UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
ENGENHARIA MECATRÔNICA

PROJETO DE GRADUAÇÃO 2

CONSTRUÇÃO DE UMA BANCADA EXPERIMENTAL BARRA-ESFERA E APLICAÇÃO DE CONTROLE FUZZY

Aluno
João Miguel Ferreira Cunha

Orientadores
Flávia Maria Guerra de Sousa Aranha Oliveira
Geovany Araújo Borges

Relatório do Projeto de Graduação apresentado ao fim do curso de graduação, como requisito parcial para a obtenção do grau de Engenheiro, em julho de 2005 na Universidade de Brasília.

Brasília, julho de 2005

Dedicatória

Dedico esse trabalho aos meus pais que sempre me incentivaram a crescer continuamente nos meus estudos.

João Miguel Ferreira Cunha

Agradecimentos

Gostaria de agradecer pela realização desse trabalho minha família e minha namorada que sempre foram inspiração para o meu trabalho e desenvolvimento pessoal.

Também gostaria de agradecer aos meus professores orientadores profa.Flávia Souza e prof.Geovany Borges que foram para mim exemplo de ótimos profissionais e colegas durante o trabalho.

Agradeço aos técnicos do laboratório SG9 que me ajudaram com muita paciência na realização desse projeto.

E agradeço por último e não menos aos meus colegas e amigos de curso pelos momentos de descontração e ajuda em todos os momentos críticos que enfrentei.

ÍNDICE ANALÍTICO

LISTA DE FIGURAS

RESUMO

Este trabalho consiste na construção de uma bancada experimental *Barra-esfera* para estudo teórico e prático da teoria de controle, e para aplicação final de um controlador fuzzy. Assim, este trabalho contém a descrição dos passos executados para a construção da plataforma experimental, desde a concepção física do projeto e instrumentação eletrônica à abordagem teórica e prática necessária para o desenvolvimento do controlador fuzzy. O trabalho apresenta como resultados a resposta do sistema simulado e a resposta experimental com a utilização de controladores fuzzy com 16 e 81 regras.

ABSTRACT

This report consists in the construction of a *Ball and Beam* experiment test-bed for theoretical studies and practice of control theory, and for a final application of a fuzzy controller. Thus, this report contains a description of the executed steps in the construction of the test-bed, since the project physical conception and electronic instrumentation until the practical and theoretical approach necessary to the development of the fuzzy controller. The report presents as results the response of the simulated system and the experimental systems using fuzzy controllers with 16 and 81 rules.

1. INTRODUÇÃO

1.1 Motivação

1.1.1 Bancadas Experimentais

Assim como a teoria é fundamental para o engenheiro, a prática é imprescindível para que possam ser exploradas características inesperadas em um sistema. Nesse sentido, a criação de plataformas experimentais para o estudo e prática da teoria de controle é comum em âmbito acadêmico.

Ainda, é por meio da construção de bancadas experimentais que o estudante de graduação pode entrar em contato com outro aspecto fundamental da engenharia: a concepção de projetos, que envolve ainda mais aspectos imprevisíveis.

Algumas plataformas bastante conhecidas em âmbito acadêmico são o pêndulo invertido, o levitador magnético e sistema de nível de líquidos. A Universidade de Brasília possui esses três sistemas disponíveis e grandes trabalhos e publicações já foram feitos usando-os como plataforma de estudo.

Dessa forma, é possível dizer que a construção de bancadas experimentais é de grande valia tanto para a universidade, que ganha a oportunidade de realizar mais pesquisas e publicações, como para o aluno agrega bastante conhecimento prático e teórico, por meio da concepção do projeto e posteriores estudos sobre a plataforma.

1.1.2 Controle Fuzzy

A engenharia de controle atualmente possui técnicas consolidadas para projetos de sistemas e controladores. Métodos ditos convencionais como controle PID ou controle em Espaço de Estados possuem ferramentas bastante úteis nas etapas de projeto e análise de desempenho, baseadas em uma linguagem matemática que sustenta sua qualidade. Porém, esses métodos possuem limitações

e restrições que estão associadas à complexidade ou existência de modelos matemáticos lineares nos sistemas. Nesse sentido, o Controle Fuzzy apresenta-se como uma técnica alternativa que procura contornar essas limitações dos métodos convencionais.

