Universidade Federal de Santa Catarina
Centro Tecnológico
Curso de Engenharia de Controle e
Automação Industrial

Implementação de um Controlador Nebuloso em Transputer

Monografia submetida à Universidade Federal de Santa Catarina
como requisito para a aprovação da disciplina:
EEL 5901: Projeto de Fim de Curso

Pedro Paulo da Silva

Florianópolis, Dezembro de 1994

Implementação de um Controlador Nebuloso em Transputer

Pedro Paulo da Silva

Esta monografia foi julgada no contexto da disciplina
EEL 5901: Projeto de Fim de Curso
e aprovada na sua forma final pelo
Curso de Engenharia de Controle e Automação Industrial

Banca Examinadora:

Eng. Josué Júnior Guimarães Ramos
Orientador, Centro Tecnológico para Informática
pesquisador visitante na Gyron Tecnologia

Prof. Guilherme Bittencourt
Orientador, ECAI-UFSC

Prof. Augusto Humberto Bruciapaglia
Responsável pela Disciplina e Coordenador do Curso

Prof. Daniel Juan Pagano, Avaliador

Frederico Scott Brusa Mesquita, Debatedor

José Paulo de Lucca Ramos, Debatedor

AGRADECIMENTOS

Não teria sentido passar por estes cinco anos de curso sem poder olhar para trás e ver quanta gente ajudou a me construir como eu sou e a chegar onde estou hoje.

Há muito a ser dito mas, de fato, não somente em palavras. Há muito que eu devo dizer ao longo da minha vida.

Agradeço a meus pais, pelo incentivo de ter voltado a estudar após sete anos de abstinência escolar. A meu pai, Inácio, porque eu sei que ele rezou por mim para que eu conseguisse entrar e sair da universidade e à minha mãe, Oscarina, pelo apoio dado a cada dia.

Agradeço aos meus irmãos e irmã, às minhas cunhadas, aos meus sobrinhos e aos meus amigos de sempre por estarem presentes e por serem um abrigo seguro de conforto.

Agradeço aos professores, em especial, aqueles loucos apaixonados que, perdidos no meio de números e sonhos abstratos, demonstraram e transmitiram muita força e determinação na sua função de ensinar.

Agradeço aos colegas e amigos do EDUGRAF, do LCMI e do curso de Automação, pelo apoio e pela amizade que, eu espero, não se limite somente ao convívio que, a partir de agora, deixará de ser tão freqüente quanto foi durante este período de cinco anos.

Agradeço ao pessoal da coordenadoria, Salete e Prof. Augusto, pelo apoio e também pela amizade.

Agradeço aos meus orientadores que durante a execução deste trabalho me proporcionaram um valioso aprendizado não somente profissional mas também intelectual. Ao Eng. Josué pelo seu esforço, pelo seu apoio e orientação desde o início até a conclusão de todo o trabalho, a quem considero hoje como um amigo, e ao Prof. Guilherme que, por ocasião do afastamento da Profa. Janete, assumiu a orientação do trabalho, dedicando muita atenção, empenho e talento, contribuindo com suas valiosas sugestões. Agradeço também a Carla Cavalcante pelo seu auxílio sempre oportuno e, também, por ter-me fornecido os arquivos de sua tese, poupando-me algumas horas de trabalho na edição de figuras.

Agradeço à Marta porque sei que durante cada instante eu tive e ainda tenho o seu apoio, sua amizade e o seu amor.

Por último, porém não menos importante, agradeço a Deus pela força de vontade, pela determinação e pelo orientação de que pude desfrutar, sustentados pela minha fé.

RESUMO

Este trabalho apresenta a implementação em linguagem Forth de uma máquina de inferência em lógica nebulosa em uma plataforma para transputer.

Esta máquina de inferência será utilizada na implementação de um controlador nebuloso a ser aplicado em um sistema automático de navegação para um helicóptero não tripulado desenvolvido pela Gyron Tecnologia.

Forth é a linguagem de programação utilizada no sistema operacional do processador central embarcado no helicóptero: o transputer da INMOS.

Este trabalho está organizado como segue:

Inicialmente são descritos detalhes relevantes do funcionamento de helicópteros. Em seguida são apresentados aspectos da teoria de lógica nebulosa e de controladores nebulosos.

Finalmente é descrita a estruturação das máquinas de inferência em módulos e é definida uma missão para o ensaio do sistema de navegação, permitindo verificar o desempenho do controlador nebuloso. Este ensaio é feito a nível de simulação sendo que os seus resultados são comparados com os resultados obtidos em [Cav 94], no qual foi projetado o controlador nebuloso.

ABSTRACT

This work presents the implementation of a fuzzy-forth inference machine in transputer.

This inference machine will be used in a fuzzy controller implementation that will be applied in a unmaned helicopter automatic guidance system developed by Gyron Tecnologia.

Forth is the programming language used in the helicopter's main processor operating system, that uses the INMOS transputer.

Tris work is organized as follows:

Initially the relevant details of helicopters functioning are described. Next, the fuzzy logic and fuzzy controller theoretical aspects are presented.

Finally the structuration of the inference machines in modules is described and a mission to test the navigation system and to allow the verification of the fuzzy controller performance is defined. This test is made by simulation and the results are compared with the results obtained in [Cav 94], where the fuzzy controller was designed.

SUMÁRIO

INTRODUÇÃO

Certas tarefas de inspeção, como ocorre em linhas de transmissão de energia elétrica, oferecem risco ao ser humano. Para este tipo de tarefas é preferível a utilização de dispositivos de monitoramento remoto em satélites ou através de imagens aéreas.

A resolução das imagens de satélites, entretanto, não é adequada a esse tipo de inspeção devido à sua baixa resolução. Outra questão a considerar seria que a operação periódica de levantamento de imagens aéreas apresenta um custo elevado.

Para aliar uma boa resolução de imagens com um baixo custo operacional surge a opção por veículos aéreos não tripulados dotados de dispositivos de visão.

Para inspeções com as características exigidas, como no caso de linhas de transmissão de energia elétrica, o veículo deve possuir as características de vôo pairado e seguimento de uma trajetória no espaço. O helicóptero aparece como a opção mais adequada.

Motivada pela inexistência de tal sistema de inspeção no mercado, a Gyron Tecnologia criou, em 1991, o projeto Helix, cujo objetivo é o desenvolvimento de um helicóptero não tripulado com rotor de 2.0 metros, para executar tarefas de observação aérea, remotamente controlado por um piloto.

Porém o treinamento de um piloto é oneroso e, além disso, as tarefas de inspeção consistem em operações repetitivas e conseqüentemente desgastantes. Sendo assim, um dos objetivos do projeto Helix é a realização de um sistema automático de navegação para permitir a substituição do piloto por um operador menos treinado.

A estratégia de controle implementada pelo piloto é baseada em um conjunto de regras que descreve as situações que podem ocorrer no sistema e as ações correspondentes que devem ser tomadas. Portanto, para simular a tomada de decisões que o piloto realiza durante a pilotagem é necessário um algoritmo que utilize um conjunto de regras construído a partir do conhecimento do piloto ou da observação de suas atitudes durante a pilotagem. Este conjunto de regras tem por objetivo permitir a tomada de decisões automática através da análise de quais ações devem ser executadas em face de uma situação qualquer do sistema. Este algoritmo de controle é implementado, neste sistema, através de um Controlador Nebuloso, que compõe o sistema automático de navegação do helicóptero.

Este sistema automático de navegação foi proposto e implementado em uma plataforma IBM-PC em [Cav 94], considerando todas as etapas envolvidas no projeto de um controlador nebuloso.

O objetivo deste trabalho é a re-implementação do sistema de navegação proposto em [Cav 94] em uma plataforma em linguagem forth para transputer. A razão do uso de tal plataforma é que o helicóptero da Gyron Tecnologia possui como processador central um

transputer com um sistema operacional em forth. Desta forma a implementação beneficia a aplicação do controlador nebuloso projetado, diretamente no helicóptero.

O projeto voltado para a implementação da máquina de inferência que irá compor o controlador nebuloso foi desenvolvido em duas etapas: especificação e implementação de um gerador e um tradutor de tabelas de regras nebulosas e especificação e implementação das máquinas de inferência. Em seguida foi estabelecida uma missão que permitiu a simulação de todo o sistema de navegação.

O trabalho foi desenvolvido segundo um cronograma proposto, no qual as atividades foram organizadas como segue:

- estudo de controladores nebulosos;
- desenvolvimento em "C" de um tradutor de tabelas de regras nebulosas para instruções do tipo IF...THEN...;
- desenvolvimento de um gerador de código forth que traduza as tabelas de regras;
- instalação da infraestrutura de desenvolvimento para transputer;
- Especificação de Máquina de Inferência para Transputer (atividade realizada em conjunto com a equipe de Controle e Navegação de Helicópteros);
- desenvolvimento da máquina de inferência para lógica nebulosa.

Este trabalho está organizado como descrito a seguir.

No capítulo 1 encontram-se os fundamentos básicos necessários à compreensão do funcionamento de helicópteros. São descritos os movimentos do helicóptero, os comandos que tornam possível a sua pilotagem e são apresentadas questões de acoplamento.

No capítulo 2 são descritos em linhas gerais os fundamentos da teoria da lógica nebulosa e de controladores nebulosos.

No capítulo 3 é apresentada a estrutura do sistema de navegação com a descrição de cada uma das tarefas que compõem a missão, considerando as variáveis de entrada, variável de saída e acoplamentos.

No capítulo 4 é descrita a estrutura lógica do controlador nebuloso, enquanto que no capítulo 5 são apresentados os detalhes de implementação em forth das máquinas de inferência, do controlador nebuloso e da missão.

No capítulo 6 é descrita a simulação do sistema de navegação. São também apresentados e discutidos os resultados obtidos bem como apresentadas soluções para a melhoria de desempenho do sistema.

Finalmente, são apresentadas as conclusões e comentadas algumas perspectivas futuras para o trabalho.

HELICÓPTERO

Neste capítulo são descritos os princípios de funcionamento de helicópteros de uma maneira geral sem entrar nos detalhes construtivos da aeronave.

São descritos os tipos de movimentos do helicóptero, os comandos que utiliza e são apresentadas questões de acoplamento.

1.1 INTRODUÇÃO

Helicópteros são aeronaves que apresentam grande versatilidade de vôo devido às suas características construtivas, que os tornam veículos adequados às operações de inspeção. Possuem capacidade de pouso e decolagem verticais, vôo pairado, vôo para frente e para trás, vôos laterais e auto-rotação.

Entretanto, a pilotagem de helicópteros exige um conhecimento dos seus princípios de funcionamento e das suas características de vôo, fazendo com que a tarefa do piloto não seja simples.

Os objetivos deste capítulo são, então, apontar as características de funcionamento de helicópteros descrevendo os fatores operacionais que devem ser considerados na sua pilotagem e promover uma familiarização com a terminologia utilizada ao longo deste trabalho.

1.2 COMPONENTES DO HELICÓPTERO

Os principais componentes aerodinâmicos de um helicóptero são (ver figura 1.1):

- *rotor principal*: responsável por fornecer propulsão, suporte e controlabilidade à aeronave [Pallet 83];
- *rotor de cauda*: responsável por contrabalançar os efeitos de reação da fuselagem ao movimento de rotação das pás do rotor principal;
- *estabilizadores vertical e horizontal*: contribuem para manter a aeronave na posição normal de vôo (nariz à frente e nivelado com a direção de vôo).

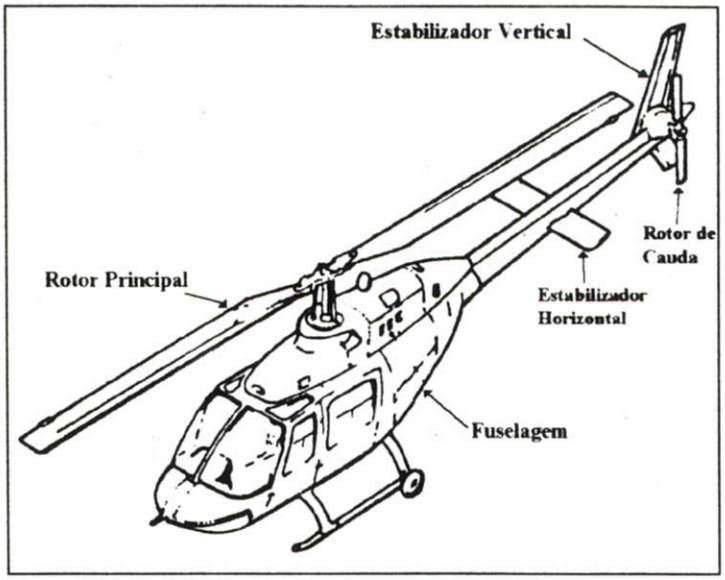


Fig. 1.1 Componentes Aerodinâmicos do Helicóptero

1.3 MOVIMENTOS DO HELICÓPTERO

Considerando um sistema de três eixos com a origem no centro de gravidade do helicóptero, podemos descrever os movimentos básicos (ver figura 1.2):

- movimentos ao longo do eixo X são chamados *longitudinais*;
- movimentos ao longo do eixo Y são chamados *laterais*;
- movimentos ao longo do eixo Z são chamados *verticais*;

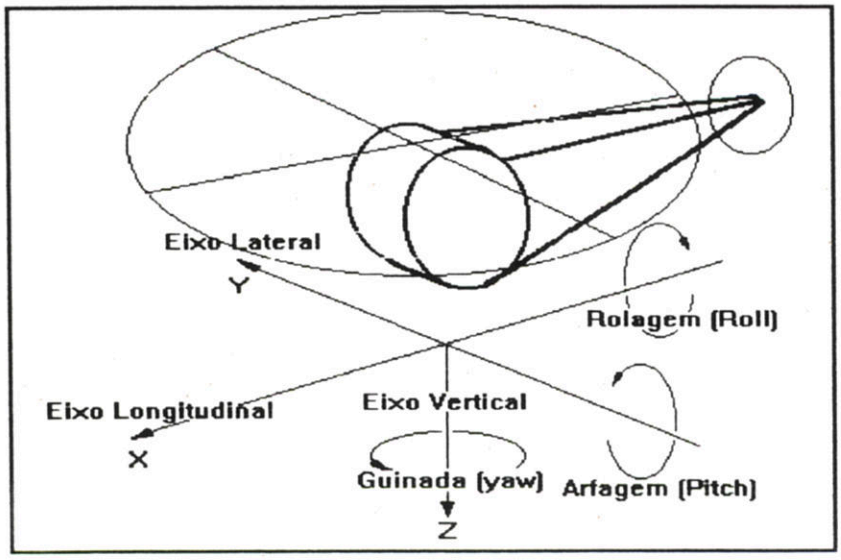


Fig. 1.2 - Os principais eixos e movimentos de rotação do helicóptero

- movimentos em torno do eixo X são chamados *rolagem (roll)*;
- movimentos em torno do eixo Y são chamados *arfagem (pitch)*;
- movimentos em torno do eixo Z são chamados *guinada (yaw)*.

1.4 COMANDOS DO HELICÓPTERO

O helicóptero utiliza quatro comandos básicos em suas manobras de vôo:

- *comando coletivo do rotor principal*: ou, simplesmente, comando coletivo. Este comando atua sobre o ângulo das pás do rotor principal do helicóptero, alterando a sua força de sustentação. Se a força de sustentação for maior que o peso, o helicóptero acelera para cima, se for menor que o peso, acelera para baixo e se for igual ao peso, o helicóptero permanece a uma altura constante (figura 1.3);

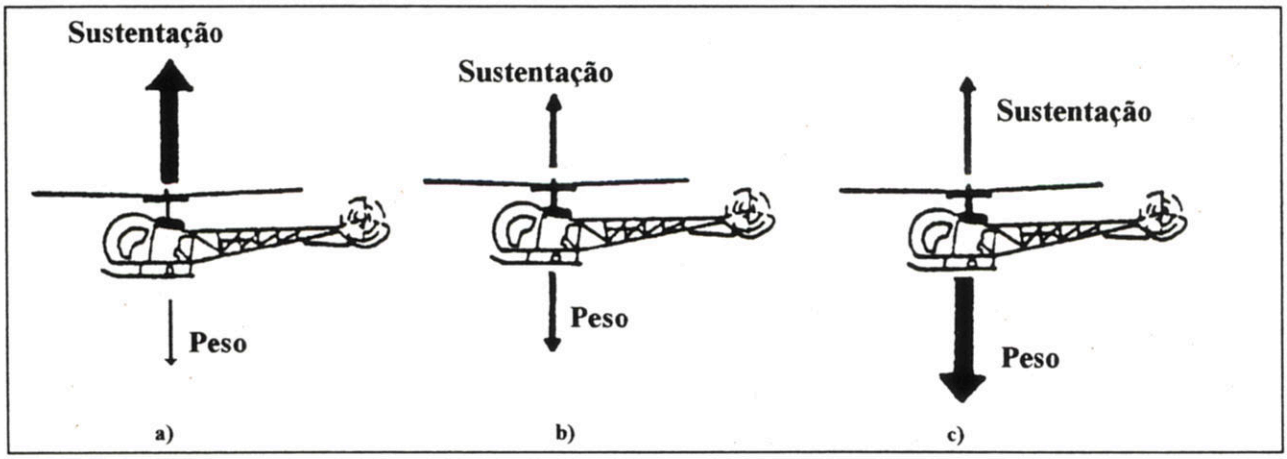


Fig. 1.3 - a) Sustentação maior que o peso, helicóptero sobe; b) sustentação igual ao peso, helicóptero permanece parado; c) sustentação menor que o peso, helicóptero desce.

- *comando coletivo do rotor de cauda*: ou, simplesmente, comando de cauda. Este comando atua sobre o ângulo das pás do rotor de cauda, variando a força que tende a equilibrar a reação da fuselagem ao giro do rotor principal. Com isso consegue-se um giro num plano horizontal em torno do eixo ***Z***, direcionando o nariz do helicóptero (figura 1.4);
- *comando cíclico longitudinal*: este comando atua sobre o ângulo das pás do rotor principal, inclinando o vetor de sustentação para frente ou para trás. O objetivo é conseguir um movimento nestas direções (figura 1.5);

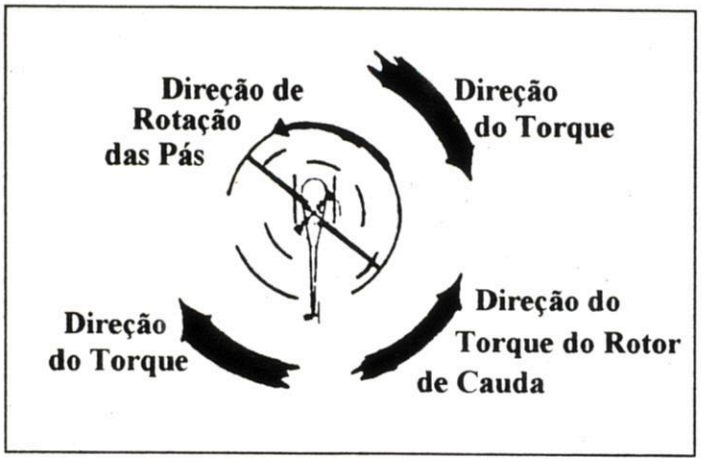


Fig. 1.4 - A atuação sobre o torque do rotor de cauda varia a força que tende a compensar os efeitos do torque do rotor principal. Quando as forças se igualam, o helicóptero permanece com um ângulo de guinada fixo.

- *comando cíclico lateral*: da mesma forma que o comando cíclico longitudinal, o comando cíclico lateral atua sobre o ângulo das pás do rotor principal, inclinando a força de sustentação para os lados, permitindo movimentos laterais (figura 1.5).

1.4 ACOPLAMENTOS

O helicóptero é uma aeronave com características de acoplamento entre os seus movimentos. Tais acoplamentos solicitam a capacidade do piloto quando este pretende efetuar uma manobra de vôo.

A proposta do sistema automático de navegação de helicópteros projetado em [Cav 94] foi a criação de um sistema automático para atenuar os efeitos dos acoplamentos na pilotagem do helicóptero, facilitando a tarefa do piloto e, também, permitindo a substituição do piloto por um operador menos treinado.

Um exemplo de acoplamento entre os movimentos pode ser dado através do seu vôo à frente.

Quando o helicóptero encontra-se em vôo pairado o vetor sustentação encontra-se em equilíbrio com o peso da aeronave (fig. 1.3 b). Ao ser inclinado o rotor do helicóptero para o vôo à frente, o vetor sustentação também é inclinado neste sentido. Como resultado temos uma componente vertical do vetor de sustentação que não mais equilibra a força peso, fazendo o helicóptero perder altitude (figura 1.6).

