(funcoes,pe,re,tipo,rve,pve,p2ve,r2ve,pa,ra)
implicit none
real(8) pe(15,3),re,rve,pve(15,3),p2ve(15,3),r2ve,pa(15,3),ra
integer funcoes,tipo, i
real(8) parou


do i=1, funcoes
 pe(i,1)=re/((funcoes-1)/2)
 if ((tipo .eq.2) .or. (tipo.eq.3) .or. (tipo.eq. 4)) pve(i,1)=rve/((funcoes-1)/2)
 if (tipo .eq.4) p2ve(i,1)=r2ve/((funcoes-1)/2)
 pa(i,1)=ra/((funcoes-1)/2)

end do

end subroutine

/*************************************************************************\


! RENATO DUTRA PEREIRA FILHO - 19/11/1998
! Calcula limites das funcoes de pertinencia
!
subroutine limites(tipo,fps,ysp,py,erro,le,lve,l2ve,la,lie,lse,live,lsve,li2ve,ls2ve,lia,lsa,re,rve,r2ve,ra)
implicit none
real (8) ysp,le,lve,l2ve,la,live,lsve,li2ve,ls2ve,lia,lsa,re,rve,r2ve,ra,lie,lse
real (8) erro(20),py(20),parou,parou2,temp
integer fps,tipo

if (fps.eq.2) then
!Valores abaixo nao estao normalizados

 erro(9)=erro(8)                ! erro anterior nao normalizado
 erro(8)=ysp-py(1)              ! erro atual nao normalizado                    *
 erro(11)=erro(10)                          ! variacao do erro anterior nao normalizada
 erro(10)=erro(9)-erro(8)    ! variaco do erro atual nao normalizada          *
 erro(12)=erro(11)-erro(10)  ! 2a. variacao do erro nao normalizada     *

 ! Normalizacao do Erro Atual
 temp=erro(8)/le
 if (temp.lt.-1.0) temp=-1.0
 if (temp.gt.+1.0) temp=+1.0
 erro(1)=temp
 if (tipo.ne.1) then
 ! Normalizacao da Variacao do Erro
 temp=erro(10)/lve
 if (temp.lt.-1.0) temp=-1.0
 if (temp.gt.+1.0) temp=+1.0
 erro(3)=temp
 end if
```

```
! Normalizacao da 2a. Variacao do Erro
if (tipo.eq.4) then
temp=erro(12)/l2ve
if (temp.lt.-1.0) temp=-1.0
if (temp.gt.+1.0) temp=+1.0
erro(5)=temp
endif

 lie=-1                ! limite inferior do erro
 lse=+1                ! limite superior do erro
 lsve=+1               ! limite superior da variacao do erro
 live=-1                                    ! limite inferior da variacao do erro
 li2ve=-1              ! limite inferior da 2a. variacao do erro
 ls2ve=+1                                   ! limite superior da 2a. variacao do erro
 if (tipo.eq.1) lia=0.0
 if (tipo.eq.2) lia=0.0
 if ((tipo.eq.3) .or. (tipo.eq.4)) lia=-1          ! limite inferior da acao
 lsa=+1                ! limite superior da acao
 re=lse-lie            ! range do erro
 rve=lsve-live           ! range da variacao do erro
 r2ve=ls2ve-li2ve        ! range da 2a. variacao do erro
 ra=lsa-lia            ! range da acao
end if


if (fps.eq.1) then
erro(2)=erro(1)                    ! erro penultimo normalizado
erro(1)=(ysp-py(1))                ! calcula erro normalizado
erro(4)=erro(3)                    ! erro antepenultimo normalizado
erro(3)=(erro(2)-erro(1))  ! calcula variacao do erro normalizda
erro(5)=(erro(4)-erro(3))  ! 2a. variacao do erro normalizada
lie=-le               ! limite inferior do erro
lse=+le               ! limite superior do erro
lsve=+lve             ! limite superior da variacao do erro
live=-lve                          ! limite inferior da variacao do erro
li2ve=-l2ve           ! limite inferior da 2a. variacao do erro
ls2ve=+l2ve                        ! limite superior da 2a. variacao do erro
if (tipo.eq.1) lia=0.0
if (tipo.eq.2) lia=0.0
if ((tipo.eq.3) .or. (tipo.eq.4)) lia=-la   ! limite inferior da acao
lsa=+la               ! limite superior da acao
re=lse-lie            ! range do erro
rve=lsve-live           ! range da variacao do erro
r2ve=ls2ve-li2ve        ! range da 2a. variacao do erro
ra=lsa-lia            ! range da acao
end if
end subroutine
/*************************************************************************\