A base para o desenvolvimento de controladores fuzzy foi apresentada em 1964 por Zadeh[1]. Neste trabalho foram descritas formas de se representar matematicamente expressões lingüísticas por meio da utilização de uma lógica multivalorada, a lógica fuzzy. Entretanto, apenas em 1974, o pesquisador Mamdani[1] apresentou o primeiro trabalho utilizando a lógica fuzzy na implementação do controle de uma máquina a vapor, ao tentar emular o comportamento do operador tradicional em um computador.

A praticidade da metodologia do controle fuzzy levou a sua crescente aplicação, principalmente no Japão, onde se estima que o gasto com produtos fuzzy fabricados em 1992 girou em torno de 2 bilhões de dólares [1]. Em [5], são citados alguns produtos fuzzy que já foram desenvolvidos. Entre eles estão o controle de uma estação de tratamento de água, o controle de uma máquina de lavar, o controle de um reator nuclear, o controle de transmissão automática de carros, sendo destacados o sistema automático de operação de trens na cidade japonesa de Sendai e o carro desenvolvido pelo pesquisador T.Sugeno, que seguia uma trajetória e estacionava em sua garagem de forma autônoma.

A partir dessas aplicações, essa nova abordagem de controle foi observada mais seriamente pela acadêmia. O que antes dependia de descrições matemáticas para ser controlado passou a ser estudado de forma mais intuitiva, baseada na investigação de ações humanas que devem ser tomadas sobre o processo. Esse método, procura sistematizar a "tradução" do conhecimento humano (mais especificamente, do chamado "especialista") para uma linguagem que possa ser implementada em controladores, tornou a estratégia de controle mais simples e independente das restrições matemáticas de modelos.

1.1.3 A Bancada *Barra-esfera*

O sistema *Barra-esfera* é uma plataforma experimental também bastante conhecida no mundo acadêmico. Ela tem como princípio o desafio de controlar a posição de uma esfera sobre uma

barra, cujo ângulo pode ser variado. Por apresentar bom apelo visual, esse sistema é bastante utilizado nas universidades para estudo da teoria de controle não-linear e também para a aplicação de técnicas mais avançadas de controle.

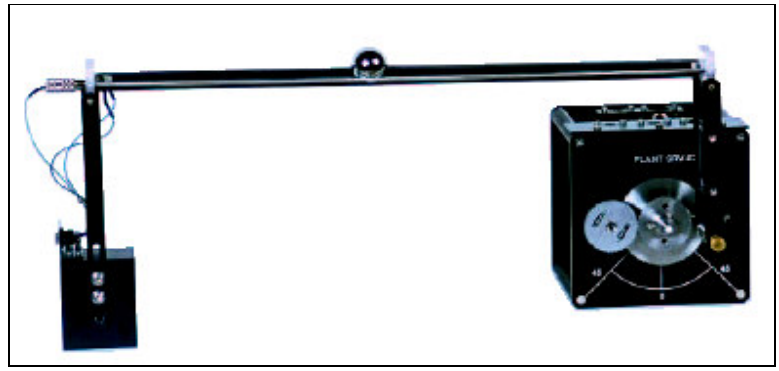

figura 1 : Exemplo de uma bancada experimental *Ball and Beam*

Diversos artigos na área de controle não-linear utilizam o sistema *barra-esfera* como base para a obtenção de seus resultados. E exatamente por estas características de não-linearidade e por exigir uma modelagem matemática mais complexa, esse sistema é também bastante citado em estudos de controle fuzzy. As suas características não-lineares e o senso intuitivo agregado ao experimento justificam a aplicação de técnicas fuzzy no controle deste sistema. Duas teses de mestrado [6] e [7] podem ser citados como exemplo de como a bancada é utilizada com em importantes estudos.

1.2 Objetivos

Nesse sentido, o objetivo desse trabalho é a construção de uma bancada experimental *barra-esfera* visando a aplicação de controle fuzzy sobre a mesma.

Assim pretende-se com o trabalho projetar a plataforma experimental desde sua base mecânica até a implementação do controle fuzzy da posição da esfera. O projeto da bancada visa ser flexível de forma que outras métodos de controle possam ser implementados e diferentes estudos possam ser feitos sobre a bancada.