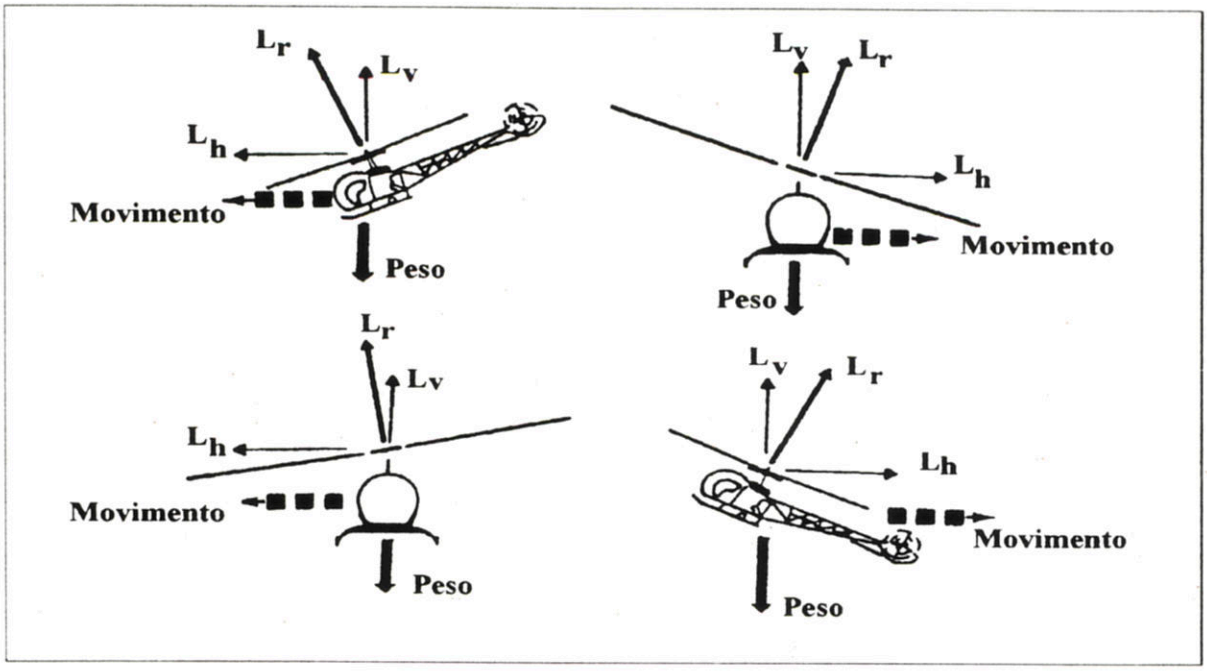


Fig. 1.5 - Ação dos comandos cíclico longitidinal e lateral. A direção da componente horizontal determina a direção do movimento

Considerando estes fatores, concluímos que o piloto ao atuar sobre o comando cíclico longitudinal deve atuar simultaneamente no comando coletivo para evitar a perda de altitude do helicóptero.

Os movimentos laterais possuem um comportamento semelhante.

Além da maior dificuldade de pilotagem devida aos acoplamentos, o piloto deve considerar os efeitos naturais que causam perturbações no movimeto da aeronave, como é o caso de rajadas de vento, por exemplo.

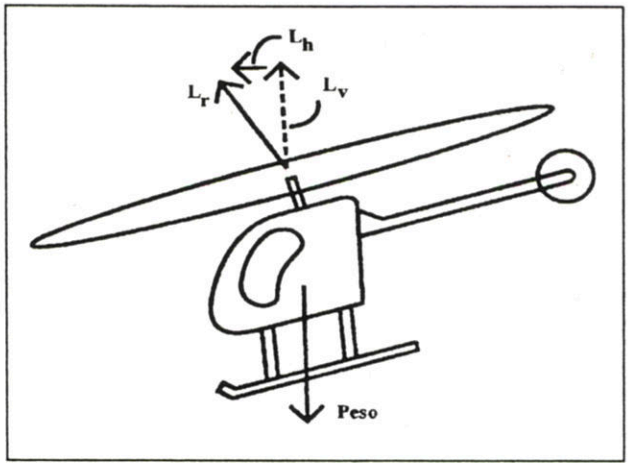


Fig. 1.6 - A componente vertical da sustentação compensa o peso, enquanto a componente horizontal produz aceleração a frente

A solução dos problemas de acoplamento e rejeição de perturbações foi favorecida, neste trabalho, pela estrutura do controlador nebuloso, onde cada comando do helicóptero é gerado de forma independente através de processos. Cada máquina de inferência do controlador nebuloso tende a buscar o seu ponto de referência e, desta forma, a mudança de referência para uma destas máquinas não interfere na atitude das outras máquinas.

1.5 CONCLUSÕES

Neste capítulo foram descritos alguns rudimentos sobre o vôo de helicópteros. Foram apresentados componentes aerodinâmicos básicos da aeronave, tipos de movimentos possíveis, e questões de acoplamento.

No próximo capítulo serão apresentados fundamentos da teoria da lógica nebulosa e de controladores nebulosos, completando a parte de fundamentos teóricos para este trabalho.

CAPÍTULO 2

CONTROLADORES NEBULOSOS

Neste capítulo são apresentados os fundamentos da lógica nebulosa e de controladores nebulosos. Esta apresentação tem por objetivo a compreensão do funcionamento e a familiarização com a terminologia de controladores nebulosos.

Controladores nebulosos são controladores que utilizam a mesma estratégia de controle utilizada por operadores humanos, ou seja, suas decisões são tomadas com base num conjunto de regras lingüísticas onde, de acordo com a situação do sistema, são executadas determinadas ações. Sua utilização se deve ao fato de que, em alguns casos, operadores humanos oferecem melhores resultados do que as estratégias automáticas de controle. É o que ocorre em sistemas complexos, com grandes não-linearidades ou com perturbações muito intensas.

Inicialmente são mostrados conceitos fundamentais da teoria de lógica nebulosa, em que consiste, e quais suas principais características. Em seguida é apresentado o controlador nebuloso, descritos quais os seus componentes básicos e dada uma visão das fases que envolvem um projeto de controladores nebulosos.

2.1 LÓGICA NEBULOSA

A lógica clássica lida com proposições que podem ser "verdadeiras" (valor lógico 1) ou "falsas" (valor lógico 0), por isso é chamada também de lógica bivalente. A lógica clássica, porém, possui restrições no que diz respeito a fatos futuros ou a fatos não precisos, que não podem por enquanto ser classificados nem como verdadeiros nem como falsos.

Por esse motivo, a lógica bivalente foi expandida para a lógica trivalente que possui um terceiro valor lógico (1/2) que indica "é possível". Novas regras dos operadores foram definidas, mas nos valores extremos (0 e 1) a lógica trivalente é coerente com a lógica bivalente [Lukasiewicz 75].

A partir dessa expansão, foi feita uma generalização definindo a lógica polivalente, onde há graus de possibilidade, chamados graus de verdade [Lukasiewicz 75]. Quando esses graus de verdade podem assumir qualquer valor no intervalo fechado [0,1], está sendo utilizada a lógica infinitamente polivalente [Klir and Folger 88].

A lógica nebulosa é um tipo de lógica infinitamente polivalente, onde os coeficientes de pertinência de um elemento u a um conjunto A podem ser interpretados como os graus de verdade da proposição "u pertence ao grupo A" [Klir and Folger 88]. Ela usa declarações

graduadas ou qualificadas em vez daquelas que são estritamente verdadeira ou falsa [Zadeh 84], [Self 90].

Os termos (constantes ou variáveis) utilizados em lógica nebulosa pertencem a classes cujos contornos não são precisamente delimitados. Os elementos ***u*** que pertencem a uma classe possuem graus (coeficientes) de pertinência, $\mu(u)$, no intervalo [0,1], que estabelecem quão bem o objeto é compatível com o conceito representado por esta classe. Um conjunto nebuloso é definido por seus elementos e respectivos coeficientes de pertinência [Sugeno and Kang 86], [Pedrycz 89], [Lee 90a].

Ex. 2.1: Tomando como exemplo o conjunto nebuloso de *pessoas altas*, pode-se definir que a partir de 1,80m, qualquer pessoa é considerada alta (coeficiente de pertinência ao conjunto igual a 1). Já as pessoas com menos de 1,60m não são consideradas altas (coeficiente de pertinência igual a 0). As pessoas de altura entre 1,60m e 1,80m possuem coeficientes de pertinência entre 0 e 1, sendo que este coeficiente representa o grau com que essas pessoas podem ser consideradas altas. Um possível conjunto nebuloso *pessoas altas* está representado na figura 2.1a.

Ex. 2.2: Atendo-se ao caso do helicóptero, conjuntos nebulosos são usados para classificar erros de posição, erro de altitude ou ângulos de atitude. Um exemplo seria o conjunto nebuloso *erro de altitude grande*, com erro de altitude ($\mathbf{E_h}$) definido por:

$$\mathbf{E_h = H_d - h},$$

onde $\mathbf{H_d}$ é a altitude desejada e **h** é a altitude atual. O conjunto nebuloso *erro de altitude grande* caracteriza erros de altitude entre 20 e 3000 metros como sendo verdadeiramente grandes (μ_g=1). Erros de altitude de até 10 metros não são considerados grandes (μ_g=0), e os erros de altitude no intervalo entre 10 e 20 metros são classificados como grandes segundo a equação:

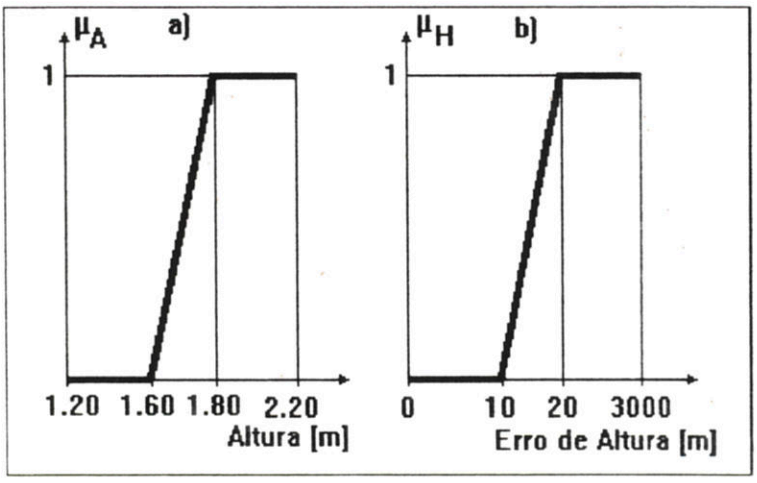


Fig. 2.1 - a) Conjunto Nebuloso Pessoas Altas; b) Conjunto Nebuloso Erro de Altitude Grande

$$\mu_g = \mathbf{E_h}/10 - 1.$$

O formato do conjunto *erro de altitude grande* é mostrado na figura 2.1b.

a) UNIVERSO DE DISCURSO

Define-se como *universo de discurso* $\boldsymbol{U}$ o conjunto genérico de objetos onde a função de pertinência de um objeto u a um conjunto nebuloso é definida [Lee 90a]. O *suporte* $\boldsymbol{S}$ de um conjunto nebuloso é definido como o conjunto dos valores do universo de discurso cuja função de pertinência possua valor maior que zero [Dubois and Prade 88],

$$\mu(u) > 0, u \in \boldsymbol{U},$$

e o *núcleo N* como o conjunto de valores do universo de discurso cuja função de pertinência é igual a 1:

$$\mu(u) = 1, u \in \boldsymbol{U}.$$

No exemplo 2.1 de altura das pessoas, o universo de discurso se restringe às pessoas com altura entre 1,20m e 2,20m, supondo que não existem pessoas adultas com altura fora desse intervalo. Já no exemplo 2.2 do erro de altitude do helicóptero, o universo de discurso é o intervalo de 0 a 3000 metros de altitude.

b) VARIÁVEL LINGÜÍSTICA

Uma variável lingüística é caracterizada por uma quíntupla (u, T(u),U,G,M) onde u é o nome da variável; T(u) é o conjunto de termos de u, onde termos são os nomes dos valores lingüísticos de u, cada um representando um conjunto nebuloso; U é o universo de discurso onde os conjuntos são definidos; G é uma regra sintática para gerar os nomes dos termos de u; e M é uma regra semântica que associa a cada termo T(u) seu significado [Lee 90a].

Ex. 2.3: A variável lingüística *erro_de_altitude* é caracterizada pela quíntupla:

u = *erro_de_altitude*;

T(u) = *pequeno*, *médio*, *grande*;

U= erro (diferença) entre altitude desejada e altitude atual, medida em metros, com valores entre 0 e 3000 metros;

G = os conjuntos obedecem a uma ordem crescente de *pequeno* para *grande*;

M = *pequeno* significa que a aeronave está a uma altitude bem próxima da desejada; *médio* significa que a aeronave está relativamente próxima da altitude desejada; *grande* significa que a aeronave está bem distante da altitude desejada.

2.2 SISTEMA DE REGRAS DE PRODUÇÃO

Uma forma usual de modelar a tomada de decisões humana é através de um conjunto de regras com condições e ações associadas.

Quando variáveis lingüísticas são utilizadas como antecedentes (onde estão agrupadas declarações a respeito das variáveis de entrada ligadas por conectivos lógicos 'e' e 'ou') e conseqüentes (onde está indicado o valor da saída associado a essa combinação de declarações no antecedente) de uma regra, está sendo estabelecida uma relação entre os universos de discurso dessas variáveis.

O conhecimento do especialista, expresso por um conjunto de regras, pode ser representado por uma tabela de regras. Essa representação é descrita a seguir, em particular para o exemplo a seguir.

Ex. 2.4: Conjunto de regras para controle da altitude do helicóptero:

- *SE erro_de_altitude é pequeno ENTÃO comando_coletivo é pequeno;*
- *SE erro_de_altitude é médio ENTÃO comando_coletivo é médio;*
- *SE erro_de_altitude é grande ENTÃO comando_coletivo é grande.*

2.2.1 TABELA DE REGRAS LINGÜÍSTICAS

Uma forma prática de representar um conjunto de regras que utilizam as mesmas variáveis é uma tabela de regras. Nesta tabela as colunas e linhas são as declarações nos antecedentes das regras, enquanto os elementos que formam a tabela são as saídas correspondes a cada combinação desses antecedentes. Essa tabela também é chamada tabela de decisão ou tabela de busca [Rutherford and Bloore 76].

O tamanho da tabela depende do número de variáveis e do número de conjuntos por variável. O formato da tabela para o conjunto de regras do exemplo 2.4 é mostrado na figura 2.2a, onde **E** é o erro de altitude, **ΔC** é a variação do comando coletivo a ser aplicado no helicóptero e **Peq**, **Méd** e **Gde** são os conjuntos nebulosos mostrados na figura 2.2b.

É importante salientar que cada elemento da tabela corresponde a uma regra. Assim, se não houver sido declarada uma regra para uma condição das variáveis de entrada, o elemento da

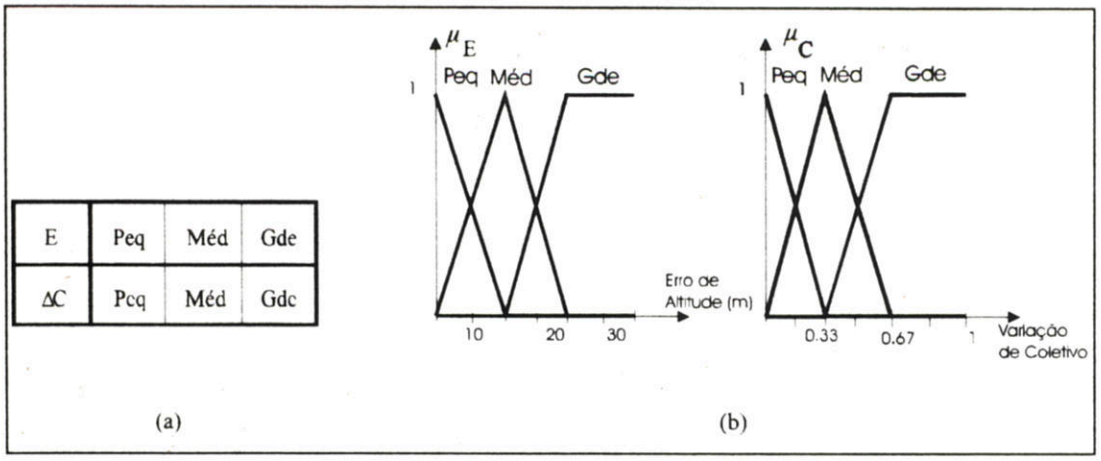


Fig. 2.2 - Tabela de Busca e Conjuntos Nebulosos para o Conjunto de Regras do Exemplo 2.4

tabela será indefinido, podendo ser utilizado, se necessário, um valor padrão ou um dos valores mais próximos. Se houver mais que uma regra para a mesma condição, o elemento da tabela terá mais que um valor possível, sendo utilizado algum critério, técnico ou econômico, para escolher um entre os valores possíveis [Braae and Rutherford 79b], [Aliev et al 91].

2.2.2 INFERÊNCIA UTILIZANDO REGRAS NEBULOSAS

Após a classificação das variáveis nos conjuntos nebulosos, as regras são avaliadas, utilizando a função de implicação para a atribuição do peso da declaração no conseqüente de cada

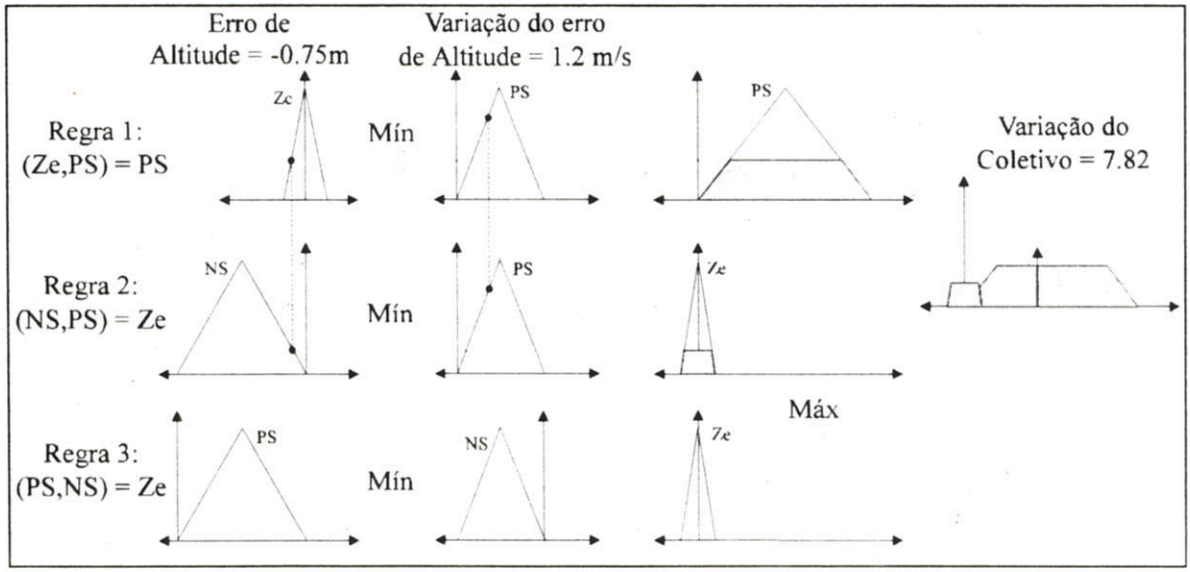


Fig. 2.3 - Exemplo de inferência utilizando regras nebulosas

regra. A função de implicação é aplicada às declarações no antecedente da regra.

Quando mais que uma regra possui uma mesma atribuição no conseqüente, a função de composição é responsável por determinar qual peso deve ser dado a essa declaração.

Na figura 2.3 é mostrado um exemplo de inferência utilizando regras nebulosas. Três regras são avaliadas, a classificação nos conjuntos das variáveis de entrada sendo mostrada abaixo do nome e do valor da variável. A função de implicação aqui utilizada é o mínimo, e a função de composição é o máximo.

O valor de atuação é obtido a partir da figura formada pelas saídas das regras utilizadas através do método do centro de massa.

2.3 CONTROLADOR NEBULOSO

A origem do desenvolvimento da teoria de controle nebuloso em malha fechada baseia-se na consideração de que o operador humano em muitos casos oferece melhores resultados que um controlador automático, principalmente para sistemas complexos, como é o caso de helicópteros.

A estratégia de controle de um operador é formada por um conjunto de regras de decisão, cuja forma depende, essencialmente, do processo sob controle e da heurística utilizada. O controlador nebuloso então necessita de um procedimento que permita a conversão da estratégia de controle lingüístico, baseada no conhecimento do especialista, em uma estratégia de controle automática [Lee 90a].

A literatura mostra que, na maior parte dos casos, os resultados obtidos com um controlador nebuloso são melhores que aqueles obtidos com os algoritmos de controle convencionais [Tang and Mulholland 87], [Li and Lau 89]. Em particular, a metodologia de controle nebuloso parece útil quando os processos são muito complexos para serem analisados pelas técnicas quantitativas convencionais ou quando as fontes de informação são julgadas não-precisas ou incertas. Assim, o controle por lógica nebulosa pode vir a ser uma aproximação entre o controle matemático preciso e a tomada de decisão humana.

Devido à simplicidade de implementação e ajuste, e da não necessidade do modelo matemático preciso do processo, o controle nebuloso vem sendo mais e mais aplicado em vários campos (médico, industrial, etc.), qualquer que seja a complexidade do sistema.

No entanto a falta de metodologia para projeto e ajustes tornam essas tarefas muito dependentes da sensibilidade que o projetista possui a respeito do processo. Se este for muito complexo, a escolha das variáveis utilizadas no controlador podem dificultar o ajuste.