!
subroutine mkwindows()

!
  use msflib
!
  implicit none
```

```
!
 integer(4)       result
 logical          mode
 type(qwinfo)     winfo
 type(windowconfig) screen
!
! Set highest resolution graphics mode.
!
 screen.numxpixels=-1
 screen.numypixels=-1
 screen.numtextcols=-1
 screen.numtextrows=-1
 screen.numcolors=-1
 screen.fontsize=-1
 screen.title= "MENSAGENS DE ERRO"C !BLANK
 mode=setwindowconfig(screen)
 if(.not.mode) stop 'ERROR: CANNOT SET GRAPHICS MODE'
 open(UNIT=12,FILE='USER',TITLE='TELA DE ACOMPANHAMENTO')
 open(UNIT=13,FILE='USER',TITLE='DADOS DE ENTRADA')
!
! Determine the minimum and maximum dimensions.
!
 mode=getwindowconfig(screen)
 winfo.TYPE = QWIN$MAX
 result=SETWSIZEQQ(QWIN$FRAMEWINDOW,winfo)
 return
end subroutine
/*****************************************************************************\
subroutine selectcanal(p)
!
 implicit none
!
 include 'cfiles.fi'
!
 integer(1)    p
!
 call write_dig(p)
 return
end subroutine
/*****************************************************************************\

!
subroutine tread(na,nt,cnt,Te,test)
!
 implicit none
!
! ms$attributes C,EXTERN :: offset
 character(1)    offset
!
 include 'cfiles.fi'
!
 integer         na,nt,cnt

 real(8)         Te(nt)
 logical(4)      test
!
```

```
 integer          p,i,j,nl1,nl2
 real(8)          a,leit,soma,leitdig, &
                & vfa1(2),vfa2(2)
 parameter          (a=0.33d0,nl1=2.d+4,nl2=1.d+4)
 save             vfa1,vfa2
!
 if(test) then
  read(15,*) Te(1),Te(2)
 else
  do i=1,cnt
   if(i.eq.1) then
             p=128+2
            else
             p=128+1
            end if
   call selectcanal(p)
            if(.not.test) then
     call sleepqq(250)
            end if
    p=p-128
    select case(na)
     case(0:1)
                      soma=0.d0
       do j=1,nl1
                       soma=soma+read_anl(char(0),offset)
                      enddo
                      leitdig=soma/nl1
                      vfa2(p)=vfa1(p)
                      vfa1(p)=leitdig
              case default
                      soma=0.d0
                      do j=1,nl2
                       soma=soma+read_anl(char(0),offset)
                      enddo
                      leit=soma/nl2
                      leitdig=leit*(a**2.d0)+(1.0d0-a)*2.d0* &
                          & vfa1(p)-((1.d0-a)**2.d0)*vfa2(p)
                      vfa2(p)=vfa1(p)
                      vfa1(p)=leitdig
            end select
            Te(i)=1.47d2*(leitdig-1050.)/(4220.-1050.)
   enddo
  end if
 return

 end subroutine


 /*************************************************************************\

 !
 subroutine valve(t,Ta,tD,tL,dt,R,test)
 !
  use portlib
 !
  implicit none
```

```
!
 real(8)         R,t,Ta,tD,tL,dt
 logical(4)      test
!
 dt=timef()-t
 tL=Ta*R/(1.+R)-dt
 if(tL.gt.0.) then
  if(.not.test) then
   call sleepqq(int(tL*1000.))
        end if
  tL=tL+dt
 else
  tL=dt
 endif
 tD=Ta-tL
 if(tD.gt.1.) then
  call selectcanal(0)
  if(.not.test) then
        call sleepqq(int(tD*1000.))
       end if
 endif
 return
end subroutine

/*************************************************************************\

/* PROGRAMA PARA AQUISICAO DE DADOS - VERSAO 1.0 */

#include "conio.h"

#define ADLSB  0      /* Porta de leitura do lsb do conv ad */
#define ADMSB  1      /* Porta de leitura do msb do conv ad */
#define ADOFF  2      /* Porta para ajuste de offset */
#define ADSTS  4      /* Porta de controle do modo de operacao */
#define DAMSB  6      /* Porta de escrita do msb do conversor da */
#define DALSB  7      /* Porta de escrita do lsb do conversor da */
#define CTL    8      /* Porta de controle do mux de ent/saida e dos sh */
#define base   0x220
#define IODIG  10     /* Entrada e saida digital */
#define TIMER0 12     /* Timer 0 do 8253 */
#define TIMER1 13     /* Timer 1 do 8253 */
#define TIMER2 14     /* Timer 2 do 8253 */
#define TIMCTL 15     /* Porta de controle do 8253 */