Ainda, espera-se com esse trabalho, contribuir como para o estudo de controle fuzzy, que não é lecionado normalmente durante o curso de graduação de Engenharia Mecatrônica na Universidade de Brasília.

1.3 Estrutura do Trabalho

Esse trabalho é divido nos seguintes capítulos:

1. **Introdução**: Neste capítulo é introduzido o tema do trabalho, junto com a motivação e objetivos do mesmo.

2. **Revisão Bibliográfica**: aqui são apresentados de forma resumida os tópicos teóricos que foram mais relevantes ao trabalho, de forma que o capítulo é dividido em 3 subseções - Modelamento Dinâmico do Sistema Barra-Esfera, Modelamento Dinâmico de um Motor DC e Controle Fuzzy.

3. **Desenvolvimento**: Neste capítulo é descrita a forma como foi implementada a bancada e como os resultados do trabalho foram alcançados. A apresentação do capítulo segue a seqüência de passos executados ao longo do trabalho de implementação: Construção da Bancada Experimental, Instrumentação, Abordagem para Controle, Controlador Fuzzy, Simulação e Implementação.

4. **Resultados**: a apresentação e análise dos resultados atingidos na simulação e na implementação do sistema é feita neste capítulo..

5. **Conclusão**: Neste capítulo são apresentados observações acerca dos objetivos do proposto e o que foi feito no trabalho.

2. REVISÃO BIBLIOGRÁFICA

2.1 Modelo Dinâmico Barra-Esfera

O sistema barra-esfera é composto de uma barra livre que girar em torno de um eixo e de uma bola que desliza sobre a pista. O objetivo do sistema de controle para este sistema é posicionar a esfera sobre qualquer ponto da barra rotacionando-a sobre o eixo.

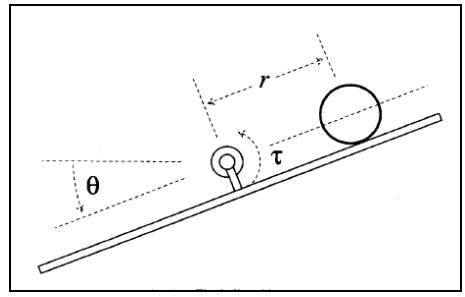

figura 2 : Sistema barra-esfera

A equação (1) que representa a dinâmica desse sistema é obtida por meio das equações lagrangianas, e é dada por [10]:

$$0 = (\frac{J_b}{R^2} + M) \cdot \ddot{r} + M \cdot G \cdot sen(\theta) - M \cdot r \cdot \dot{\theta}^2 \qquad (1)$$

, onde: J_b é o momento de inércia da bola, M é a massa da bola, G é aceleração da gravidade local, R é o raio da bola, θ, $\dot{\theta}$ e $\ddot{\theta}$ são respectivamente a posição, a velocidade e aceleração do ângulo da pista e r, $\dot{r}$ e $\ddot{r}$ são respectivamente a posição, a velocidade e aceleração da bola.

Assim, considerando a equação (1), e definindo $B = M/(\frac{J_b}{R^2} + M)$ a descrição em espaço do sistema é dada como:

$$\begin{bmatrix} \dot{r} \\ \ddot{r} \\ \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} \dot{r} \\ B \cdot (r \cdot \dot{\theta}^2 - G \cdot sen\,\theta) \\ \dot{\theta} \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \cdot u \qquad (2)$$

, sendo a entrada do sistema *u* a aceleração angular da pista.

Observando o espaço de estados (2) pode-se perceber as não linearidades do sistema representadas nos termos $r \cdot \dot{\theta}^2$ e $sen\,\theta$, o que impossibilita uma abordagem direta dos métodos convencionais de controle.

No entanto, esse sistema pode ser linearizado em torno de pequenos ângulos de forma que: $sen\,\theta \approx \theta$ e $\dot{\theta}^2 \approx 0$, como sugerido em [carnegie mellon]. Essa aproximação resulta na equação (3) e no modelo de espaço de estados linearizado (4).

$$0 = (\frac{J_b}{R^2} + M) \cdot \ddot{r} + M \cdot G \cdot \theta \qquad (3)$$

$$\begin{bmatrix} \dot{r} \\ \ddot{r} \\ \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -B \cdot G \cdot \theta & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} r \\ \dot{r} \\ \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \cdot u \qquad (4)$$

Ambos os sistemas foram considerados nas simulações. Os parâmetros físicos do sistema foram medidos e seus valores estão indicados nos arquivos de simulação.