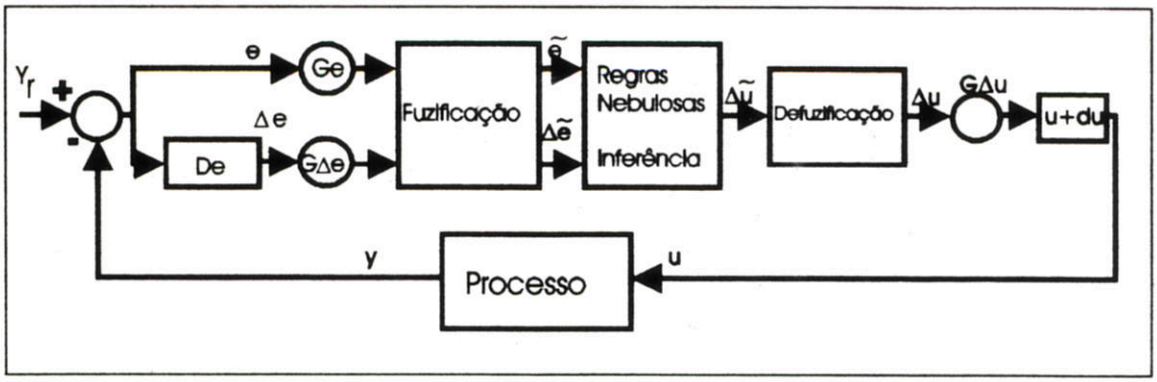


Fig. 2.4 - Esquema de Funcionamento de um Controlador Nebuloso

2.3.1 COMPONENTES DE UM CONTROLADOR NEBULOSO

A maior parte dos controladores desenvolvidos utilizam o esquema simples, representado na figura 2.4, proposto por Mandani [King and Mandani 77] para os sistemas monovariáveis. O valor do controle correspondente a uma dada situação pode ser obtido seguindo as etapas [Ketata 92]:

- cálculo do erro atual (**E** = **Yr** - **Y**) e sua variação (**ΔE**), onde **Y** é a saída do processo e **Yr** é a referência que a saída deve alcançar ;
- multiplicação desses valores por ganhos **GE** e **GΔE**, se forem utilizados conjuntos normalizados;
- conversão dos valores obtidos de **E** e **ΔE** em variáveis nebulosas (níveis de quantificação), ou fuzificação;
- inferência utilizando regras e obtenção do valor nebuloso da saída;
- multiplicação desse valor inferido por um ganho **GΔU**, se forem utilizados conjuntos normalizados;
- cálculo da entrada determinística para regular o processo, ou defuzificação.

A figura 2.5 mostra o mesmo controlador nebuloso mostrado na figura 2.4, enfatizando os blocos que o compõem e o tipo de dados (nebulosos ou não nebulosos) que cada bloco usa [Lee 90a], [Aptronix 93]. Esses blocos são os seguintes:

- ***interface de fuzificação*** é responsável pela conversão dos dados de entrada (variáveis medidas no processo e que determinam seu estado) em valores linguísticos adequados (coeficientes de pertinência);

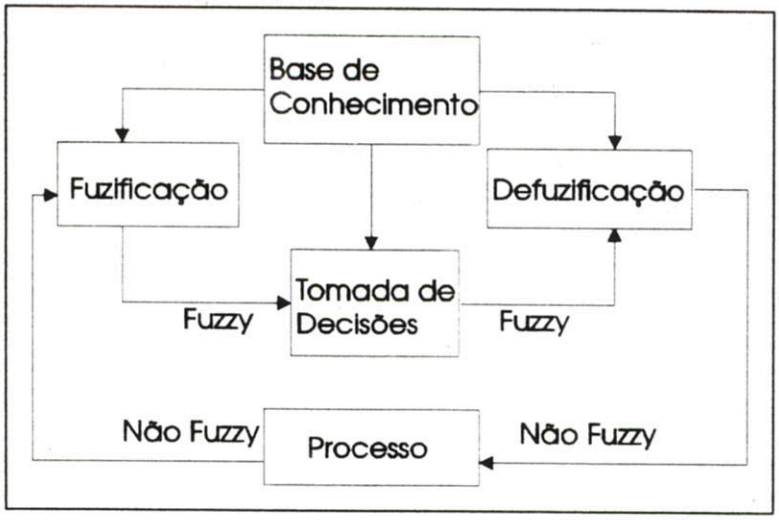


Fig. 2.5 - Componentes de um Controlador Nebuloso

- ***base de conhecimento*** consiste na base de dados e na base de regras de controle. A base de dados fornece as definições necessárias para manipulação dos dados nebulosos. A base de regras caracteriza os objetivos e a política de controle por meio das regras de controle;

- ***lógica de tomada de decisões*** tem a capacidade de simular o comportamento humano na tomada de decisões e de inferir ações de controle utilizando as regras da base de regras;

- ***interface de defuzificação*** tem a função de extrair uma ação de controle determinística (não nebulosa) do conjunto nebuloso inferido a partir das regras.

Cada um desses componentes possui parâmetros que devem ser determinados no momento do projeto, como veremos na próxima seção.

2.3.2 PROJETO DE UM CONTROLADOR NEBULOSO

Não existe uma metodologia sistemática para projeto de um controlador nebuloso, mas, de acordo com a estrutura do controlador mostrada na figura 2.5, geralmente é seguida a seguinte seqüência de passos [Self 90], [Bernard 88]:

- definir as variáveis de entrada e de saída do sistema;

- dividi-las em conjuntos nebulosos e estabelecer suas funções de pertinência;

- determinar as regras a serem utilizadas inicialmente e os parâmetros de inferência;

- testar e ajustar o controlador.

Em cada um desses passos, existem parâmetros a serem estabelecidos ou ajustados de acordo com o que houver disponível sobre o sistema (modelo linguístico ou operador humano experiente).

a) VARIÁVEIS DE ENTRADA E SAÍDA [LEE 90A]

As variáveis de entrada e saída dependem do processo a ser controlado e das grandezas disponíveis para medição. A escolha deve considerar também as variáveis que o operador humano utiliza. As variáveis mais usadas são a saída do sistema, erro entre a saída desejada e a saída atual, taxa de variação do erro e somatório do erro. Em [Buckley 91], é proposto um controlador nebuloso que possui tanto variáveis de entrada e saída determinísticas (sensores e atuadores) quanto nebulosas, que caracterizariam uma iteração do controlador com um operador humano, quando a interferência deste for necessária.

b) DIVISÃO DOS UNIVERSOS DAS VARIÁVEIS EM SUBESPAÇOS

A escolha do número e forma das funções de pertinência para cada variável influi em quanto o sistema pode ser considerado incerto ou nebuloso. A divisão dos espaços das variáveis em subespaços está definida na tabela do modelo lingüístico da planta, assim como o operador humano, na descrição do seu conhecimento, associa termos às variáveis. Cabe ao projetista escolher a função e a discretização dos conjuntos nebulosos correspondentes a cada subespaço, sendo que não há solução única e o método mais usado é o de tentativa e erro. Inicialmente, recomenda-se escolher um número ímpar de conjuntos, simétricos em relação ao zero, utilizando-se o menor número possível.

É preciso assegurar a **integridade** da base de dados, certificando-se que os universos de discurso dos conjuntos nebulosos cobrem toda a faixa de valores possíveis das variáveis.

c) DEFINIÇÃO DA FUNÇÃO DE PERTINÊNCIA DOS CONJUNTOS NEBULOSOS

Há duas maneiras de se definir a função de pertinência dos conjuntos nebulosos:

- método numérico, se o universo de discurso é discreto. Consiste em se representar a função de pertinência por um vetor cuja dimensão depende do grau de discretização do universo de discurso;

- método funcional, se o universo de discurso é contínuo. Expressa a função de pertinência por uma função contínua $\mu(u)$.

d) Escolha da Estratégia de Fuzificação

A estratégia de fuzificação tem um importante papel na manipulação de dados do controlador. Um operador de fuzificação converte um valor real em um conjunto unitário (*fuzzy singleton*), ou seja, um conjunto nebuloso que é 0 em todo o universo, a não ser no valor medido, onde o coeficiente de pertinência é 1 [Zadeh 73].

Se o valor a ser fuzificado está sujeito à incerteza (ruído), o operador de fuzificação deve converter esse dado probabilístico em um número nebuloso. Por fim, existem dados híbridos, que envolvem tanto incerteza quanto aleatoriedade. O operador de fuzificação deve usar o conceito de números híbridos para gerar conjuntos nebulosos para esses dados [Lee 90a].

e) Escolha da Estratégia de Defuzificação

É desejado que a estratégia de defuzificação produza uma ação de controle não-nebulosa que melhor represente a distribuição de possibilidade de uma ação de controle nebulosa inferida. As mais usadas e citadas na literatura são [Lee 90a]:

- critério máximo, ponto no qual a distribuição de possibilidade alcança o valor máximo;
- média dos máximos, valor médio de todas as ações locais cujas funções de pertinência alcançam o máximo e
- centro de área, que gera o centro de gravidade da distribuição de possibilidade de uma ação de controle.

Em [Lee 90a] são citados alguns trabalhos de análise destas estratégias de defuzificação que mostram que o centro de área é o método que permite obter melhores resultados do ponto de vista de representar o raciocínio humano.

f) Regras de Controle

As regras de controle e o método de inferência formam a estratégia do controle. De acordo com o estado do sistema é derivada uma ação sobre o processo que o leva até a condição desejada. Estas regras podem ser estabelecidas a partir:

- da modelagem do controle feito por um operador experiente. Neste caso, o modelo do controle do operador pode ser encontrado com base na descrição das ações executadas por ele ou por observação de suas atitudes[Lee 90a] [Kickert and van Nauta Lemke 76], [Sugeno and Nishida 85], [Takagi and Sugeno 83], [Bernard 88];

- do modelo lingüístico do sistema[Lee 90a], [Takagi and Sugeno 83]. Neste caso, utiliza-se a tabela de busca da planta, que é uma tabela onde consta a saída (nebulosa) do sistema para as diversas combinações das variáveis de entrada;

- de algoritmos de auto-aprendizagem, ou algoritmos que aprendem e alteram as próprias regras baseados na experiência.

Um aspecto importante a considerar com relação ao projeto é o número de regras. A princípio, devem haver tantas regras quantas sejam necessárias para cobrir todas as combinações das variáveis e seus subconjuntos (***completeness***), de maneira que, em qualquer condição, exista uma regra dominante ou, no caso extremo, duas regras disparadas com igual peso. Mas a experiência do projetista pode reduzir tal número, se restringindo aos estados mais freqüentes [Lee 90a],[Bernard 88].

Outras importantes propriedades das regras de controle a serem verificadas são a **consistência** da base de regras (é preciso minimizar a possibilidade de contradição) e a **interação** entre elas, controlada pela escolha da função de implicação e dos operadores de composição [Lee 90a].

g) MECANISMOS DE INFERÊNCIA

Os mecanismos de inferência empregados em um controlador nebuloso são geralmente muito mais simples que aqueles usados em um sistema especialista típico, já que os conseqüentes das regras não fazem parte dos antecedentes.

A definição da **função de implicação** altera o modo como as relações entre os antecedentes das regras de controle e entre os antecedentes e os conseqüentes dessas regras são inferidas. Existem várias funções de implicação definidas na literatura. A escolha de qual utilizar depende do sistema e do controle desejado. Estudos comparativos tentam determinar a função que melhor reflita o significado linguístico da implicação [Mizumoto and Zimmermann 82], [Ruan and Kerre 93], [Altrock et al 92]. A função de implicação mais simples é a proposta por Mandani (min-max) onde é escolhido o mínimo coeficiente entre os antecedentes (operador lógico nebuloso AND), sendo que este coeficiente é atribuído a todos os conseqüentes da regra.

A definição do **operador composicional** relaciona-se com o modo como as diferentes regras serão compostas para gerar um único conjunto nebuloso. Neste trabalho o operador composicional utilizado é o *máximo*.

h) AJUSTES

Ajustar o sistema é fornecer a forma e a localização ótimas para as funções de pertinência e o melhor conjunto de regras. Este passo no projeto ocorre durante a fase de simulação [Togai 92], [Bernard 88], [Braae and Rutherford 79a].

Um controlador nebuloso pode ser ajustado de três maneiras diferentes [King and Mandani 77]:

- modificando as regras de controle;
- modificando o universo de discurso e alterando os níveis de discretização;
- alterando o conjunto suporte dos conjuntos nebulosos (ampliando, reduzindo ou deslocando).

Neste último caso, a forma da função de pertinência é menos importante que sua superposição com outras funções, já que esta última é que determina as características de transição entre as funções e portanto entre regras ativas [Self 90]. [Krhöling 94] sugere uma superposição inicial de 10% a 50% do espaço do conjunto nebuloso, e, na literatura são usualmente adotadas superposições de 25%.

Segundo [Tong 76] uma alteração no número de conjuntos, nos níveis de discretização e no intervalo de amostragem não altera significativamente o desempenho do sistema em malha fechada. [Tong 76] indica ainda a modificação de regras como o método mais eficaz de alterar o desempenho de sistema.

2.4 CONCLUSÕES

Neste capítulo foi apresentada a lógica nebulosa como um meio de modelar o raciocínio humano considerando aspectos de imprecisão e incerteza. Foram mostrados os principais conceitos da lógica nebulosa, como conjuntos nebulosos, variáveis lingüísticas, regras de controle e mecanismos de inferência.

Foi apresentado também neste capítulo o controlador nebuloso, quais seus componentes básicos e qual o procedimento de projeto.

No próximo capítulo será apresentada a estrutura do sistema de navegação proposto em [Cav 94].

CAPÍTULO 3

ESTRUTURA DO SISTEMA DE NAVEGAÇÃO

Este capítulo tem por objetivo mostrar a estrutura do sistema de navegação do helicóptero não tripulado desenvolvido pela Gyron Tecnologia.

O objetivo deste sistema de navegação automático é substituir o piloto por um operador menos treinado no controle da aeronave, cumprindo missões como inspeção aérea de linhas de transmissão. Tais missões consistem em ordens ao helicóptero, com a função de conduzi-lo a uma posição ou movimento especificado. Por exemplo, para verificar uma linha de transmissão, o helicóptero deve inicialmente pairar a 5m do solo, sobre um dado ponto de coordenadas (x,y), e então percorrer um trecho de linha com uma velocidade de aproximadamente 10 m/s.

Para reproduzir a habilidade do piloto na condução do helicóptero, foi proposto em [Cav 94] um sistema especialista baseado em regras utilizando lógica nebulosa. Neste capítulo é descrita a estrutura básica deste sistema de navegação.

3.1 Estrutura do Sistema de Navegação

Como foi apresentado na introdução, o sistema de navegação tem o objetivo de substituir o piloto do helicóptero por um operador menos treinado no uso dos comandos, de maneira a executar uma missão desejada.

Uma missão consiste de movimentos específicos da aeronave, como decolar, pairar ou voar a frente, por exemplo. Esses movimentos são aqui chamados de "tarefas", possuindo cada uma parâmetros que exprimem a condição desejada que o helicóptero deve alcançar. Assim, uma missão do helicóptero começa com a decolagem e acaba com o pouso, passando por diferentes condições de voo nesse ínterim.

Uma possível missão seria:

1. decolar até uma altura especificada;

2. modificar azimute para 40°, leste;

3. pairar por 10 minutos na posição (X,Y,Z);

4. pousar.

Onde cada item numerado constitui uma tarefa com parâmetros adequados para o movimento que essa tarefa especifica. Assim, a tarefa decolar possui como parâmetro a altura a

ser alcançada a partir do chão, enquanto a tarefa pairar possui a posição espacial em que deve permanecer pairando e o tempo que deve permanecer assim.

Essa divisão da missão em tarefas segue a divisão natural efetuada pelo piloto. Dada uma missão, é necessário escolher as tarefas que ele executará para cumpri-la. Na execução de cada uma dessas tarefas, o piloto determina que controles utilizar e quais técnicas de pilotagem aplicar. A aplicação dessas técnicas controla a posição ou atitude do helicóptero, fazendo com que ele se mova como desejado.

O sistema de navegação está estruturado de modo a conter os níveis de interpretação da missão, de execução das tarefas e de controle.

O nível de interpretação da missão tem como função identificar a missão especificada, ativando as tarefas a serem executadas individualmente.

O nível de execução de tarefas encarrega-se de executar as tarefas incluídas no nível de interpretação de missão.

Uma tarefa corresponde a um movimento específico do helicóptero, como por exemplo, decolar, pousar, voar a frente ou mudar o azimute (ou ângulo de guinada), possuindo parâmetros, ou valores de referência a serem alcançados.

A execução de uma tarefa inclui os seguintes aspectos:

- ativação da base de conhecimento que o controlador nebuloso irá utilizar para executar o movimento especificado na tarefa;
- o critério que determina o fim de uma tarefa (critério de regime permanente).

O nível de execução de tarefas deve incluir a base de conhecimento (variáveis, conjuntos e regras) que o controlador nebuloso deve utilizar para alcançar o movimento desejado da tarefa. Sendo assim, cada tarefa possui uma base de conhecimento definida de acordo com o seu movimento.

O nível de controle consiste de um algoritmo de controle nebuloso que, a partir de uma base de conhecimento estabelecida no nível de execução de tarefa, utiliza os controles do helicóptero para levá-lo a condição de vôo desejada.

Esse controlador utiliza como variáveis de entrada os valores dos sensores e como variáveis de saída os valores inferidos dos controles do helicóptero. A estrutura do controlador nebuloso em blocos é apresentada na figura 3.1.

3.2 ACOPLAMENTOS

Um determinado movimento do helicóptero pode acarretar conseqüências em outros movimentos, conforme explicado no capítulo 1. Como cada uma das tarefas é executada como um processo independente, a medida que uma determinada tarefa detecta um afastamento do seu ponto de referência, ela tenta corrigir esta situação. Como um exemplo pode-se citar o caso em que o helicóptero encontra-se pairando a uma altitude constante. Nele estão agindo, por exemplo, a tarefa decolar que o mantém a uma altitude constante e a tarefa guinada que o mantém orientado em um determinado ângulo de guinada. Ao ser determinada uma tarefa de vôo a frente, a tendência do helicóptero seria uma perda de altitude, devido à redução no módulo do vetor sustentação. Quando o helicóptero se afasta da altura de referência, a tarefa decolar encarrega-se de corrigir esse afastamento atuando sobre o comando coletivo do rotor principal. Sendo assim o helicóptero passa a fazer um vôo a frente a uma altura constante. Este procedimento resolve o problema gerado pelos acoplamentos entre os movimentos do helicóptero.

3.3 ESTABILIDADE DA TAREFA

A tarefa correntemente sendo executada visa alcançar uma posição ou condição de vôo própria. Ao alcançar essa condição, a próxima tarefa estipulada pelo nível de interpretação de missão deve ser executada.

A maneira de determinar se o objetivo da tarefa foi alcançado deve ser função do erro entre o valor de referência e o valor atual da saída. A precisão desejada a partir das especificações também deve ser levada em consideração.

Bastaria, portanto, monitorar a saída e verificar que o erro entre a referência e a saída atual esteja dentro da precisão mínima especificada. O intervalo de tempo que se deve monitorar,

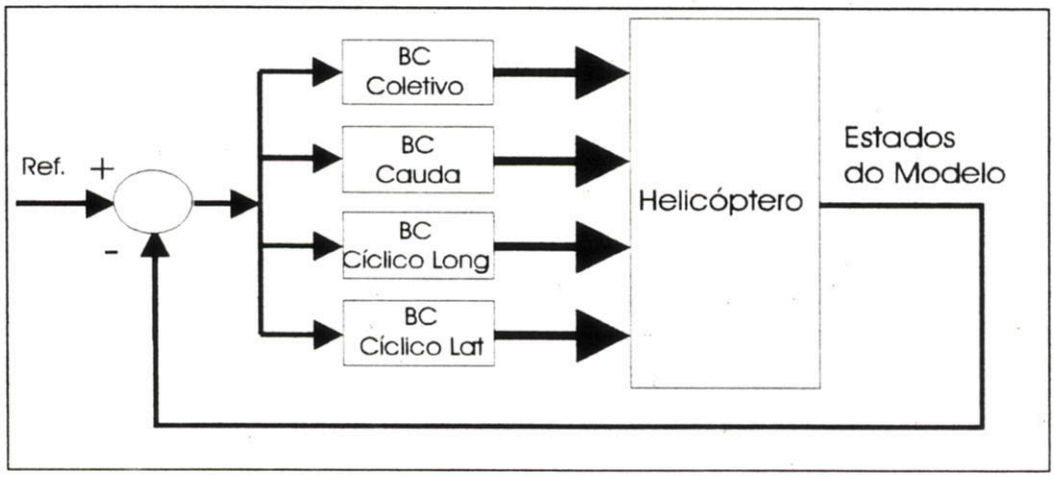


Fig. 3.1 - Estrutura do controlador nebuloso em blocos

porém, deve ser suficientemente longo para garantir que o regime permanente foi atingido, mas não tão grande que provoque atrasos desnecessários ao início da execução da próxima tarefa.

Esse intervalo ótimo de monitoramento é função da rapidez de resposta do helicóptero a cada um de seus comandos. Neste trabalho, é utilizado um tempo proporcional ao intervalo de tempo que o sistema leva para alcançar 36.8% do erro inicial, ou seja:

$$t = 3(t2 - t1),$$

onde t é o intervalo de monitoramento; t1 é o tempo de início da execução da tarefa e t2 é o tempo que o helicóptero atinge 36.8% do erro entre o valor de referência e o valor inicial da variável controlada. Este intervalo de tempo corresponde à constante de tempo dominante do helicóptero [Ogata 82].