/* Definicao dos bits de importancia */

#define BSHEAN 0x10  /* Bit de controle do sample-hold entanl (1=sample) */
#define BSHSA  0x08  /* Bit de controle do sample-holds das saidas(1=sample)*/
#define MASC0_5 0x00  /* Mascara no modo de operacao 0-5V */
#define MASC1_5 0x29  /* Mascara no modo de operacao 1-5V */
#define MASC0_4 0x08  /* Mascara no modo de operacao 0-4V */
#define MASCBIP 0x23  /* Mascara no modo de operacao BIPOLAR */
#define TIME_OUT 25   /* Tempo de espera do fim da conversao A/D */

extern char erro;     /* Codigo de erro */
unsigned char modoper; /* Byte que contem o modo de operacao da placa */
```

```
unsigned int read_anl(),adj_offset();
unsigned char offset;
void selchda();
void write_dig();
void write_anl();
void wait_eoc();          */

/*-------------------------------------------------------------------*/

     /*  INICIO DAS ROTINAS DE TRATAMENTO DA AD/DA  */

/*-------------------------------------------------------------------*/

   /* SELECAO DO CANAL DO MUX DE ENTRADA E DA SAMPLE NA ENTRADA */

/*-------------------------------------------------------------------*/

selchad(canal)
unsigned char canal;
{
 unsigned char chad;
 chad = canal << 5;              /* Posiciona o end do mux (badchan ) */
 chad &= 0xe0;                   /* Isola somente badchan0-2 */
 _outp(base + CTL,chad);         /* Seleciona o canal */
 modoper |= BSHEAN;              /* Introduz bit de sample */
 _outp(base + ADSTS,modoper);    /* Sample-hold da inanl em sample */
 modoper &= ~BSHEAN;             /* Retira bit de sample */
 _outp(base + ADSTS,modoper);    /* Sample-hold da inanl em hold */
 return(canal);
}

/*-------------------------------------------------------------------*/

   /* SELECIONA O CANAL DO MUX DE SAIDA E DA SAMPLE NA SAIDA */

/*-------------------------------------------------------------------*/

void selchda(canal)
unsigned char canal;
{
 unsigned char chda;
 canal &= 0x07;          /* Isola */
 chda = (canal << 5);    /* Posiciona o end do mux (badchan0-2 ) */
 chda |= canal;          /* Soma c/ o mux do sample-hold (bdachan0-2) */
                         /* Seleciona o canal+sample-hold corresp */
 chda |= BSHSA;          /* Introduz o bit de sample */
 _outp(base + CTL,chda); /* Coloca o sample-hold do canal em sample */
}

/*-------------------------------------------------------------------*/

       /* ROTINA DE ESPERA DO FIM DA CONVERSAO */

/*-------------------------------------------------------------------*/

wait_eoc()
```

```
{
 unsigned register int ciclos,status;
 for (ciclos = TIME_OUT;ciclos;ciclos --)
 {
 }
 return 0;
}

/*-------------------------------------------------------------------*/

          /* ROTINA DE LEITURA DO CONVERSOR */

      /* Parametro de entrada: numero do canal
        Parametro de saida:  retorna o valor da conversao
                                    se ocorrer erro retorna -1 */

/*-------------------------------------------------------------------*/
wait_eoc(void);
selchad(unsigned char canal);
unsigned int read_anl(canal,offset)
unsigned char canal,offset;
{
 unsigned int dado;
 unsigned int dadols,dadoms;
 selchad(canal);
 _outp(base+ADOFF,offset);      /* Normaliza o valor do offset */
 dadols = _inp(base+ADLSB);     /* Envia o start ao ad */
 wait_eoc();                    /* Delay para conversao */
 dadoms = _inp(base+ADMSB);     /* le os 4 bit`s mais signific. */
 dadols = _inp(base+ADLSB);     /* le os 8 bit's menos signific. */
 dado = (dadoms << 8) + dadols;
 return (dado);
}

/*-------------------------------------------------------------------*/

    /* ESCREVE O VALOR DE 'dado' NA SAIDA ANALOGICA DE 'canal' */

/*-------------------------------------------------------------------*/
void selchda(unsigned char canal);
void write_anl(dado,canal)
unsigned int dado;
unsigned char canal;
{
 unsigned char dadols,dadoms;
 dadols = dado;                 /* Inicia o deslocamento do dado */
 dadoms = dado >> 8;            /* Desloca os dois bits mais signific. */
 _outp(base + DALSB,dadols);  /* Escreve byte menos significativo */
 _outp(base + DAMSB,dadoms);  /* Escreve byte mais significativo */
 selchda(canal);                /* Transf para o canal de saida desejado */
 return;
}
/*-------------------------------------------------------------------*/