2.2 Modelo Dinâmico de Motores DC

Motores DC são motores de corrente contínua, geralmente designados por servomotores, e são usados em diversas aplicações. Tanto para sistemas de baixa potência, em equipamentos periféricos de computadores, como para sistemas de baixa potências, sistemas robóticas e máquinas de comando numérico, seu uso é bastante difundido.

Um servomotor é comumente representado por uma parte elétrica, uma resistência em série com uma indutância e com a tensão induzida, e uma parte mecânica, que relaciona o acoplamento do motor com a carga do sistema. Essa representação está ilustrada na figura 3.

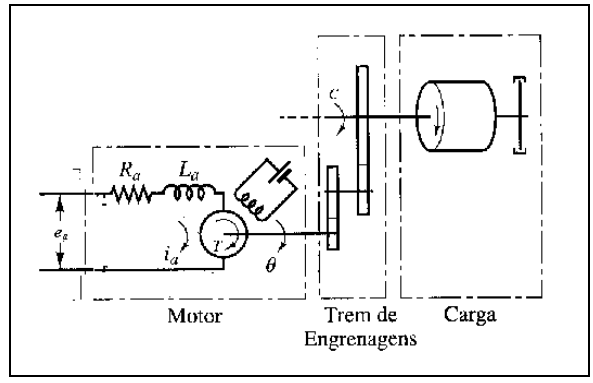

figura 3 : Representação de um servomotor

A partir dessa representação, as equações da dinâmica do motor são obtidas, dadas por [14]:

$$L_a \frac{di_a}{dt} + R_a i_a + K_3 \frac{d\theta}{dt} = e_a \qquad (5)$$

$$J_o \frac{d^2\theta}{dt^2} + b_o \frac{d\ \theta}{dt} = K_3 \cdot i_a - \frac{\mathrm{T}_{carga}}{n} \qquad (6)$$

, onde L_a, R_a são a indutância e resistência do motor, K_3 e K_2 são a constante de força-eletromotriz e a constante de torque, J_o e b_o representam o momento de inércia e o atrito viscoso da combinação entre carga, trem de engrenagens referidas ao eixo do motor, e n é a relação de engrenagens de forma que o ângulo de saída $c = \theta/n$.

Na equação (6) o termo $\frac{\mathrm{T}_{carga}}{n}$ pode ser desprezado caso a relação de engrenagens n seja alta o suficiente quando comparado a T_{carga}.

Assim, a função de transferência que representa o sistema é dada por [14];

$$\frac{\theta(s)}{e_a(s)} = \frac{K_2/n}{s^2(L_a \cdot s + R_a)\cdot(J_o \cdot s + b_o) + K_2 \cdot K_3 \cdot s} \qquad (7)$$

Considerando que $c = \theta/n$, e que a indutância L_a é muito pequena e pode ser desprezada para grande parte dos servomotores, a função de transferência fica como:

$$\frac{C(s)}{e_a(s)} = \frac{K_2/n}{s^2(R_a J_o) + s\cdot(b_o + K_2 \cdot K_3)} \qquad (8)$$

A partir dessa equação, o espaço de estados do sistema é fica sendo:

$$\begin{bmatrix} \dot{c} \\ \ddot{c} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & -(b_o + K_2 \cdot K_3)/(R_a J_o) \end{bmatrix} \cdot \begin{bmatrix} c \\ \dot{c} \end{bmatrix} + \begin{bmatrix} 0 \\ K_2 / n \end{bmatrix} \cdot e_a \qquad (9)$$

O sistema definido por (9) foi usado para implementar a simulação. Os parâmetros do motor foram adquiridos no catálogo técnico do mesmo.