3.4 Conclusões

Este capítulo descreve a estrutura do sistema de navegação dividida em níveis.

Esses níveis obedecem a ordem hierárquica de interpretar uma missão, executar as tarefas que compõem a missão e controlar a posição e a atitude do helicóptero de acordo com a tarefa. No nível de controle foi apresentado um controlador nebuloso, cuja estrutura e implementação são descritas nos próximos capítulos.

CAPÍTULO 4

DESCRIÇÃO DO CONTROLADOR NEBULOSO

Neste capítulo é descrita a estrutura projetada para o controlador nebuloso, consistindo da escolha dos mecanismos de inferência e estruturação da base de conhecimento.

São descritos também os objetivos, a técnica de pilotagem, as variáveis de entrada e saída, seus respectivos conjuntos nebulosos, considerações a serem feitas para a base de regras, acoplamentos e precisão de cada uma das tarefas que compõem a missão.

4.1 INTRODUÇÃO

O sistema de navegação está dividido em três níveis hierárquicos: nível de missão, nível de tarefa e nível de controle, conforme visto no capítulo 3. O nível de controle tem como função levar o helicóptero ao estado desejado pela tarefa atualmente em execução. Neste nível está implementado o controlador nebuloso, assunto deste capítulo.

Conforme discutido no capítulo 2, o projeto de um controlador nebuloso consiste em escolher os mecanismos de inferência e determinar a base de conhecimento. A inferência diz respeito a como os dados e regras são manipulados para a obtenção do valor apropriado da saída e compreende os métodos de fuzificação e defuzificação e funções de implicação e composição. A base de conhecimento agrega o conhecimento do especialista, incorporando o objetivo do controle, e consiste em determinar as variáveis, conjuntos e regras. A captura de conhecimento, o estudo e conseqüente escolha dos métodos de fuzificação e defuzificação e das funções de implicação e composição utilizadas neste trabalho foram objeto do trabalho [Cav 94].

Seguindo a estrutura hierárquica do sistema de navegação, é associada uma base de conhecimento do controlador nebuloso para cada tarefa. A seguir são descritas, para cada base de conhecimento, as variáveis de entrada e saída, os conjuntos nebulosos e as regras utilizadas, assim como o objetivo da tarefa e a técnica utilizada para realizá-la. Inicialmente, são apresentados os mecanismos de inferência utilizados.

4.2 MECANISMOS DE INFERÊNCIA

O mecanismo de inferência do controlador nebuloso diz respeito ao procedimento de lidar com os valores de entrada (fuzificação), disparar as regras da base de conhecimento, considerar as incertezas (funções de implicação e composição) e cálculo do valor de saída (defuzificação).

Foi escolhida a função *mínimo* como implicação, onde o grau de verdade do conseqüente de uma regra ativa é o mínimo entre as funções de pertinência dos antecedentes. A função de

composição é *máximo*, indicando que se regras com o mesmo conseqüente estão ativas simultaneamente, o máximo coeficiente de pertinência entre essas regras será usado para cálculo da saída defuzificada. Estas funções foram escolhidas devido a sua simplicidade e por serem as mais freqüentemente utilizadas na literatura [Viot 93], [Aliev et al 91], [Vachtsevanos 93], [Chiu et al 91].

Numa primeira implementação, todas as regras da tarefa eram avaliadas a cada iteração. Com a compilação de regras, o número de regras avaliadas em cada iteração diminuiu significativamente.

O método de defuzificação utilizado é o centro de área, já que, como comentado no capítulo 2, esse método fornece o melhor valor não nebuloso equivalente à distribuição de possibilidade da saída inferida pelo controlador nebuloso.

O mecanismo de inferência aqui descrito é único, sendo utilizado para todas as tarefas.

4.3 Base de Conhecimento

Como foi visto no capítulo 3, o nível de execução de tarefas estabelece o bloco de regras que deve ser usado pelo controlador. Neste bloco estão definidas as regras de controle, as variáveis de entrada e saída e os conjuntos nebulosos usados nestas regras.

O controlador, portanto, possui para cada tarefa executada, uma base de conhecimento independente com:

- as variáveis sobre as quais serão articuladas as ações de controle: variáveis de entrada do controlador (erros e variações destes erros);
- as variáveis de controle: variáveis de saída do controlador (comandos do helicóptero);
- os conjuntos nebulosos associados às variáveis de entrada e saída;
- as regras que, considerando o erro entre a referência e a saída atual, determinam ações que corrijam esse erro.

A seguir são descritos alguns aspectos do projeto da base de conhecimento das tarefas *Decolar*, *Guinada*, *Pairar* que correspondem às tarefas utilizadas a nível de simulação para o teste de desempenho do controlador.

4.3.1 TAREFA DECOLAR

a) OBJETIVO E COMANDOS

A tarefa *Decolar* consiste em tirar o helicóptero do solo, fazendo-o subir até uma altitude (Hr) desejada em metros.

Para alterar a velocidade vertical do helicóptero, deve-se atuar no comando coletivo, aumentando-o para fazer a aeronave subir e diminuindo-o para fazê-la descer.

b) TÉCNICA DE PILOTAGEM

O piloto altera o comando coletivo aos poucos, de forma acumulativa e proporcional ao erro de altitude, quando este não é muito grande. Quando o erro de altitude é muito grande, o piloto altera o comando inversamente proporcional à velocidade vertical do helicóptero. Neste caso, o piloto tenta conduzir o helicóptero até a altitude desejada, com velocidade constante e que não seja muito grande nem muito pequena (média).

c) VARIÁVEIS DE ENTRADA E SAÍDA

Acelerar o helicóptero para cima ou para baixo depende da altitude atual (**h**), ou mais especificamente, do erro de altitude (**Eh**), definido como:

$$E_h = H_r - h$$

e da variação do erro de altitude ($d\mathbf{E_h}/dt$), obtida derivando a equação acima:

$$\frac{dE_h}{dt} = -\frac{dh}{dt} = -V_z ,$$

onde $\mathbf{V_Z}$ é a velocidade vertical do helicóptero. A figura 4.1 apresenta a situação de decolagem do helicóptero, mostrando as variáveis envolvidas.

Assim o bloco coletivo da tarefa *Decolar* possui como variáveis de entrada o erro de altitude **Eh** e a variação de altitude $d\mathbf{E_h}/dt$. A variável de saída é a variação do comando coletivo, $\Delta\mathbf{C}$. Essa variação é integrada, após a defuzificação, para obter o comando a ser enviado ao modelo simulado do helicóptero. Isto é necessário porque a alavanca do coletivo possui uma posição de equilíbrio (Trim) variável com a altitude, que deve ser alcançada aos poucos. Essa integração, entretanto, possui a desvantagem de acrescentar retardo na resposta.

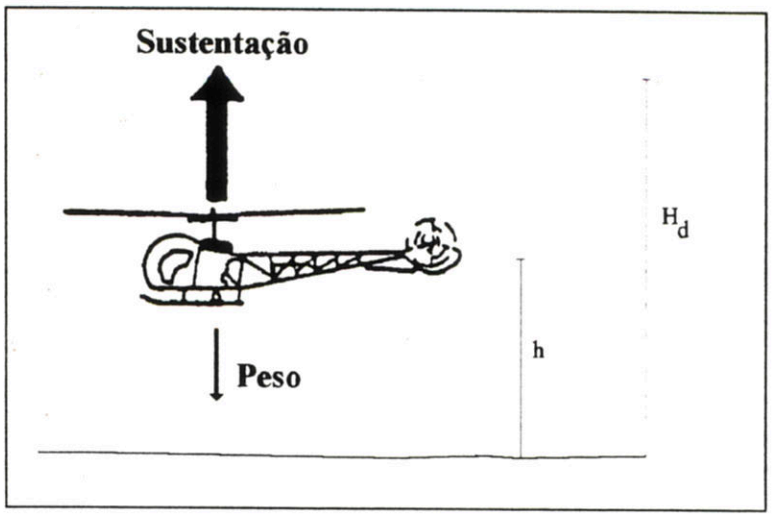


Fig. 4.1 - Decolagem do Helicóptero

d) CONJUNTOS NEBULOSOS

Os conjuntos nebulosos associados às variáveis de entrada são em número de sete: *Negative Big*, *Negative Medium*, *Negative Small*, *Zero*, *Positive Small*, *Positive Medium* e *Positive Big*, simétricos em torno de zero. Esse número permite razoável versatilidade em troca de uma relativa complexidade na formulação das 49 regras necessárias para cobrir todo o espaço de valores das variáveis. O formato dos conjuntos é triangular, exceto os conjuntos *Big*, que possuem forma trapezoidal, conforme mostrado na figura 4.2.

O conjunto *Zero* do erro é estreito, e representa a tolerância aceitável da altitude alcançada pelo helicóptero. O conjunto *Zero* da velocidade do erro também é bem estreito, indicando junto com erro *Zero* o estado de estabilidade do sistema.

O conjunto *Small* do erro representa a condição de proximidade da altitude escolhida. Nesta condição a velocidade de aproximação deve ser pequena (velocidade do erro *Small*), de maneira a evitar o sobrepasso, não ultrapassando a altitude desejada. Como se vê, os conjuntos *Small* se justificam para detecção do estado em que é necessário evitar o sobrepasso.

O conjunto *Medium* do erro é usado na maioria das situações, e representa um erro de

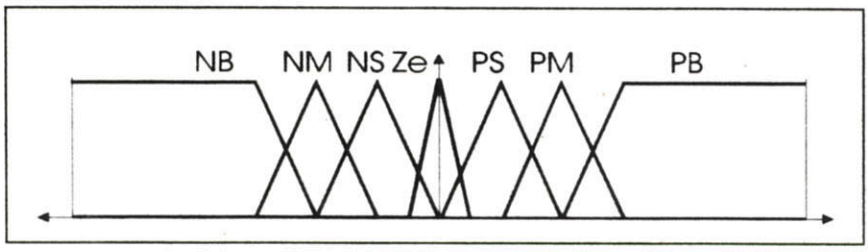


Fig. 4.2 - Formato dos Conjuntos Nebulosos Utilizados na Tarefa Decolar

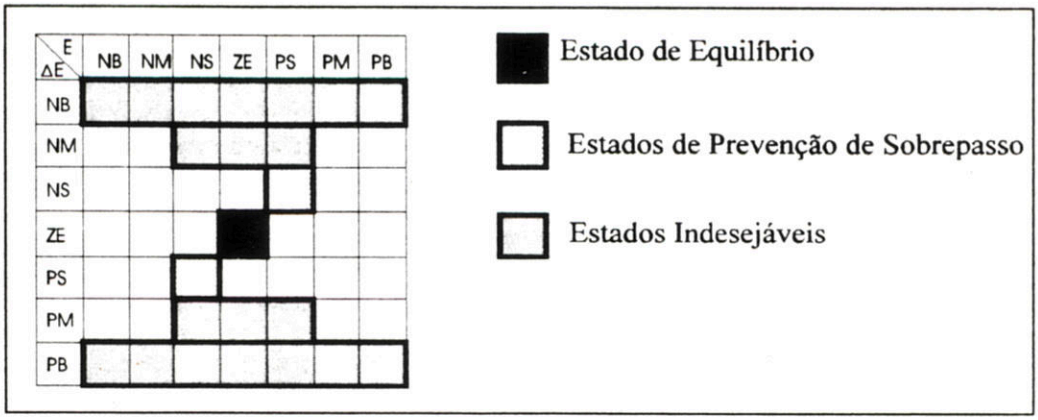


Fig. 4.3 - Estados na Tabela de Busca da Tarefa Decolar

altitude médio, não muito distante da altitude desejada. A velocidade do erro *Medium* expressa a velocidade de aproximação ideal; esta não dever ser tão pequena que acarrete atrasos, nem tão grande que ocasione perda de controle.

O conjunto de erro *Big* representa situações extremas, onde altitude do helicóptero se encontra longe da desejada. A velocidade do erro *Big* não é uma situação desejada, já que pode ocasionar descontrole da aeronave.

As situações que justificam o número de conjuntos nebulosos para as variáveis de entrada estão representadas na figura 4.3, onde vemos quais estados importantes (equilíbrio, prevenção do sobrepasso e estados proibidos) são determinados por esses conjuntos.

Os conjuntos nebulosos associados à variável de saída variação do coletivo são cinco: *Negative Medium*, *Negative Small*, *Zero*, *Positive Small* e *Positive Medium*. O conjunto *Zero* é bem estreito e representa a posição de equilíbrio (*Trim*) em que variação da saída é mínima. Como a atuação no comando é realizada através da variação e não de um valor absoluto, neste caso a saída caracterizada pelo conjunto *Zero* não significa que o comando do helicóptero será nulo, mas que a variação é nula, e o atual comando se mantém.

O conjunto *Small* significa uma pequena correção no comando, para corrigir erros pequenos ou médios. Apesar do inerente atraso na integração de valores pequenos, o uso dos conjuntos *Small* na atuação tenta evitar o sobrepasso e a perda de controle por variação brusca do comando.

O conjunto *Medium* só é usado em situações extremas de correção de movimentos em altas velocidades, nunca de posição, devido ao possível descontrole mencionado anteriormente.

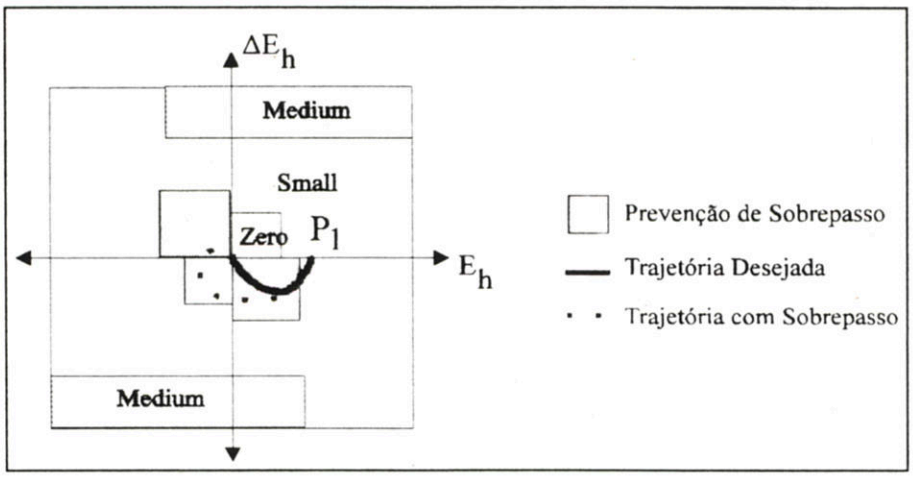


Fig. 4.4 - Regras e Trajetórias no Espaço de Estados

e) REGRAS

Na figura 4.4, pode-se ver qualitativamente o uso dos conjuntos de saída no espaço de estados (erro de altitude e variação do erro de altitude). A condição inicial é o ponto P_1, sendo a trajetória desejada a linha contínua e a trajetória com sobrepasso a linha pontilhada. A área hachurada representa a zona de prevenção de sobrepasso, onde nota-se que as trajetórias descritas acima se separam.

f) ACOPLAMENTOS

Nesta tarefa não foram considerados os acoplamentos entre o movimento vertical e os outros movimentos, já que o simulador utilizado não prevê situações de rajadas de vento ou de decolagem de terreno em declive, quando esses acoplamentos se tornam relevantes.

g) PRECISÃO

Para a tarefa *Decolar* a precisão escolhida para assegurar a estabilidade desta tarefa, e portanto iniciar a tarefa seguinte (conforme visto no capítulo 3) é de aproximadamente 10cm.

4.2.2 TAREFA GUINADA

a) OBJETIVO E COMANDOS

O objetivo da tarefa *Guinada* é girar o corpo do helicóptero em torno do eixo vertical em um plano horizontal, de forma a alinhar a aeronave com a direção de vôo. Este movimento altera o ângulo de guinada do helicóptero, modificando o azimute da aeronave.

O comando coletivo do rotor de cauda, responsável por contrabalançar os efeitos de reação da fuselagem a rotação do rotor principal, permite a mudança do ângulo de guinada.

Quando o comando de cauda é aumentado, o nariz gira no mesmo sentido do rotor principal e quando é diminuído, o nariz gira no sentido contrário.

b) TÉCNICA DE PILOTAGEM

Para alterar o ângulo de guinada da aeronave, o piloto atua no comando de cauda, mantendo rígido controle da velocidade de mudança do ângulo. Essa velocidade não pode ser grande, sob risco de gerar um comportamento incontrolável da aeronave. Isso é importante principalmente quando o erro de ângulo de guinada é grande. A alteração no comando é feita de forma suave em função do erro, quando este é pequeno.

c) VARIÁVEIS DE ENTRADA E SAÍDA

A atuação no comando de cauda é função do erro de ângulo de guinada ($\mathbf{E_g}$), definido por:

$$E_g = G_r - g\,,$$

onde $\mathbf{G_r}$ é o ângulo de guinada desejado e **g** é o ângulo de guinada atual. A variação desse erro ($d\mathbf{Eg}/dt$) informa se ele está aumentando ou diminuindo, influindo também no comando de cauda a ser aplicado. Essa variação é calculada por:

$$\frac{dE_g}{dt} = -\frac{dg}{dt} = -V_g\,,$$

onde $\mathbf{V_g}$ é a velocidade de guinada do helicóptero. O ângulo de guinada é positivo no sentido horário e negativo no sentido anti-horário. A figura 4.5 mostra a vista superior do helicóptero quando está sendo alterado seu ângulo de guinada.

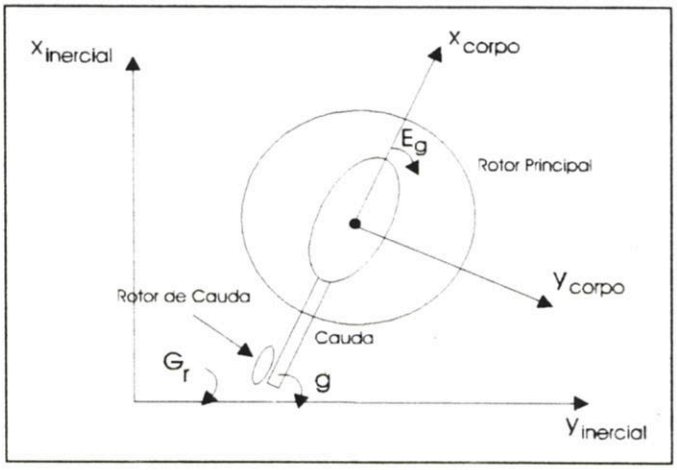


Fig. 4.5 - Vista Superior do Helicóptero durante a Alteração do Ângulo de Guinada

Portanto o bloco de comando de cauda da tarefa *Guinada* possui o erro de ângulo de guinada ($\mathbf{E_g}$) e sua variação ($d\mathbf{E_g}/dt$) como variáveis de entrada. A variável de saída utilizada é a variação do comando de cauda ($\Delta\mathbf{T}$), que é integrada após a defuzificação para a obtenção do comando a ser enviado ao simulador do helicóptero. Isto porque a alavanca do comando de cauda, assim como a do coletivo, também possui uma posição de equilíbrio (Trim) variável com o ângulo a ser alcançado (ou altitude no caso do coletivo) , e variações bruscas de comando podem causar perda de controle.

d) CONJUNTOS NEBULOSOS

Para estas variáveis do bloco de comando de cauda da tarefa *Guinada*, o mesmo número de conjuntos da tarefa *Decolar* foi utilizado, com o mesmo formato dos mostrados na figura 4.2. A justificativa é equivalente, com o reforço de que devido a atrasos de resposta, os conjuntos da variável de saída são mais largos, no sentido de reduzir o atraso adicional imposto pelo integrador.

e) REGRAS

As regras de controle para a tarefa *Guinada* foram projetadas da mesma maneira que as da tarefa *Decolar*, a não ser pelos sinais. As regras da tarefa *Guinada* são mais restritivas quanto à condição de impedir que a aeronave atinja velocidades *Big*, já que, para este movimento, grandes velocidades podem causar instabilidades.

f) ACOPLAMENTOS

A tarefa de alterar o ângulo de guinada possui acoplamentos com os movimentos vertical e horizontal. O acoplamento com o movimento vertical é corrigido pela ativação do bloco de regras que atua sobre o coletivo, o mesmo usado na tarefa *Decolar*.

O acoplamento com o movimento horizontal é corrigido, após a estabilização, com a ativação das regras da tarefa *Pairar*. Isto porque, após ter o nariz alinhado com a direção de vôo, o helicóptero pode pairar ou voar a frente. Ambas as tarefas corrigem esse acoplamento. Como a tarefa *Vôo a Frente* não está implementada ainda, a tarefa *Pairar* é obrigatória após alteração do ângulo de guinada.

g) PRECISÃO

Para essa tarefa, a precisão de 0.1 radianos foi utilizada para a determinação da estabilidade da tarefa *Guinada*.