                  /* ESCREVE NAS ENTRADAS DIGITAIS */
```

```
/*-------------------------------------------------------------------*/

void write_dig(dado)
char dado;
{
  _outp(base + IODIG,dado);
}

/*-------------------------------------------------------------------*/

          /* ALGORITMO PARA AJUSTE AUTOMATICO DE OFFSET */

          /* Retorna os seguintes valores:

         100h -> Se nao existe o sinal de referencia de 3.500V na
                 entrada analogica 7
         200h -> Se em  4 tentativas de ajustar o offset isto nao
                 for conseguido
            Num -> De 0h a 0FFh que  e' o  valor para zerar o offset */

/*-------------------------------------------------------------------*/
unsigned int read_anl(unsigned char canal,unsigned char offset);
unsigned int adj_offset()
{
  unsigned int in1,in2,i,inatual,trigger = 0x800;
  unsigned char flag = 0,delay;
  if(modoper & 0x01)       /* modo 1 a 5 V ? */
  trigger = 0xa00;         /* sim, armazene a00h como valor de comparacao */
  else
  trigger = 0xb33;
  inatual = read_anl(7,140);      /* le a ent de refer. centrando o offset */
  if(inatual < (trigger - 0x100)) /* existe a referencia de (3500mV)?  */
  return(0x100);                  /* nao,retorne uma condicao de erro */
  for(i=0;flag ==0;++i)           /* inicio do integ. p/ busca do pto otimo */
  {
   inatual = read_anl(7,(i&0xff)); /* tente com i valor deoffset */
   for(delay=0;delay<50;++delay)   /* rotina para atraso */
   if(inatual == trigger)          /* erro de leitura = zero? */
   flag = 1;                     /* termine a execucao, ponto encontrado */
   if(i == 0x400)                  /* feita o scan 4 vezes sem sucesso? */
   return(0x200);                  /* termine e retorne condicao de erro */
  }
  return (i-1);                    /* termine e retorne o valor ajustado */
 }

 /*-------------------------------------------------------------------*/

          /* ROTINA PARA SELECAO AUTOMATICA DO MODO DE OPERACAO
            ACEITA OS SEGUINTES PARAMETROS: */

          /* n = 0 ->  0-5V  (modo default)
            n = 1 ->  1-5V
            n = 2 ->  0-4V
            n = 3 ->  bipolar

            OBS : modo de leitura do conversor : pooling       */
```

```
/*-------------------------------------------------------------------*/

modo_oper(n)
char n;
{
 unsigned char masc;
 masc=MASC0_5;      /* default operacao de 0-5V */
 switch(n)          /* scan do modo de operacao alternativo */
 {
  case 1:
   masc=MASC1_5;
   break;
  case 2:
   masc=MASC0_4;
   break;
  case 3:
   masc=MASCBIP;
   break;
 }
 modoper = masc;            /* armazena o modo de operacao setado */
 _outp(base+ADSTS,masc);  /* envia a placa */
 return 0;
}

/*-------------------------------------------------------------------*/
```

```
!
interface
subroutine baseregras(tipo,br,funcoes)
integer tipo, funcoes,br(15,15,15)
end subroutine
end interface
!
interface
subroutine areagauss(tipo,area,grau,peso,va,funcoes,lia,lsa)
integer funcoes,tipo
real (8) area,grau,peso,va,lia,lsa
end subroutine
end interface
!
interface
subroutine
centros(funcoes,pe,re,tipo,rve,pve,p2ve,r2ve,pa,lie,live,li2ve,lia,lse,ls2ve,l
sa)
real(8)
pe(15,3),re,rve,pve(15,3),p2ve(15,3),r2ve,pa(15,3),lie,live,li2ve,lia,lsa,lse,
ls2ve
integer funcoes,tipo
end subroutine
end interface
!
interface
```

```
subroutine
confuzzy1(fps,binario,tipo,funcoes,la,le,lve,l2ve,Te,py,erro,u,ysp,nt,iparam,t
est,ced,nc,cp,R,louco,filtro)
logical (4) test
real(8) la,le,lve,l2ve,te(8),py(20),erro(20),u(20),ysp,louco,filtro
integer tipo, funcoes,binario,nt,iparam(9),ced,nc,fps
real(8) cp(nc,nt),R,texp(2)
end subroutine
end interface
!
interface
subroutine
confuzzy2(fps,binario,tipo,funcoes,la,le,lve,l2ve,Te,py,erro,u,ysp,nt,iparam,t
est,ced,nc,cp,R,louco,filtro)
logical (4) test
real(8) la,le,lve,l2ve,te(8),py(20),erro(20),u(20),ysp,louco,texp(2),filtro
integer tipo, funcoes,binario,nt,iparam(9),ced,nc,fps
real(8) cp(nc,nt),R
end subroutine
end interface
!
interface
subroutine fgauss(x,u,teta,yy)
real(8) x,u,teta,yy
end subroutine
end interface
!
interface
subroutine dfgauss(x,u,teta,yy)
real(8) x,u,teta,yy
end subroutine
end interface
!
interface
subroutine fuzzifica(tipo,erro,pe,funcoes,pve,p2ve)
real(8) erro(20), pe(15,3), pve(15,3), p2ve(15,3)
integer tipo, funcoes
end subroutine
end interface
!
interface
subroutine fuzzificagauss(tipo,erro,pe,funcoes,pve,p2ve)
integer tipo,funcoes
real(8) erro(20),pe(15,3),pve(15,3),p2ve(15,3)
end subroutine
end interface
!
interface
subroutine infecompo(Te,py,filtro)
real(8) Te(8), py(20),filtro
end subroutine
end interface
!
interface
subroutine infefuzzy(funcoes,tipo,pe,pve,p2ve,br,acao,u,pa)
real (8) pe(15,3), pve(15,3),p2ve(15,3),br(15,15,15), acao,u(20),pa(15,3)
integer funcoes, tipo
```

```
end subroutine
end interface
!
interface
subroutine largura(funcoes,pe,re,tipo,rve,pve,p2ve,r2ve,pa,ra)
real (8) pe(15,3), re,rve,pve(15,3),p2ve(15,3),r2ve,pa(15,3),ra
integer funcoes, tipo
end subroutine
end interface
!
interface
subroutine
limites(tipo,fps,ysp,py,erro,le,lve,l2ve,la,lie,lse,live,lsve,li2ve,ls2ve,lia,
lsa,re,rve,r2ve,ra)
real (8)
ysp,py(20),erro(20),le,lve,l2ve,la,lie,lse,live,lsve,li2ve,ls2ve,lia,lsa,re,rv
e,r2ve,ra
integer fps,tipo
end subroutine
end interface
!
interface
subroutine graphic(tipo,na,ced,nc,nt,nv,np,iparam, &
& par,cp,sp,Te,R)
integer tipo,na,ced,nc,nt,nv,np,iparam(9)
real(8) par(nc,nv),cp(nc,nt),sp(nc),Te(nt),R
end subroutine
end interface
!
interface
subroutine initgraph(tipo)
integer tipo
end subroutine
end interface
!
interface
subroutine inputdata(fuzzyparam,iparam,rparam,fparam,nc,nv, &
& par,sp,test)
logical(4) test
character(50) fparam(5)
integer nc,nv,iparam(9)
real(8) par(nc,nv),sp(nc),rparam(9),fuzzyparam(10)
end subroutine
end interface
!
interface
subroutine mkwindows()
end subroutine
end interface
!
interface
subroutine selectcanal(p)
integer p
end subroutine
end interface
!
interface
```

```
subroutine valve(t,Ta,tD,tL,dt,R,test)
real(8) t,Ta,tD,tL,dt,R
logical(4) test
end subroutine
end interface
!
interface
subroutine tread(na,nt,cnt,Te,test)
integer na,nt,cnt
real(8) Te(nt)
logical(4) test
end subroutine
end interface
!
interface
subroutine yinfer(ced,nc,nt,Te,cp,R)
integer ced,nc,nt
real(8) Te(nt),cp(nc,nt),R
end subroutine
end interface
!
!
interface
subroutine write_dig(dado)
!ms$attributes C,ALIAS:'_write_dig' :: write_dig
integer(1) dado
end subroutine
end interface
!
interface
integer(4) function read_anl(canal,offset)
!ms$attributes C,ALIAS:'_read_anl' :: read_anl
character(1) canal,offset
end function
end interface
```

UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA QUÍMICA

ÁREA DE CONCENTRAÇÃO SISTEMAS DE PROCESSOS QUÍMICOS E INFORMÁTICA

DESENVOLVIMENTO E IMPLEMENTAÇÃO DE CONTROLADORES NEBULOSOS EM UMA COLUNA PILOTO DE DESTILAÇÃO EM BATELADA

Autor: Renato Dutra Pereira Filho.
Orientadora: Profª. Drª. Ana Maria Frattini Fileti.
Co-Orientador: Prof. Dr. João Alexandre F. R. Pereira.

Dissertação de Mestrado

Campinas – SP Brasil

Março - 1999