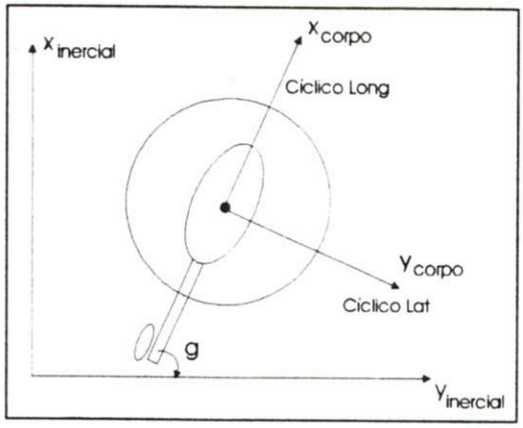


Fig. 4.6 - Movimento do Helicóptero com os Comandos Cíclicos

4.2.3 TAREFA PAIRAR

a) OBJETIVO E COMANDOS

A tarefa *Pairar* tem como objetivo imobilizar o helicóptero a uma altitude constante sobre um ponto fixo no chão.

No controle da altitude e direcionamento, atua-se nos comandos coletivo e de cauda, conforme explicado nas tarefas *Decolar* e *Guinada*.

Para controlar a posição do helicóptero sobre um ponto fixo no chão (posição XY), é necessário atuar nos comandos cíclicos lateral e longitudinal. Conforme explicado no capítulo 1, esses comandos deslocam o helicóptero para a esquerda/direita e para frente/para trás, respectivamente, conforme se vê na figura 4.6.

O controle de posição através dos comandos cíclicos ocorre na seguinte seqüência, ilustrada na figura 4.7:

1. a atuação nos comandos cíclicos causa uma rotação do corpo do helicóptero num plano

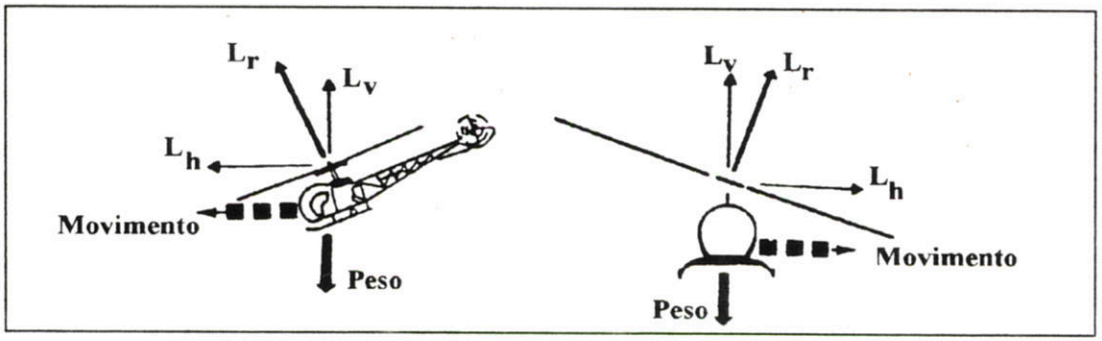


Fig. 4.7 -Arfagem do helicóptero no movimento longitudinal e rolagem no movimento lateral

vertical: arfagem no caso do comando longitudinal e rolagem no caso do comando lateral;

2. a rotação provoca a existência de uma componente horizontal da força de sustentação. Esta força causa uma aceleração na direção da inclinação;

3. a aceleração causa deslocamento nessa direção, com retardos devido à dupla integração.

Controlar a posição consiste, portanto, em comandar a inclinação do corpo do helicóptero na direção e sentido adequados a reduzir o erro. Essa inclinação, porém, deve ser limitada sob pena de causar perda de controle.

b) TÉCNICA DE PILOTAGEM

Para controlar altitude e ângulo de guinada, o piloto se comporta como descrito nas tarefas *Decolar* e *Guinada*, respectivamente.

No caso do controle de posição horizontal, o piloto atua na alavanca dos comandos cíclicos na forma de rápidos e pequenos pulsos em torno da posição de zero. A resposta aos pulsos possui os retardos descritos acima, e o piloto mantém a alavanca na posição zero, até que o helicóptero responda ao comando.

A velocidade é sempre baixa, pelo fato da tarefa *Pairar* lidar somente com pequenos erros em torno da posição desejada.

c) VARIÁVEIS DE ENTRADA E SAÍDA

Para executar esta tarefa são necessários quatro blocos, um para cada comando:

- o bloco de comando coletivo, para manter o helicóptero a uma altitude constante, que corresponde ao bloco utilizado na tarefa *Decolar*;
- o bloco de comando de cauda, para manter o direcionamento do helicóptero, que corresponde ao bloco utilizado na tarefa *Guinada;*
- os blocos de comandos cíclicos longitudinal e lateral, cuja atuação permite controlar a posição do helicóptero sobre um ponto fixo no chão (posição XY).

No projeto destes dois últimos blocos é preciso estudar o comportamento do helicóptero no plano XY.

Como se vê na figura 4.8, dada uma posição no solo desejada ($\mathbf{P_r}$), de coordenadas ($\mathbf{X_r}$,$\mathbf{Y_r}$), e a posição atual da aeronave (**p**), de coordenadas (**x**,**y**), deseja-se, através dos comandos cíclicos longitudinal e lateral, conduzir o helicóptero de **p** até $\mathbf{P_r}$.

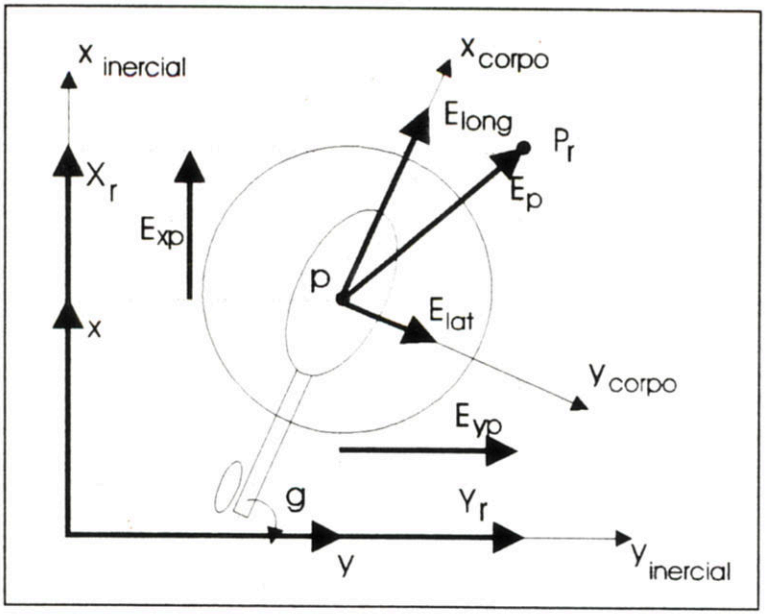


Fig. 4.8 - Vista Superior do Helicóptero Mostrando Variáveis para Controle da Posição XY

Seja:

$$E_p = P_r - p,$$

o erro entre a posição desejada e a posição atual do helicóptero. O objetivo é, portanto, obter:

$$E_p = 0,$$

As coordenadas de $\mathbf{E_p}$ ($\mathbf{E_{xp}}$,$\mathbf{E_{yp}}$), definidas por:

$$E_{xp} = X_r - x$$

$$E_{yp} = Y_r - y.$$

O comando longitudinal, quando aumentado, leva o helicóptero para frente, na direção apontada pelo nariz (ângulo de guinada **g**). Portanto, ele deve ser função de $\mathbf{E_{long}}$, que é a projeção de $\mathbf{E_p}$ no eixo X no corpo do helicóptero, conforme se vê na figura 4.8.

Já o comando lateral, quando aumentado, leva o helicóptero para a direita do corpo da aeronave. Ele, então, deve ser função de $\mathbf{E_{lat}}$, que é a projeção de $\mathbf{E_p}$ no eixo Y no corpo do helicóptero.

Equacionando, temos:

$$E_{long} = E_{xp} * \cos(g) + E_{yp} * \sin(g) \quad (1)$$

$$E_{lat} = -E_{xp} * \sin(g) + E_{yp} * \cos(g). \quad (2)$$

Como:

$$\frac{dE_{xp}}{dt} = -\frac{dx}{dt} = -V_x$$

$$\frac{dE_{yp}}{dt} = -\frac{dy}{dt} = -V_y$$

temos:

$$\frac{dE_{long}}{dt} = -V_x * \cos(g) - V_y * \sin(g) \quad (3)$$

$$\frac{dE_{lat}}{dt} = V_x * \sin(g) - V_y * \cos(g) \cdot \quad (4)$$

As equações (1), (2), (3) e (4) mostram como calcular $\mathbf{E_{long}}$, $\mathbf{E_{lat}}$, $d\mathbf{E_{long}}/dt$ e $d\mathbf{E_{lat}}/dt$, que são as variáveis de entrada dos blocos cíclico longitudinal e lateral, respectivamente.

Para evitar que a inclinação seja demasiada, é necessário colocar os ângulos de inclinação e suas variações: arfagem no caso do longitudinal e rolagem no caso do lateral.

Cada bloco, então, utiliza quatro variáveis de entrada: erro de posição, variação do erro de posição, ângulo de inclinação e variação do ângulo de inclinação.

A variável de saída desses blocos é o próprio comando, não sendo necessária a integração, porque as alavancas dos comandos cíclicos possuem uma posição de equilíbrio (Trim) fixa, os comandos consistindo de rápidos pulsos em torno de zero na direção do movimento desejado.

d) CONJUNTOS NEBULOSOS

Como os blocos possuem quatro variáveis de entrada, um número grande de conjuntos resultaria em uma grande quantidade de regras, aumentando a complexidade do controlador nebuloso, razão pela qual a quantidade de conjuntos por variável nessa tarefa foi diminuída.

Para as variáveis $\mathbf{E_{long}}$ e $\mathbf{E_{lat}}$, utiliza-se cinco conjuntos: *Negative Medium*, *Negative Small*, *Zero*, *Positive Small* e *Positive Medium*. O conjunto *Zero* indica a situação de equilíbrio, bem como a tolerância de erro em regime permanente. Os conjuntos *Small* indicam a situação de prevenção de sobrepasso, enquanto os conjuntos *Medium* servem para as correções de posição normais.

Para as variáveis $d\mathbf{E}_{\mathbf{long}}/dt$ e $d\mathbf{E}_{\mathbf{lat}}/dt$, bastam três conjuntos: *Negative*, *Zero* e *Positive*, para indicar a direção do movimento.

Os ângulos de inclinação (rolagem e arfagem) também utilizam três conjuntos *Negative*, *Zero* e *Positive*, assim como as variações dos ângulos. Esses três conjuntos bastam para prever a tendência de aumentar o ângulo descontroladamente.

e) REGRAS

Na figura 4.9, vê-se como as regras se distribuem na tabela de busca do controle do erro lateral. Os estados indesejáveis consistem de situações que indicam a rolagem crescendo

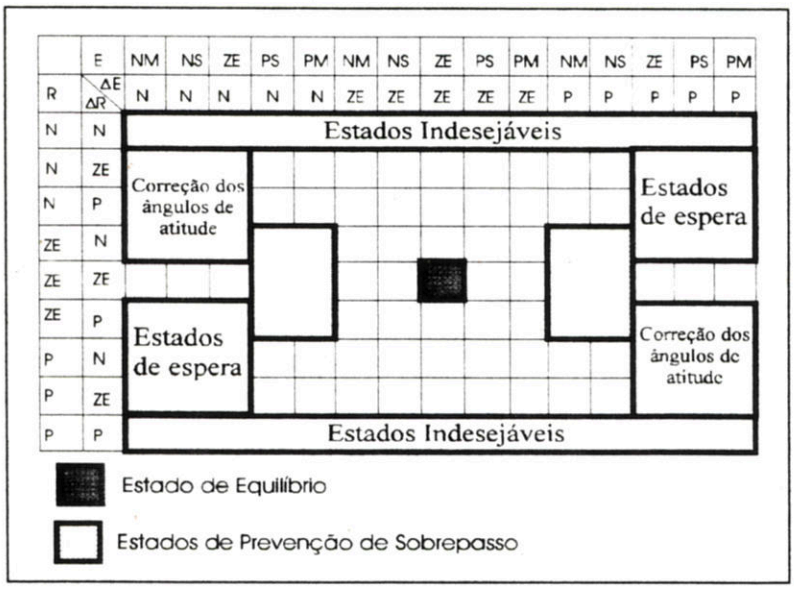


Fig. 4.9 - Tabela de Busca do Comando Cíclico Lateral da Tarefa Pairar

ilimitadamente. Os estados de prevenção de sobrepasso indicam a situação onde a trajetória desejada e a trajetória com sobrepasso divergem. Os estados de espera ocorrem quando já se atuou nos comandos, mas não houve resposta ainda devido os atrasos inerentes do helicóptero. Nestes estados o comando deve ser *Zero*.

f) ACOPLAMENTOS

Nesta tarefa, todos os comandos estão ativos através dos blocos de regras, sendo que quaisquer acoplamentos são simultaneamente corrigidos.

g) PRECISÃO

Esta tarefa prevê, para os blocos de comandos coletivo e de cauda, as mesmas precisões das tarefas *Decolar* e *Guinada*. Para os blocos de comandos cíclicos, a precisão utilizada é de 10 centímetros, considerada boa se comparada com a conseguida por um piloto, já que esta depende da distância do piloto ao helicóptero.

4.3 CONCLUSÕES

O nível de controle do sistema de navegação consiste em um controlador nebuloso, com mecanismo de inferência fixo e base de conhecimento variável de acordo com a tarefa correntemente sendo executada.

Foram descritas neste capítulo as características fixas (fuzificação, defuzificação e funções de implicação e composição) e aspectos do projeto da base de conhecimento das tarefas *Decolar*, *Guinada* e *Pairar*.

A implementação do controlador nebuloso, com descrição da missão e dos aspectos construtivos da máquina de inferência são descritos no próximo capítulo.

CAPÍTULO 5

IMPLEMENTAÇÃO DO CONTROLADOR NEBULOSO

Nos capítulos anteriores, foram descritos aspectos funcionais do helicóptero, teoria básica de controladores nebulosos, aspectos da estrutura do sistema de navegação e do controlador nebuloso.

Neste capítulo é descrita a implementação do controlador nebuloso numa plataforma baseada na linguagem Forth para Transputer.

Inicialmente são apresentadas algumas características da linguagem Forth e do transputer. Em seguida é apresentada a implementação da máquina de inferência e por último a implementação do controlador nebuloso.

5.1 A LINGUAGEM

A linguagem escolhida para a implementação do controlador nebuloso foi Forth. A linguagem Forth é uma linguagem interativa, interpretada, baseada em pilhas, e foi criada por Charles H. Moore.

Sua característica principal é a extensibilidade, ou seja, a facilidade que ela oferece para a criação de novas palavras no seu dicionário que passarão a fazer parte de uma biblioteca de instruções criadas diretamente para uma aplicação.

A velocidade de execução da linguagem é também uma de suas características favoráveis à implementação deste trabalho, mesmo se tratando de uma linguagem interpretada.

O interpretador utilizado neste trabalho foi um interpretador de forth para transputer desenvolvido pela Gyron Tecnologia. Este interpretador foi instalado numa plataforma montada no LCMI, sendo o mesmo interpretador utilizado no processador principal embarcado no helicóptero: o transputer.

5.2 O PROCESSADOR

O processador utilizado no sistema é o transputer T800 da INMOS.

Suas características principais são canais seriais de alta velocidade para a comunicação entre processadores, comunicação entre processos baseada em canais síncronos, distribuição do tempo entre os processos microcodificada, capacidade de processamento em ponto flutuante e possibilidade de aumentar o número de processadores utilizados em uma aplicação.

A característica do transputer que favorece a implementação das máquinas de inferência como processos independentes é sua capacidade natural de multiprogramação.

5.3 MÁQUINA DE INFERÊNCIA

Os trabalhos que culminaram com a implementação da máquina de inferência Forth para transputer foram desenvolvidos em duas etapas.

A fase inicial dos trabalhos foi a especificação e posterior implementação de um gerador e de um tradutor de tabelas de regras lingüísticas. O objetivo desta fase seria a criação de um mecanismo que permitisse a geração automática de parte do código a ser utilizado na implementação da máquina de inferência, a partir da tradução de uma tabela de regras lingüísticas.

Em seguida foi especificada a máquina de inferência para lógica nebulosa em forth seguida de sua implementação.

A seguir são apresentadas cada uma destas etapas, com a descrição da estrutura implementada em ambos os casos.

5.3.1 IMPLEMETAÇÃO DE UM GERADOR E DE UM TRADUTOR DE TABELAS DE REGRAS NEBULOSAS

A representação de regras lingüísticas através de tabelas proporciona facilidade de interpretação e melhor visibilidade das situações que podem ocorrer na execução de uma determinada tarefa, permitindo verificar, também, aspectos de tendências do controle para diversas situações. A tabela permite agrupar um conjunto de regras que possuem as mesmas variáveis de entrada para uma determinada variável de saída.

Traduzir a tabela seria gerar o conjunto de regras correspondente a partir de uma tabela preenchida de acordo com o conhecimento especialista obtido por meio de técnicas de aquisição de conhecimento. Este conjunto de regras resultante irá compor a base de regras de inferência, utilizado na operação da máquina de inferência.

Na figura 5.1 temos um exemplo de tabela para quatro variáveis de entrada (*erro lateral*, *variação do erro lateral*, *angulo de rolagem* e *variação do ângulo de rolagem*) para a saída *ciclico lateral*.

Cada situação possível para a variável de saída (*ciclico lateral)* está representada na tabela da figura 5.1.

	ER	pm	pm	pm	ps	ps	ps	ze	ze	ze	ns	ns	ns	nm	nm	nm
	DE	ps	ze	ns	ps	ze	ns	ps	ze	ns	ps	ze	ns	ps	ze	ns
RO	DR															
ps	ps	ZE	ZE	ZE	PM	NM	NM	NM	NM	NM	ZE	ZE	ZE	ZE	ZE	NM
ps	ze	ZE	ZE	ZE	ZE	**ZE**	ZE	ZE	ZE	NM	NM	NM	NM	ZE	ZE	ZE
ps	ns	ZE	ZE	PS	ZE	ZE	ZE	ZE	ZE	ZE	ZE	ZE	ZE	ZE	ZE	ZE
ze	ps	ZE	ZE	ZE	PM	ZE	ZE	PS	ZE	NS	ZE	ZE	NM	ZE	ZE	ZE
ze	ze	ZE	PS	ZE	PM	ZE	ZE	ZE	ZE	ZE	ZE	ZE	NM	ZE	NS	ZE
ze	ns	ZE	ZE	ZE	PM	ZE	ZE	PS	ZE	NS	ZE	ZE	NM	ZE	ZE	ZE
ns	ps	ZE	ZE	ZE	ZE	ZE	ZE	ZE	ZE	ZE	ZE	ZE	ZE	ZE	ZE	ZE
ns	ze	ZE	ZE	ZE	PM	PM	PM	PM	ZE	ZE	ZE	ZE	ZE	ZE	ZE	ZE
ns	ns	PM	ZE	ZE	ZE	ZE	ZE	PM	PM	PM	PM	PM	NM	ZE	ZE	ZE

Fig. 5.1 - Exemplo de tabela de regras para a variável *cíclico lateral*

Uma regra do tipo *IF erro IS ps AND derro IS ze AND roll IS ps AND droll IS ze THEN saída IS ze ENDIF* pode ser vista, também, como uma estrutura que possui antecedentes e conseqüente da seguinte maneira: *IF antecedentes THEN conseqüente ENDIF*.

A seguir são descritos a especificação e o funcionamento do gerador de tabelas e do tradutor de tabelas de regras lingüísticas.

A) GERADOR DE TABELAS DE REGRAS LINGÜÍSTICAS

Com o objetivo de padronizar a estrutura de uma tabela de regras e, também, de facilitar a montagem da mesma, foi especificado um programa gerador de tabelas de regras. Este programa recebe como entradas todos os dados fornecidos pelo usuário a respeito da tabela que ele deseja construir e tem como saída um arquivo texto com o protótipo de uma tabela vazia. O preenchimento desta tabela pode ser feito através de um programa editor de texto com a informação obtida através de métodos de aquisição de conhecimento. A tabela pode ser total ou parcialmente preenchida, representando, no segundo caso, situações indefinidas para a variável de saída.

A especificação do programa gerador de tabelas foi feita considerando a estrutura pretendida para uma tabela. A linguagem escolhida para a sua implementação foi a linguagem "C". As características principais exigidas para a tabela são:

- o número mínimo de variáveis deve ser 2 (dois) e o seu número máximo deve ser 6 (seis), uma vez que para um número maior de variáveis estaríamos perdendo algumas das características favoráveis da representação de regras lingüísticas por tabelas, e também, porque dentro da aplicação a que se pretende, o número de variáveis encontra-se dentro do intervalo citado;

- no caso de um número ímpar de variáveis, o maior número delas deve ficar nas linhas para evitar que a tabela tenha uma largura maior que a altura, dificultando sua visualização;
- o número máximo de conjuntos nebulosos por variável deve ser 7 (sete) e, também, esse número deve ser sempre ímpar (esta característica deriva diretamente da forma de organização simétrica dos conjuntos nebulosos dentro do universo de discurso);
- as variáveis a serem representadas diretamente na tabela devem possuir um identificador com 2 caracteres para facilitar sua visualização na tabela;
- o programa gerador de tabelas deve ter como entradas as informações do usuário sobre as variáveis de entrada e saída, seus respectivos conjuntos nebulosos (número de conjuntos e os pontos que definem cada conjunto no universo de discurso) e os identificadores de dois caracteres que irão representar cada uma das variáveis na tabela;
- a tabela de regras nebulosas deverá ser gerada em arquivo texto para facilitar seu preenchimento o que, conseqüentemente, poderá ser feito através de um processador de texto.

B) TRADUTOR DE TABELAS DE REGRAS LINGÜÍSTICAS

O Tradutor de tabelas é um programa que toma como entrada um arquivo texto contendo uma tabela de regras nebulosas e gera arquivos contendo parte do código para a implementação da máquina de inferência, além de um arquivo contendo o conjunto com todas as regras definidas na tabela.

Inicialmente havia em funcionamento uma versão do tradutor de tabelas que funcionava para uma tabela de tamanho fixo. A tabela era montada e preenchida em um editor de texto sendo que todos os pontos da tabela deveriam ter um valor definido. O tradutor funcionava para uma tabela com 4 variáveis de entrada, sendo que uma variável possuía 5 conjuntos e as 3 variáveis restantes possuíam 3 conjuntos nebulosos cada (tabela de 135 pontos).

A proposta deste trabalho foi a elaboração de um tradutor para tabelas de tamanho variável, que gerasse um conjunto de regras nebulosas em arquivo, além de parte do código necessário à implementação do controlador nebuloso. Este tradutor, a exemplo do gerador de tabelas, foi implementado em linguagem "C". O tradutor de tabelas foi especificado com as seguintes características:

- receber como entrada um arquivo padronizado contendo uma tabela de regras;

- gerar na saída arquivos contendo o conjunto de regras correspondentes aos pontos da tabela e parte do código para a implementação da máquina de inferência;

- ter capacidade de tradução para tabelas de qualquer tamanho, respeitando-se o número máximo de variáveis especificado para a tabela de regras (no caso, seis variáveis) e o número máximo de conjuntos por variável (no caso, sete conjuntos);

- considerar a possibilidade de haver pontos sem definição dentro da tabela, resultantes da não consideração daquela situação durante a fase de captura do conhecimento ou da não ocorrência daquela situação durante a execução de uma tarefa. Como resultado, o conjunto de regras que o gerador deve criar deverá ser formado apenas pelas regras que possuem uma saída definida.

A idéia básica de ambos os programas é vista na figura 5.2, onde *Tabela* e *Regras + Código Forth* correspondem aos arquivos criados no primeiro e segundo estágios do processo de geração/tradução da tabela de regras lingüísticas.

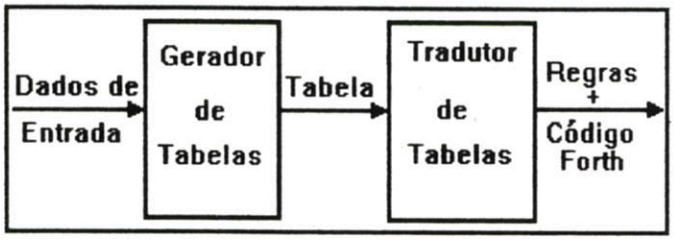


Fig. 5.2 A estrutura funcional do gerador/tradutor de tabelas

5.3.2 A Implementação da Máquina de Inferência

O protótipo da máquina de inferência foi desenvolvido em forth considerando três etapas básicas do processo de inferência:

- fuzificação;

- avaliação das regras;

- defuzificação.

A proposta inicial foi a criação de um ou mais módulos que executassem as três etapas da maneira mais eficiente possível, sem perder de vista a legibilidade e estruturação do código, o que iria favorecer a sua posterior análise e reutilização.

O resultado foi a criação de sete módulos, sendo que alguns destes foram estruturados de modo a permitir a sua geração automática através do tradutor de tabelas de regras nebulosas.

Como visto no capítulo anterior, a máquina de inferência foi projetada de modo a utilizar o mínimo como função de implicação e o máximo como função de composição. A estratégia de defuzificação utilizada foi a de *Centro de Área* (COA - Center of Area) que permite obter melhores resultados como visto no capítulo 2.

Uma versão posterior da máquina de inferência considerou também a estratégia de avaliação de regras proposta por Sugeno que consiste, basicamente, na definição de uma função de saída para cada regra. Esta função toma como argumentos os valores determinísticos (não nebulosos) das variáveis de entrada, gerando um resultado que é armazenado em um vetor. A estratégia de defuzificação seria, então, a aplicação de uma média ponderada à saida de cada uma das funções, cujo peso seria dado pelo grau de pertinência resultante da avaliação da respectiva regra.

A seguir são descritos de maneira suscinta os módulos implementados.

A) SUPORTE.4TH

Neste módulo estão definidas palavras de suporte no dicionário forth que auxiliam na definição dos módulos restantes. As palavras aqui definidas são palavras que criam funções para operações especiais com pilhas, para a criação de vetores e matrizes, geração de números randômicos, etc.

Este módulo deve ser o primeiro módulo de um projeto a ser carregado na memória, uma vez que as suas palavras são definidas a partir de palavras já existentes no dicionário forth e, também, porque todos os módulos restantes dependem das definições nele contidas.

B) FUZZIFIC.4TH

É o módulo onde estão definidas as palavras que implementam a fuzificação de uma variável.

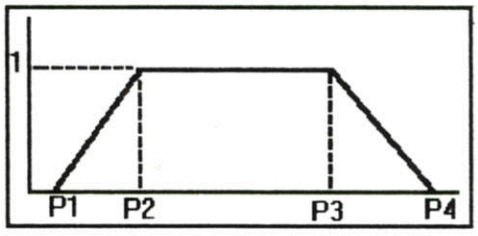


Fig. 5.3 Identificação dos quatro pontos da função de pertinência p1, p2, p3 e p4

A palavra MEMB, definida neste módulo, recebe como argumentos o valor da variável a ser avaliada e os quatro pontos que definem a função de pertinência associada a um conjunto

nebuloso dentro do universo de discurso (figura 5.3). Como resultado esta palavra (ou função) armazena o valor do grau de pertinência da variável ao conjunto nebuloso na pilha.

C) MÓDULO DE DEFINIÇÃO DOS CONJUNTOS NEBULOSOS

Este módulo é um dos módulos gerados automaticamente pelo tradutor de tabelas e, sendo assim, o seu nome é definido pelo usuário.

Neste módulo são declaradas as variáveis de entrada do sistema como sendo constantes para o código forth. Isso significa que a cada uma das variáveis de entrada é associada uma constante inteira que permitirá sua identificação pelo restante do sistema.

São definidos também o número de variáveis de entrada do sistema e o nome da variável de saída que é declarada como uma palavra vazia (possui função apenas mnemônica ou de documentação quando declarada no conseqüente das regras).

Neste módulo são definidas ainda funções que permitem determinar o grau de pertinência de uma determinada variável a qualquer um dos conjuntos nebulosos associados a ela. Estas funções tomam como argumentos o valor da variável e um índice que permite identificar em relação a qual conjunto deseja-se fazer a avaliação. Isso é possível a partir da definição das funções de pertinência de cada conjunto nebuloso, que pertencem a cada uma das variáveis nebulosas. A definição destas funções de pertinência corresponde à identificação dos quatro pontos que definem a função de pertinência do conjunto no universo de discurso (pontos p1, p2, p3 e p4 na figura 5.3). Estas funções são definidas uma para cada variável de entrada e seus endereços são armazenados em um vetor (VetorCi). A maneira de executar estas funções seria através do acesso às posições deste vetor (execução vetorada).

Neste módulo são definidas também palavras que têm a função de armazenar os quatro pontos de cada um dos conjuntos nebulosos da variável de saída na pilha. Desta forma é necessário definir uma palavra para cada conjunto desta variável.

Estas palavras serão necessárias para o cálculo do centróide e da área de cada conjunto nebuloso (onde são utilizados os quatro pontos da função de pertinência e o grau de pertinência). Ambas as informações são necessárias para a estratégia de defuzificação pelo centro de área.

D) REGRA.4TH

Neste módulo são declaradas as constantes que representam os conjuntos nebulosos, a MatrizGrau que armazena os graus de pertinência de todas as variáveis a cada um dos seus conjuntos e a variável Grau que armazena os valores de grau de pertinência durante a avaliação de uma regra.

É também definida neste módulo a palavra AvaliaVariável que, tomando como argumentos o valor de uma variável e a constante que a representa, calcula o grau de pertinência desta variável a todos os seus conjuntos nebulosos associados. O resultado deste cálculo é armazenado na MatrizGrau.

Todas as palavras e conectivos que permitem estruturar uma regra linguística também estão definidos neste módulo (If, Is, And, Or, Then, EndIf).

E) DEFUZIFIC.4TH

Neste módulo estão definidas as palavras que permitem implementar a estratégia de defuzificação segundo o método do Centro de Área.

A palavra *Saída* executa todo o cálculo e geração do valor resultante da máquina de inferência.

F) MÓDULO DA BASE DE REGRAS

O módulo da base de regras é também um dos módulos gerados automaticamente a partir da tradução das tabelas de regras. Ele contém a definição da palavra *Base* que possui a declaração de todo o conjunto de regras utilizado pela máquina de inferência e que é resultante da tradução da tabela propriamente dita. A execução da palavra Base, desta forma, permite a avaliação de todo um conjunto de regras.

G) MÓDULO DE EXECUÇÃO

Este módulo define o processo que irá executar a máquina de inferência e os canais de comunicação de entrada e saída. Este módulo também é um dos módulos gerados automaticamente a partir da tradução de tabelas.

5.3.3 O PROCESSO MÁQUINA DE INFERÊNCIA

O processo que implementa a máquina de inferência possui uma estrutura funcional dividida em três elementos:

- entrada de estados e fuzificação;

- avaliação das regras;

- defuzificação e saída de controle.

Um exemplo da estrutura desta máquina de inferência em código forth é apresentado na figura 5.4.

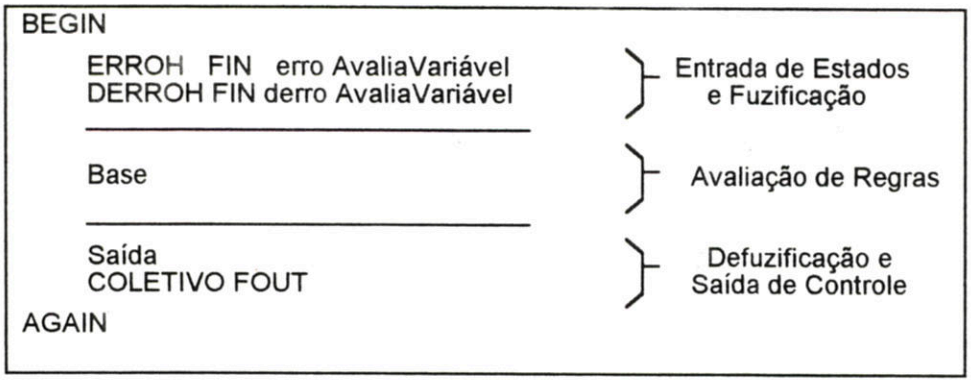


Fig. 5.4 Implementação do Processo Máquina de Inferência em Forth

Vemos através da figura 5.4 que os elementos que compõem a máquina de inferência estão colocados dentro de um *looping* eterno (BEGIN...AGAIN). Essa é uma das exigências para a estruturação de uma processo em Forth.

A seguir são descritos cada um dos elementos componentes do processo máquina de inferência.

A) ENTRADA DE DADOS E FUZIFICAÇÃO

A entrada de dados é feita através da leitura do valor de um estado (ou de seu erro) em um canal de entrada. A sintaxe da instrução de entrada seria, como por exemplo

ERROH FIN

onde ERROH é o canal definido para a transmissão do valor da variável *erro*. Esta instrução armazena o valor que chega pelo canal na pilha de ponto flutuante.

A fuzificação é executada pela palavra *AvaliaVariável* onde os valores do grau de pertinência de cada variável a cada um de seus conjuntos são armazenados na *MatrizGrau*. A palavra *AvaliaVariável* toma como argumentos um valor constante correspondente à identificação da variável e o valor atual desta variável obtido do helicóptero.

A *MatrizGrau* possui uma estrutura semelhante à estrutura representada na figura 5.5.

Como um exemplo de interpretação da matriz da figura 5.5 podemos citar que **G21** representa o grau de pertinência da variável 2 ao conjunto 1.

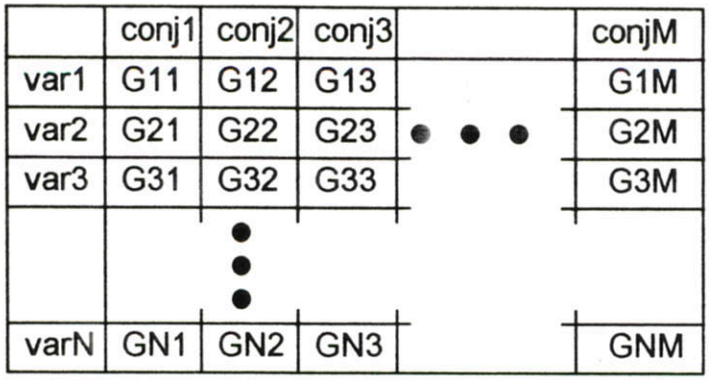


Fig. 5.5 Representação da MatrizGrau

Sendo as variáveis e conjuntos nebulosos definidos em forth identificados como constantes numéricas, a declaração de uma variável e de um conjunto nebuloso fornecem índices que permitem uma pesquisa na *MatrizGrau*.

B) AVALIAÇÃO DAS REGRAS

Uma regra lingüística, segundo a definição em forth, possui a seguinte sintaxe:

If var1 Is conj1 And var2 Is conj2 ... Then varSaída Is conjS Endif.

A regra como acima definida funciona com o auxílio de três variáveis básicas:

- *MatrizGrau* que, como já visto, é uma matriz que armazena os graus de pertinência de todas as variáveis de entrada a todos os seus respectivos conjuntos;

- *Grau*, que é uma variável que armazena o grau de pertinência mínimo resultante da avaliação de uma regra, e que corresponde à aplicação da função de implicação *mínimo*;

- *ResultGrau*, que é um vetor que armazena os graus de pertinência resultantes da avaliação de uma base de regras. O vetor *ResultGrau* armazena os graus de pertinência de regras com o mesmo conseqüente em uma determinada posição, segundo o critério de máximo, o que corresponde à aplicação da função de composição *máximo*. Cada posição deste vetor corresponde a um conjunto nebuloso da variável de saída. Como os conjuntos nebulosos são associados a constantes numéricas no código forth, a referência a um desses conjuntos fornece um índice para a busca e armazenamento no vetor ResultGrau.

Partindo da consideração das três variáveis principais utilizadas na avaliação de uma regra, podemos agora descrever como ocorre a avaliação de uma dessas regras no sistema proposto. Assim, o funcionamento de uma regra se resume a:

- inicializar a variável Grau com o valor 1 (função da palavra If);

- buscar um valor na MatrizGrau para cada antecedente da regra, comparar este valor com o valor contido na variável Grau armazenando-o nesta variável se ele for menor do que aquele valor que a variável possuía antes (implementação da função de implicação *mínimo*). Os índices para a pesquisa na *MatrizGrau* são fornecidos naturalmente pela declaração de uma variável e de um conjunto nebuloso em qualquer dos antecedentes das regras. Esta função é implementada pelas palavras And e Then;

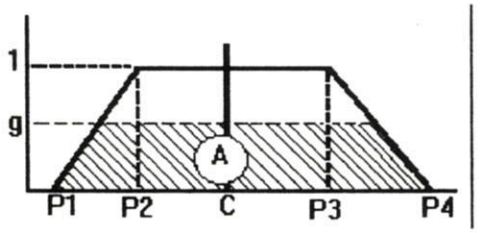
- buscar o valor armazenado na variável Grau comparando-o com o valor presente numa das posições do vetor ResultGrau. Se este valor for maior que o valor contido no vetor ResultGrau, ele será armazenado no vetor (implementação da função de composição *máximo*). O índice desta posição do vetor é fornecido pelo conjunto nebuloso do conseqüente da regra.

C) DEFUZIFICAÇÃO

O resultado da avaliação de um conjunto de regras são todos os graus de pertinência obtidos através da aplicação da função de implicação e da função de composição, e que são armazenados no vetor ResultGrau.

Estes valores de graus de pertinência são necessários para que possamos calcular a área de cada conjunto nebuloso da variável de saída.

A defuzificação, segundo o critério do Centro de Área, consiste numa média ponderada dos centróides de todos os conjuntos da variável de saída. O peso atribuído a cada centróide é dado pela área do conjunto nebuloso correspondente, considerando o seu grau de pertinência resultante. Na figura 5.6 pode ser vista a representação da função de pertinência de um conjunto nebuloso com o centróide e a área do conjunto, considerando-se o grau de pertinência.



g : grau de pertinência; C: centróide do conjunto nebuloso;

A: área do conjunto limitada por g

Fig 5.6 Representação do Centróide e da Área do Conjunto Nebuloso

Tanto a área como o centróide de cada conjunto são armazenados em vetores (VetorÁrea e VCentro, respectivamente) cujas posições correspondem a cada um dos conjuntos nebulosos de saída. O acesso a estes vetores é realizado pela palavra Saída para o cálculo da saída da máquina de inferência.

A saída da máquina de inferência é calculada por:

$$\text{saída} = \frac{\sum Ai * Ci}{\sum Ai}$$

onde Ai e Ci são os valores de área e centróide lidos diretamente dos vetores VetorÁrea e VCentro.

5.4 A Estruturação do Controlador Nebuloso e da Missão

A estrutura criada para permitir o funcionamento de todo o controlador nebuloso foi construída utilizando os conceitos de processos e de multiprogramação. Cada máquina de inferência foi implementada como um processo independente executando em um mesmo processador, cuja comunicação se efetiva através de canais síncronos.

As informações que chegam do helicóptero e são transmitidas para cada um dos processos que implementam as máquinas de inferência, bem como as informações originárias destes processos e que devem ser enviados ao helicóptero são gerenciadas por um processo *gerente*. O processo *gerente* é o processo que efetivamente calcula o valor das variáveis de controle e que possui também a função de coordenar a missão.

Um esquema simplificado da estrutura apresentada pelo controlador pode ser visto na figura 5.7.

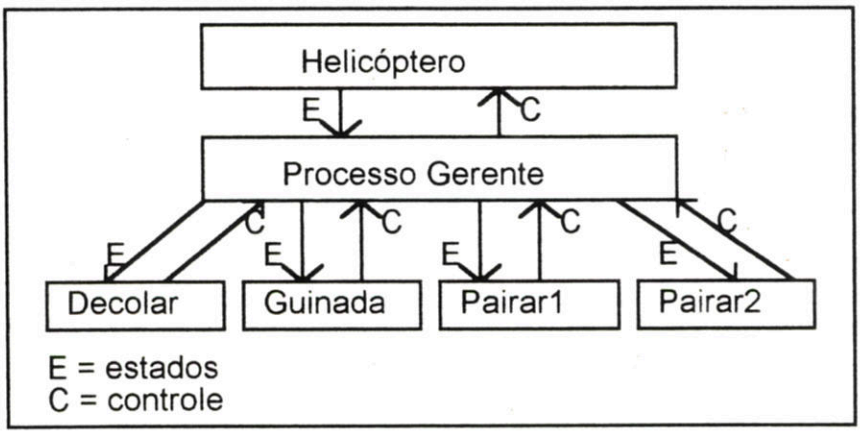


Fig. 5.7 Estrutura do Sistema para Aplicação do Controlador Nebuloso

A missão definida para verificar o desempenho do controlador nebuloso está implementada a nível de código forth no processo *gerente*. Qualquer modificação na missão refletirá na mudança estrutural do processo *gerente*. A missão é composta das seguintes tarefas:

- decolar até uma altura de 4,0 metros;
- girar para a esquerda de até atingir um ângulo de 0.5 radianos;
- pairar sobre uma posição que se encontra distante da posição de decolagem 3,0 metros no eixo **x** e 3,0 metros no eixo **y**.

A descrição de cada um dos processos que permitem implementar esta missão é feita a seguir.

A) DECOLAR

A tarefa *Decolar*, implementada pelo processo *decolar*, tem por objetivo o movimento vertical ascendente do helicóptero até atingir uma altura de referência.

A máquina de inferência que implementa esta tarefa utiliza como variáveis de entrada o erro de altitude e sua variação e possui como saída um valor que irá compor o comando coletivo do rotor principal.

O comando coletivo do rotor principal implementa um controle integral e, desta forma, o valor fornecido pela máquina de inferência vai sendo acrescentado ao controle em cada ciclo.

B) GUINADA (YAW)

A tarefa *Guinada*, implementada pelo processo *guinada*, tem por objetivo o direcionamento do nariz do helicóptero num plano horizontal, em relação a um referencial fixo na terra.

A máquina de inferência que implementa esta tarefa utiliza como variáveis de entrada o erro do ângulo de guinada e sua variação e, como saída, um valor que irá compor o comando coletivo do rotor de cauda.

A exemplo do que ocorre no comando coletivo do rotor principal, o comando coletivo do rotor de cauda também implementa um controle integral e, portanto, o valor fornecido pela máquina de inferência vai sendo acrescentado ao controle em cada ciclo.

C) PAIRAR

A tarefa pairar é implementada por processos independentes como duas tarefas independentes: *Pairar1* (pelo processo *pairar1*) e *Pairar2* (pelo processo *pairar2*).

A tarefa Pairar1 tem por objetivo posicionar o helicóptero em uma coordenada de referência no eixo **y** (posicionamento lateral), em relação a um referencial fixo na terra.

A máquina de inferência que implementa esta tarefa utiliza como variáveis de entrada o erro de posicionamento lateral e sua variação e o ângulo de rolagem (roll) e sua variação. Como variável de saída esta máquina fornece o comando de cíclico lateral do rotor principal que é diretamente aplicado ao helicóptero.

A tarefa Pairar2 tem por objetivo posicionar o helicóptero em uma posição de referência no eixo **x** (posicionamento longitudinal), em relação a um referencial fixo na terra.

A máquina de inferência que implementa esta tarefa utiliza como variáveis de entrada o erro de posicionamento longitudinal e sua variação e o ângulo de arfagem (pitch) e sua variação. Como variável de saída esta máquina fornece o comando de cíclico longitudinal do rotor principal que é diretamente aplicado ao helicóptero.

D) GERENTE

No processo *gerente* é onde está definida a missão. Este processo coordena a leitura de estados do helicóptero, que corresponde à leitura dos seus sensores, tratamento da informação de entrada, envio de valores às máquinas de inferência, leitura das saídas de cada uma das máquinas, composição dos controles e envio destes controles ao helicóptero.

O processo *gerente* implementa também critérios de estabilidade que irão determinar a ativação da próxima tarefa dentro de uma missão, como visto no capítulo 3. Assim que o helicóptero atinge a estabilidade em uma determinada tarefa, o processo gerente poderá ativar o estágio seguinte da missão.

A ativação do controlador ocorre com a ativação de todos os processos envolvidos: decolar, guinada, pairar1, pairar2 e gerente.

Como os canais são síncronos, os processos que não recebem informação ficam em estado de espera, sem consumir tempo do processador. Este é o caso das tarefas guinada, pairar1 e pairar2 no início da missão, que corresponde ao início do processamento do controlador nebuloso.

A seqüência de funcionamento do controlador nebuloso para a missão definida, acontece da seguinte maneira:

1. Todos os processos são ativados sendo que somente o processo gerente e o processo decolar permanecem ativos (todos os outros processos entram em estado de espera).

2. À medida que o helicóptero, que está decolando, atinge uma altura de 4 metros e encontra-se estável nessa altura, a missão que estava no estado 0 passa para o estado 1, o que significa a ativação de mais um processo: o da tarefa *guinada*.

3. Assim que o helicóptero atinge a estabilidade na tarefa guinada, a missão passa para o estado 3 onde são ativados também os processos das tarefas pairar1 e pairar2. Assim o helicóptero deve se deslocar para uma posição de referência e permanecer pairando sobre ela durante um tempo indeterminado. Desta forma a missão está concluída.

5.5 Conclusão

Neste capítulo foram apresentados os aspectos construtivos e funcionais do controlador nebuloso para uma missão proposta.

Todo o desenvolvimento dos trabalhos ocorreu nas dependências do LCMI (Laboratório de Controle e Microinformática - UFSC) com base na infraestrutura em Forth para transputer instalada neste laboratório.

Toda a implementação foi caracterizada por um aprendizado intenso que atingiu desde a familiarização com as linguagens utilizadas ("C" e Forth), até a consideração da estrutura necessária para a implementação de um controlador nebuloso.

No próximo capítulo é descrita a fase de simulação onde são também discutidos problemas e soluções encontradas, e os resultados obtidos.

CAPÍTULO 6

SIMULAÇÃO

No capítulo anterior foram descritas a implementação da máquina de inferência e do controlador nebuloso considerando a missão proposta.

Neste capítulo é descrita a simulação que permite verificar o desempenho do sistema desenvolvido. Os resultados obtidos a partir da simulação são apresentados e discutidos. São apontadas limitações do sistema e discutidas soluções para melhoria do seu desempenho.

6.1 INTRODUÇÃO

A tarefa de simulação visa verificar o funcionamento de um controlador nebuloso em Forth para transputer, em especial no que diz respeito ao desempenho e confiabilidade do sistema desenvolvido sobre a esta plataforma.

O sistema foi testado através da interação com o simulador HSIM, desenvolvido pela Gyron Tecnologia. Como este simulador roda no sistema operacional MS-DOS, em IBM-PC, foi necessário estruturar um canal de comunicação para tornar possível a integração do controlador ao simulador.

A infraestrutura em Forth para transputer instalada no LCMI é composta de placa módulo para transputer com um processador instalado, interpretador Forth e console de desenvolvimento (ambos, interpretador e console, desenvolvidos pela Gyron Tecnologia).

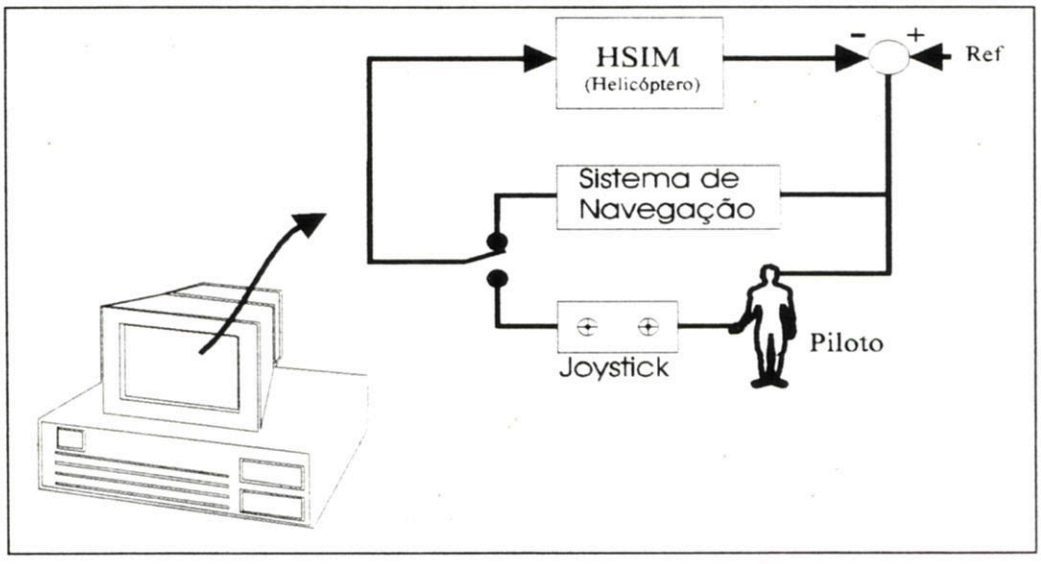


Fig. 6.1 - Interação do Sistema de Navegação com o simulador HSIM

A console de desenvolvimento foi desenvolvida de maneira a estar em comunicação direta com o transputer, ou seja, ela só pode ser rodada se a placa de transputer estiver instalada no computador. Desta forma a maneira encontrada para efetivar a comunicação entre o controlador, que roda no transputer, e o simulador, que roda no IBM-PC, foi através da console de desenvolvimento. Todos os estados do helicóptero são enviados ao transputer como palavras definidas dentro da console. Da mesma forma, os controles são lidos pelo simulador através da leitura dos resultados que chegam até à console.

O sistema de navegação interage com o simulador HSIM substituindo o "joystick", enviando ao simulador os valores dos comandos equivalentes aos que seriam gerados através do "joystick", conforme se vê na figura 6.1.

6.2 SIMULAÇÃO

Neste trabalho foram implementadas as tarefas Decolar, Guinada e Pairar. Para análise do desempenho do sistema, foi realizada a simulação da seguinte missão:

- decolar até a altitude de 4 metros;
- fixar o ângulo de guinada em 0.5 radianos;
- permanecer pairado em uma posição distante 3.0 m em **x** e 3.0 m em **y** da posição de decolagem.

Toda a fase de aquisição de conhecimento, definição dos mecanismos de inferência e ajuste de regras foi realizada em [Cav 94], como já mencionado. Sendo assim, o objetivo de simulação neste caso não é o ajuste do controlador. Outrossim, o objetivo de simulação aqui proposto é a verificação do funcionamento do algoritmo de controle implementado numa plataforma em linguagem Forth para transputer, análise do desempenho de tal sistema de controle em termos de tempo de processamento, além de uma solicitação do sistema a fim de testar a sua robustez e confiabilidade.

As conclusões a respeito do desempenho do algoritmo de controle foram tiradas da comparação entre os resultados obtidos em [Cav 94] e os resultados obtidos no presente trabalho, ambos a nível de simulação. Assim [Cav 94] será o nosso sistema padrão, cujos resultados a nível de simulação foram considerados satisfatórios para a aplicação a que se destina: o helicóptero da Gyron Tecnologia.

Inicialmente faz-se uma comparação dos gráficos temporais para as quatro máquinas. Estes gráficos são traçados em função do número de iterações do sistema, sendo que, desta forma, permitem apenas verificar a correção do algoritmo de controle.

Em seguida é descrita uma análise do desempenho em termos de tempo de processamento do controlador implementado, onde são apresentados os resultados iniciais desta análise e possíveis soluções para a melhoria de desempenho.

6.2.1 Análise da Execução das Tarefas

Nesta seção é descrita a análise da execução da missão através da análise de cada tarefa independentemente.

Esta análise foi realizada através da comparação de gráficos da resposta temporal do sistema padrão e do sistema proposto neste trabalho. Porém, como já mencionado, o gráfico da resposta temporal possui como abcissa não o tempo decorrido mas o número de iterações do sistema. Sendo assim, tais gráficos não demonstram aspectos de tempo de resposta do sistema, mas permitem verificar informações importantes em relação ao desempenho dinâmico e estático do sistema como sobrepasso e erro em regime permanente.

A seguir são apresentados os resultados obtidos para cada uma das tarefas componentes da missão.

a) Decolar

A tarefa decolar é a primeira tarefa a ser executada dentro da missão. Ela consiste na decolagem do helicóptero até uma altura de 4,0 metros, que como definida na missão, corresponde à altura de referência.

O gráfico da saída do sistema (no caso a altura) versus o número de iterações (figura 6.2) demonstrou um funcionamento do sistema semelhante ao funcionamento obtido no sistema padrão. Pode-se observar entretanto que o sistema padrão atinge o regime permanente num menor número de iterações e que o erro em regime permanente para este sistema (padrão) não é superior a 10 cm. No caso do sistema em teste o erro em regime permanente não é superior a 5 cm.

O sobrepasso do sistema em teste é também menor que o sobrepasso apresentado pelo sistema padrão.

As divergências em pequena escala observadas entre os dois sistemas foram observadas já a nível de fuzificação. Os graus de pertinência considerados no sistema padrão para um mesmo valor de entrada divergem ligeiramente dos obtidos do sistema em teste. Isto se deve, basicamente, aos erros de truncamento, uma vez que o sistema padrão utiliza uma aritmética inteira no cálculo dos graus de pertinência, enquanto que o sistema em teste utiliza aritmética em ponto flutuante.

Sabendo que o cálculo do comando coletivo depende diretamente do grau de pertinência das variáveis de entrada aos seus conjuntos nebulosos, está justificada a razão da divergência entre as variáveis de controle consideradas e, conseqüentemente, a razão da divergência entre os gráficos apresentados.

Nas figuras 6.3 e 6.4 são apresentados os gráficos da velocidade vertical e do comando coletivo para ambos os sistemas.

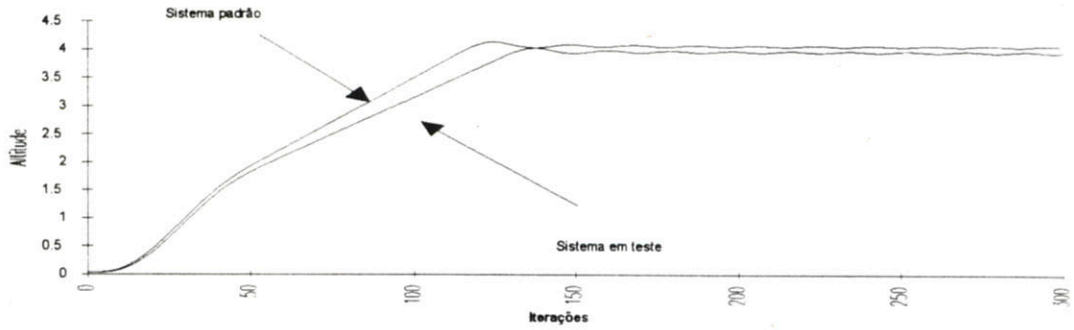


Fig. 6.2 Gráficos de Altitude **X** número de iterações

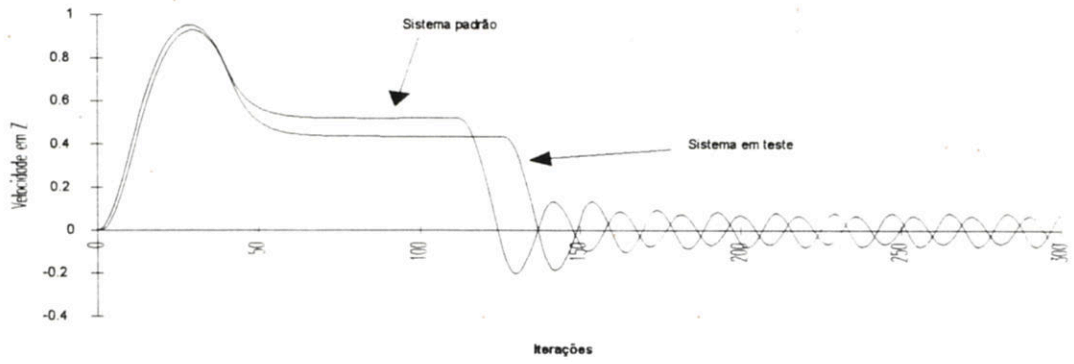


Fig. 6.3 Gráficos de Velocidade no eixo *Z* **X** número de iterações

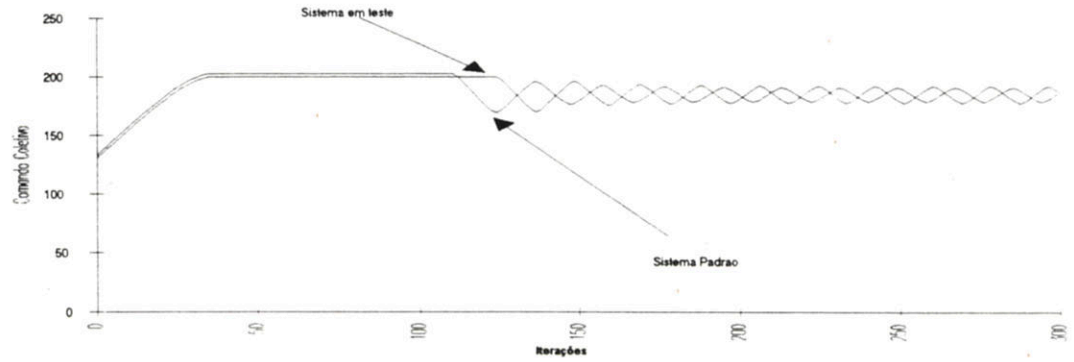


Fig. 6.4 Graficos de Comando Coletivo **X** número de iterações

B) GUINADA

Assim que o helicóptero encontra-se estável ao final da tarefa decolar, o processo que executa a tarefa guinada passa ao estado *ativo*.

Sendo assim o helicóptero tende a se orientar numa posição de referência girando em torno do eixo Z.

O gráfico que demonstra a evolução do ângulo de guinada em função do número de iterações pode ser visto na figura 6.5. Este gráfico demonstra que tanto o sistema padrão quanto o sistema em teste atingem o regime permanente no mesmo número de iterações. É possível observar também que o sobrepasso proporcionado pelo sistema em teste é menor.

Nas figuras 6.6 e 6.7 são apresentados os gráficos de velocidade de guinada e do comando de cauda em função do número de iterações.

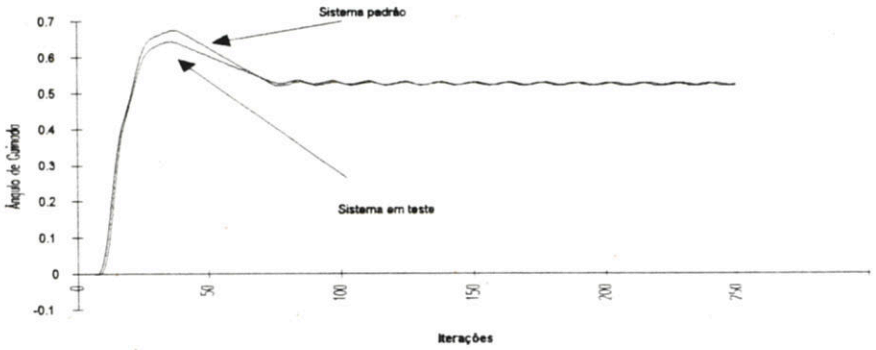


Fig. 6.5 Gráficos do Ângulo de Guinada **X** número de iterações

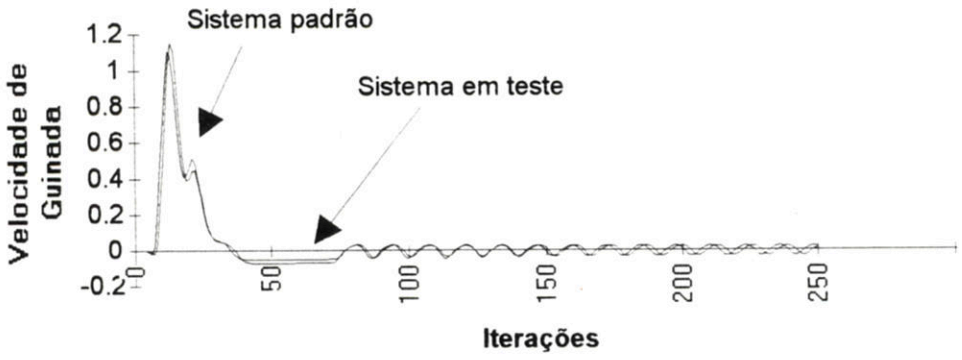


Fig. 6.6 Gráficos da Velocidade de Guinada X número de iterações

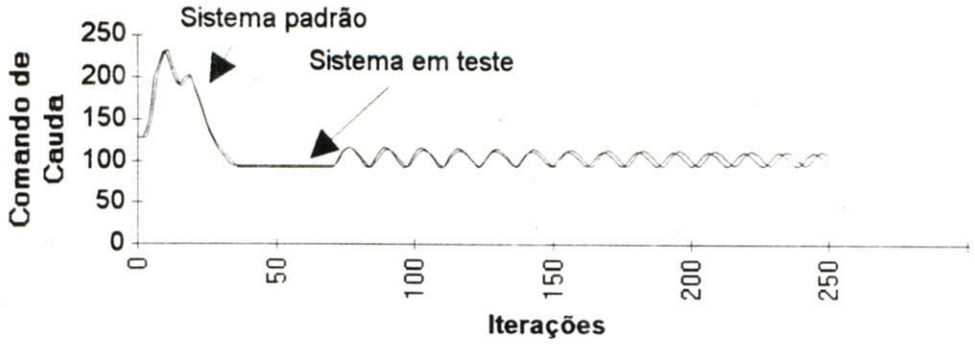


Fig. 6.7 Gráficos do Comando de cauda X número de iterações

c) PAIRAR

O último processo a ser ativado na missão é o processo *Pairar*. A ativação deste processo ocorre quando a tarefa *Guinada* atinge a estabilidade.

A posição de referência sobre a qual o helicóptero deve pairar encontra-se a 3.0 metros em **X** e a 3.0 metros em **Y** da posição de decolagem (a posição inicial é X=8,0 e Y=2,0 e a posição final é X=5,0 e Y=-1,0).

Como a tarefa pairar é composta de dois processos independentes que controlam a posição em X e a posição em Y do helicóptero, o comportamento em cada um dos eixos pode ser analisado separadamente.

Vemos nas figuras 6.8, 6.9 e 6.10 os gráficos da posiçao do helicóptero no eixo X, velocidade em X e do comando cíclico longitudinal em função do número de iterações.

Pela análise dos gráficos podemos verificar um comportamento próximo entre os dois sistemas considerados.

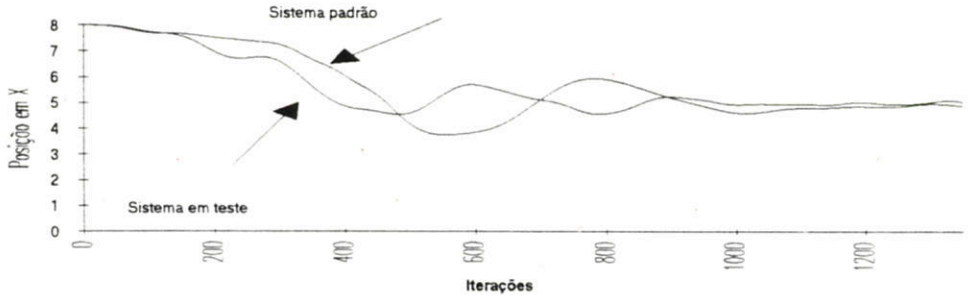


Fig. 6.8 Gráfico da Posição em *X* **X** número de iterações

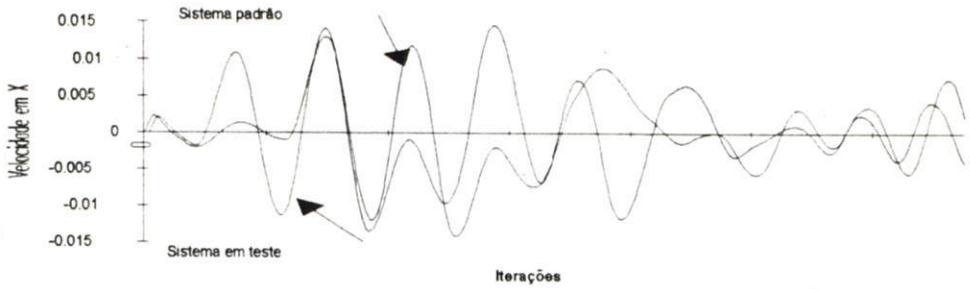


Fig. 6.9 Gráfico da Velocidade em *X* **X** número de iterações

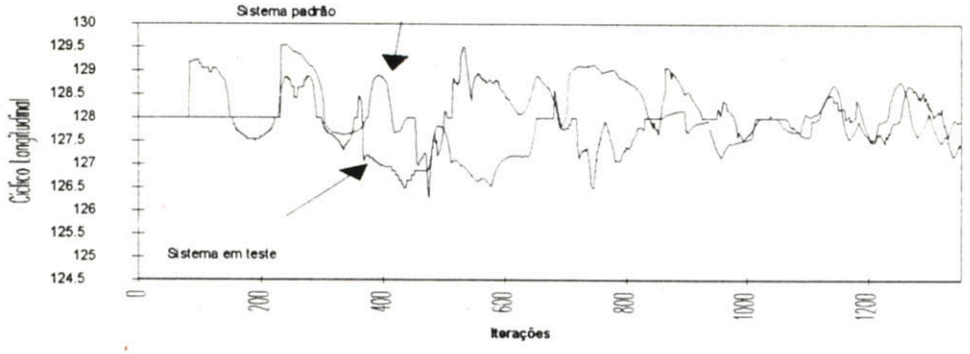


Fig. 6.10 Gráfico do Comando Cíclico Longitudinal **X** número de iterações

Os gráficos 6.11, 6.12 e 6.13 apresentam, respectivamente, a evolução da posição do helicóptero em Y, da sua velocidade de aproximação em Y e do comando cíclico lateral.

Identifica-se nesta análise uma divergência significativa entre os sistemas considerados, especialmente no que diz respeito ao tempo que o sistema leva para atingir a situação de estabilidade na posição de referência em Y.

A ação de controle no sistema em teste é mais intensa, resultando numa maior velocidade de aproximação e na maior dificuldade em atingir a estabilidade.

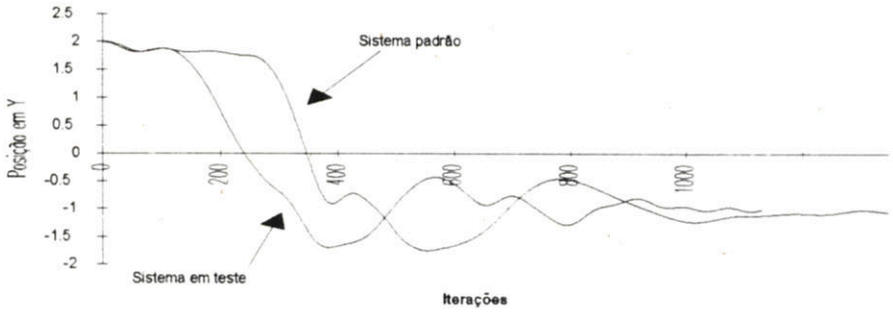


Fig. 6.11 Gráfico da Posição em *Y* **X** número de iterações

Fig. 6.12 Gráfico da velocidade em *Y* **X** número de iterações

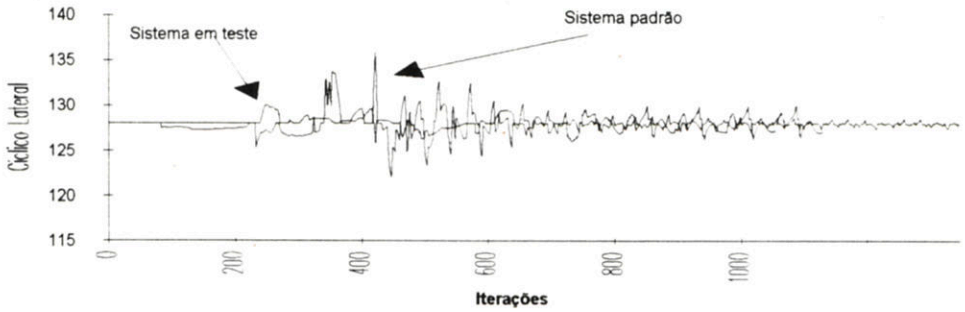


Fig. 6.13 Gráfico do Comando Cíclico Lateral **X** número de iterações

6.2.2 Análise do Desempenho Computacional

A análise do desempenho computacional foi efetuada comparando-se o tempo de uma iteração entre o sistema em teste e o sistema padrão. O procedimento foi a execução do sistema *simulador/sistema de navegação* um número de vezes pré-estabelecido onde foram comparados os tempos obtidos.

O resultado do teste demonstrou que o sistema implementado em [Cav 94] apresenta um desempenho, em termos do tempo de execução, da ordem de 70 ms, enquanto que o sistema implementado neste trabalho apresentou um tempo de execução de, aproximadamente, 260 ms.

Procedeu-se então uma análise sistemática para buscar a causa do desempenho inferior proporcionado por esta implementação. Esta análise sistemática se deu sobre a máquina de inferência da tarefa *pairar1* sendo que os resultados foram atribuídos também às outras máquinas.

Para esta máquina o tempo de uma iteração encontrava-se da ordem de 90 ms divididos da seguinte maneira:

- entrada de dados e fuzificação: 15 ms;
- avaliação das regras: 65 ms

- defuzificação: 10 ms.

Estava então determinado que a operação mais cara em termos de consumo de tempo estava sendo a avaliação das regras.

A partir deste ponto foram consideradas algumas propostas de solução para a melhoria de desempenho. Tais propostas estão descritas a seguir.

6.3 SOLUÇÕES PARA O DESEMPENHO

A base de regras como definida a princípio, exigia a avaliação de todas as regras, ativas ou inativas, em cada iteração. Esta situação determinava a análise de 368 regras lingüísticas a cada iteração do sistema.

Analisando-se o problema sobre a máquina de inferência *pairar1* foi verificado que um determinado antecedente ocorria em um determinado número de regras, sendo que sua execução era repetida. Foi proposta então uma estrutura em forma de árvore que, para sistemas com um número considerável de antecedentes, proporciona uma ganho razoável em termos do tempo de execução. Um exemplo entre a configuração original da base de regras e da configuração atual, considerando a estrutura em árvore, para um sistema com duas variáveis de entrada, pode ser visto na figura 6.14.

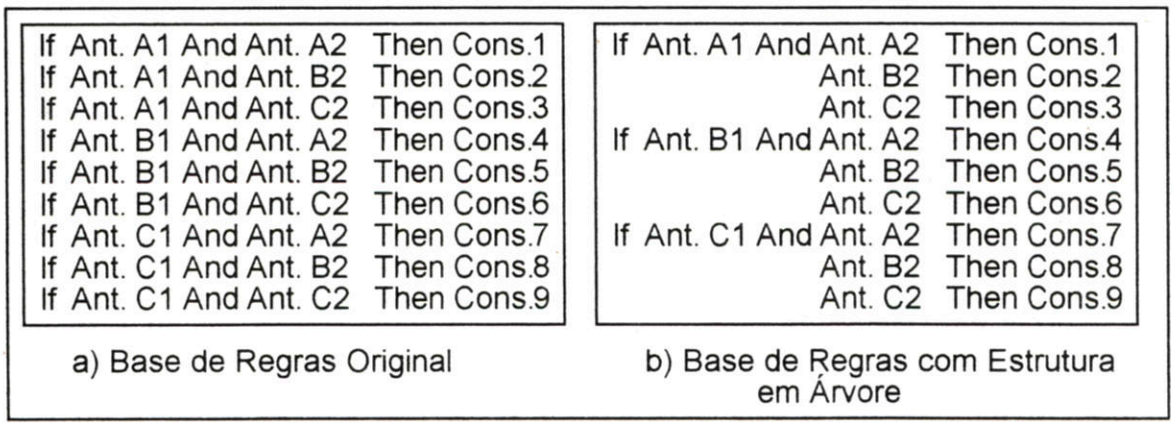


Fig. 6.14 a) Base de Regras Original; b) Base de Regras com Estrutura em Árvore

Para o sistema em árvore a variável *Grau* foi substuída por um vetor com um número de posições igual ao número de variáveis de entrada. O objetivo deste vetor é guardar em sua posição mais elevada o grau de pertinência mínimo entre os antecedente que já foram avaliados em uma determinada regra. A posição do vetor é indexada a cada instante pela profundidade em que o sistema se encontra dentro da árvore e que seria, no máximo, o número de variáveis de entrada (ou o número de antecedentes).

Assim que é detectado um grau de pertinência igual a zero em qualquer profundidade da árvore o sistema aborta a avaliação para qualquer dos antecedentes que estão a um nível mais baixo do ramo atual.

A estrutura proposta forneceu um ganho de, aproximadamente, 40% no tempo de execução, sendo que uma iteração de todo o sistema passa a consumir 150 ms, o que corresponde ainda a um tempo maior que 100% em relação ao desempenho obtido em [Cav 94].

A análise que se seguiu a partir de então foi no sentido de apontar as fontes do consumo de tempo dentro do códico em forth, ou seja, dentro da definição das palavras.

Foi constatado que o tempo de execução para uma regra com quatro antecedentes encontrava-se em, aproximadamente, 100 μs. Analisando cada uma das palavras que compõem essa regra, verificou-se que as palavras *And* e *Then*, que possuem o mesmo código de implementação, são responsáveis pela maior contribuição no que diz respeito ao consumo do tempo de uma regra.

As palavras *And* e *Then* são palavras que executam instruções de busca em memória e de comparações de valores, basicamente. Destas, as operações de busca em memória foram apontadas como as principais responsáveis pelo consumo excessivo de tempo.

Como a implementação das palavras de busca em memória que estão na definição de *And* e *Then* é feita com operações simples que não justificam o tempo consumido, concluiu-se que o grande responsável pelo consumo de tempo é o interpretador forth.

Neste ponto foram realizadas diversas hipóteses sobre o interpretador que, levadas ao conhecimento da Gyron Tecnologia, contribuíram para o planejamento de uma re-estruturação deste interpretador.

Uma destas hipóteses seria que, se a busca às novas palavras definidas no dicionário Forth ocorre de maneira seqüencial, uma solução mais adequada seria a organização das novas palavras através de uma *hashtable*, permitindo uma economia de tempo considerável nas operações de busca em memória.

6.4 CONCLUSÕES

Este capítulo apresenta a simulação do sistema automático de navegação do helicóptero da Gyron Tecnologia, utilizando o controlador nebuloso implementado sobre a plataforma em forth para transputer. Buscou-se através desta simulação, verificar o funcionamento do controlador nebuloso e de todo o sistema para a plataforma citada. Chegou-se à conclusão de

que a capacidade de processamento paralelo do transputer torna-o um processador adequado para o tratamento dos acoplamentos do helicóptero.

Concluiu-se também que o desempenho do sistema manteve-se abaixo da expectativa em função de diversos elementos, dentre eles, alguns que apontaram para o interpretador utilizado.

Os resultados obtidos com o controlador nebuloso (fuzzy) foram considerados, numa primeira análise, adequados em relação aos obtidos com o controlador padrão. Entretanto torna-se necessário um estudo mais aprofundado para explicar as divergências encontradas.

CONCLUSÕES E PERSPECTIVAS

Este trabalho apresentou a implementação de um controlador nebuloso que foi projetado em [Cav 94]. Sendo assim, os aspectos teóricos sobre o helicóptero, sobre controladores nebulosos e sobre a estrutura da missão aqui apresentados foram baseados nesse trabalho.

A implementação foi feita em linguagem Forth, utilizando o interpretador desenvolvido por Gyron Tecnologia, favorecendo uma estruturação do código para as máquinas de inferência e a geração automática de código a partir da tradução de tabelas de regras lingüísticas. Esta geração automática proporciona uma economia razoável de tempo, facilitando o desenvolvimento das máquinas de inferência. Além do mais permitirá à empresa a utilização no sistema de controle do helicóptero, de qualquer aplicação que requeira lógica nebulosa, já que essa máquina de inferência opera na arquitetura computacional do helicóptero.

O trabalho aqui desenvolvido exigiu o conhecimento de diversas áreas dentre elas as áreas de controle de processos, inteligência artificial e informática industrial. O estudo e utilização de controladores nebulosos pode ser visto como a grande contribuição dada por este trabalho à formação acadêmica e profissional, principalmente por se tratar de um tema recente e que oferece muitas perspectivas para pesquisa.

A oportunidade de estar em contato com profissionais pesquisadores do CTI, da empresa (Gyron) e com professores ligados à área de inteligência artificial, contribuiu para a formação de uma organização de raciocínio e de firmeza nas conclusões. Este contato foi fértil também no sentido de criar um espírito profissional.

Como perspectivas futuras para o trabalho, é ainda necessário um investimento na melhoria da performance computacional da máquina de inferência. Entretanto o trabalho detectou que essa melhora de performance está principalmente ligada ao aprimoramento da execução de algumas palavras básicas do interpretador forth, e não do código da máquina de inferência. Em relação às divergências obtidas entre os resultados obtidos na operação do algoritmo de controle e a aplicação padrão, torna-se necessário fazer um estudo mais aprofundado, que implica também em caracterizar melhor os resultados obtidos pelo sistema de referência. A pista para o problema deve estar no fato de uma máquina ser em ponto flutuante enquanto que a outra é inteira.

Algumas das sugestões que surgiram do contato com professores com especialização em áreas afins, foi da utilização de uma linguagem compilada (mesmo forth) ao invés de interpretada, otimização da busca de palavras no dicionário forth com a criação de uma tabela *hashing* para as palavras definidas pelo usuário e utilização de um número maior de processadores aproveitando as características do transputer.

BIBLIOGRAFIA

- [Aliev et al 91]: R.Aliev, F.T.Aliev, M.Babaev: *Fuzzy Process Control and Knowledge Engineering in Petrochemical and Robotic Manufacturing*, Verlag TÜV Rheinland, Bönn, 1991;

- [Altrock et al 92]: C.von Altrock, B.Krause, H.-J.Zimmermann, "Advanced Fuzzy Logic Control of a Model car in Extreme Situations", *Fuzzy Sets and Systems*, vol. 48 pp. 41-52, 1992;

- [Aptronix 93]: *Fuzzy Logic- from Concept to Implementation*, Aptronix Inc., 1993;

- [Bernard 88]: J.A.Bernard, "Use of a Rule-Based System for Process Control", *IEEE Control System Magazine*, vol. 8, No. 5, pp. 3-13, outubro 1988.

- [Braae and Rutherford 79a]: M.Braae and D.A.Rutherford, "Selection of Parameters for a Fuzzy Logic Controller", *Fuzzy Sets and Systems*, vol. 2, pp. 185-199, 1979.

- [Braae and Rutherford 79b]: M.Braae and D.A.Rutherford, "Theoretical and Linguistic Aspects of the Fuzzy Logic Controller", *Automatica*, vol. 15, pp. 553-577, 1979;

- [Buckley 91]: J.J.Buckley, "Fuzzy I/O Controller", *Fuzzy Sets and Systems*, (?), pp. 127-137, 1991;

- [Cav 94]: C.Cavalcante, "Sistema de Navegação para Helicópteros não Tripulados Utilizando Controlador Nebuloso", Dissertação de Mestrado, CPGEEL, UFSC, Junho de 1994;

- [Chiu et al 91]: S.Chiu, S.Chag, D.Moore, A.Chaudhaary, "Fuzzy Logic for Control of Roll and Moment for a Flexible Wing Aircraft", *IEEE Control Systems*, vol. 11, N° 4, pp. 42-48, June 1991;

- [Dubois and Prade 88]: D.Dubois and H.Prade, "An Introduction to Possibilistic and Fuzzy Logics", in *Non-Standart Logics for Automated Reasoning*, pp. 288-326, Academic Press Limited, 1988.

- [Ketata 92]: R.Ketata: *Methodologies de Regulation Numerique incluant la Logique Floue*, These (Spécialité Automatique-Informatique Industrielle), LAAS-CNRS, Toulose, France, Juillet 1992;

- [Kickert and van Nauta Lemke 76]: W.J.M.Kickert and H.R.van Nauta Lemke, "Application of a Fuzzy Controller in a Warm Water Plant", *Automatica*, vol. 12, pp. 301-308, 1976;

- [Klir and Folger 88]: G.J.Klir and T.A.Folger: *Fuzzy Sets, Uncertainty and Information*, Prentice-Hall, New Jersey, 1988;

- [King and Mandani 77]: P.J.King and E.H.Mandani, "The Application of Fuzzy Control Systems to Industrial Processes", *Automatica*, vol. 13, pp.235-242, 1977.

- [Kröhling 94]: R.A.Kröhling: "Algoritmos de Controle não Convencionais: Estudo de um Problema Clássico", Tese de Mestrado, Universidade Federal do Espírito Santo, 1994;

- [Lee 90a]: C.C.Lee, "Fuzzy Logic in Control Systems: Fuzzy Logic Controller - Part I", *IEEE Transactions on Systems, Man and Cybernetics*, vo;. 20, No. 2, pp. 404-418, March/April 1990;

- [Li and Lau 89]: Y.F.Li, C.C.Lau, "Development of Fuzzy Algorithms for Servo Systems", *IEEE Control Systems Magazine*, pp. 65-72, April 1989;

- [Lukasiewicz 75]: J.Lukasiewicz: *Estudios de Lógica y Filosofia*, Biblioteca de La Revista de Occiedente, Madrid, 1975;

- [Mizumoto and Zimmermann 82]: M.Mizumoto and H.J.Zimmermann, "Comparison of Fuzzy Reasoning Methods", *Fuzzy Sets and Systems*, vol. 8, pp. 253-283 82, 1982;

- [Ogata 82]: K.Ogata: *Engenharia de Controle Moderno*, Prentice/Hall do Brasil Ltda., Rio de Janeiro, 1982;

- [Pallet 83]: E.H.J.Pallet, *Automatic Flight Control* , Granada, Herts, 1983;

- [Pedrycz 89]: W.Pedrycz: *Fuzzy Control and Fuzzy Systems*, Research Studies Press Ltda., Jonh Wiley, New York, 1989;

- [Ruan and Kerre 93]: D.Ruan, E.E.Kerre, "Fuzzy Implication Operators and Generalized Fuzzy Method of Cases", *Fuzzy Sets and Systems*, vol. 54, pp. 23-27, 1993;

- [Rutherford and Bloore 76]: D.A.Rutherford and G.C.Bloore, "The Implementation of Fuzzy Algorithms for Control", *Proceedings of the IEEE*, pp. 572-573, April 1976;

- [Self 90]: K.Self, "Designing with Fuzzy Logic", *IEEE Spectrum*, pp. 42-44 e 105, November 1990;

- [Sugeno and Kang 86]: M.Sugeno and G.T.Kang, "Fuzzy Modeling and Control of Multilayer Incineration", *Fuzzy Sets and Systems*, vol. 18, pp. 329-346, 1986;

- [Sugeno and Nishida 85]: M.Sugeno and M.Nishida, "Fuzzy Control of Model Car", *Fuzzy Sets ans Systems*, vol. 16, pp. 103-113, 1985;

- [Takagi and Sugeno 83]: T.Takagi and M.Sugeno, "Derivation of Fuzzy Control Rules from Human Operator's Control Actions", in *Proc. of the IFAC Symp. on Fuzzy Information Knowledge Representaion and Decision Analisys*, Marseilles, France, pp. 55-60, July 83;

- [Tang and Mulholland 87]: K.L.Tang, R.J.Mulholland, "Comparing Fuzzy Logic with Classical Controller Design", *IEEE Transactions on Systems, Man and Cybernetics*, vol. SMC-17, N°6, pp. 1085-1087, November/December 1987;

- [Togai 92]: Togai Infralogic, "Motor Control: Fuzzy Logic Wins out over Classical Method", mailing list "fuzzy_server@til.com",1992;

- [Tong 76]: R.M.Tong, "Analysis of Fuzzy Control Algorithms using the Relation Matrix", *International Journal Man-Machine Studies*, N° 8, pp. 679-686, 1976;

- [Vachtsevanos 93]: G.Vachtsevanos, S.S.Farinwata, D.K.Pirovolou, "Fuzzy Logic Control of an Automative Engine", *IEEE Control Systems*, vol. 13, N° 3, pp. 62-68, June 1993;

- [Viot 93]: Greg Viot, "Fuzzy Logic in C", *Dr. Dobb's Journal*, February 1993;

- [Zadeh 73]: L.A.Zadeh, "Outline of a New Approach to the Analysis of Complex Systems and Decision Processes", *IEEE Transactions on Systems, Man and Cybernetics*, vol. SMC-3, No. 1, pp. 28-44, January 1973;

- [Zadeh 84]: L.A.Zadeh, "Making Conputers Think like People", *IEEE Spectrum*, pp. 26-32, August 1984.