2.3 Controle Fuzzy

2.1.1 Estrutura do Controlador Fuzzy

Controle Fuzzy é uma técnica de controle baseada em informações heurísticas sobre o processo. Como citado em [8], o "controle fuzzy fornece uma metodologia formal para representar, manipular e implementar conhecimento humano heurístico sobre como controlar um sistema". De forma mais clara, o controle fuzzy usa, em vez de equações matemáticas, equações lingüísticas na forma "**se** - **então**" para projetar controladores. Desse modo, usando esta linguagem alternativa a limitação da necessidade de modelos matemáticas é contornada. O controle fuzzy também é conhecido como controle especialista, onde suas ações são representadas pelas ações de um controlador especialista como ilustra a figura 4.

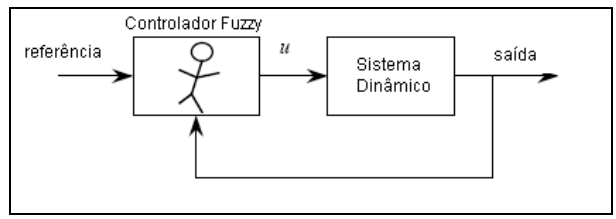

figura 4: Representação de um controlador fuzzy

De maneira geral, o controlador fuzzy é composto por 4 partes: fuzzificador, base de regras, máquina de inferência e defuzzificador, como são mostrados na figura 5.

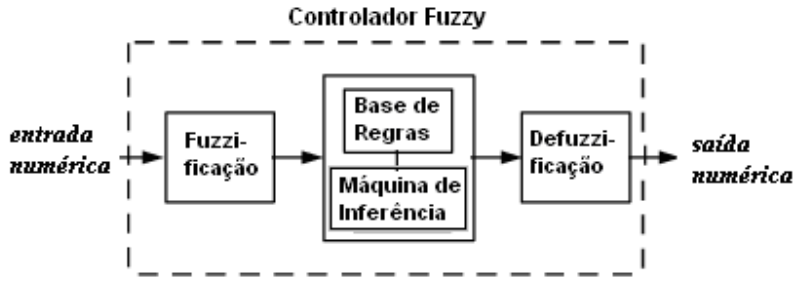

figura 5 : Esquema de um controlador fuzzy

A partir desses componentes o controlador fuzzy processa as entradas numéricas, interpretando-as de forma lingüística, infere conclusões sobre a ação que dever ser tomada, e a traduz em uma ação numérica para o processo em questão. A seguir cada um dos componentes do controlador fuzzy é descrito.

Fuzzificador

Um controlador fuzzy atua com base em expressões lingüísticas que representam o conhecimento humano sobre o processo. Porém a informação disponibilizada pelo processo é dada de uma forma numérica. Desse modo, a fuzzificação se encarrega de traduzir para termos fuzzy as expressões numéricas do sistema. Tal tradução é feita a partir de um mapeamento das entradas em conjuntos fuzzy definidos pela lógica fuzzy.

Os conjuntos fuzzy são representados por termos lingüísticos como por exemplo positivo, negativo, grande, pequeno e são definidos por funções de pertinência (vide apêndice para mais detalhes).

Alguns métodos de fuzzificação citados em [2] e descritos em anexo são: fuzzificação singleto, fuzzificação gaussiana e fuzzificação triangular.

Base de Regras

A representação das ações a serem tomadas pelo controlador fuzzy são armazenadas na base de regras. Como citado anteriormente, a ação do controlador fuzzy é descrita por meio de regras na

forma "**se – então**". Usando como exemplo o controle de temperatura em uma sala por meio de um ar condicionado, a base de regras para o processo poderia ser descrito por:

***SE* temperatura é fria *ENTÃO* potência do ar condicionado é baixa**

***SE* temperatura é morna *ENTÃO* potência do ar condicionado é média**

SE temperatura é quente ENTÃO potência do ar condicionado é alta

---

figura 6 : Base de regras para controle de temperatura

Desse modo, a base de regras representa o conhecimento especialista embarcado sobre o controlador fuzzy. No exemplo descrito, podem ser incluídas mais variáveis sobre o processo, como por exemplo a umidade ou número de pessoas na sala, caso seja necessário.

Máquina de Inferência

Cada regra de um controlador fuzzy possui sua premissa e sua conseqüência (**se** premissa **então** conseqüência). Ao avaliar somente uma regra, seu resultado pode ser inferido a partir da aplicação direta das operações definidas pela lógica fuzzy (operações t-norma e operação de implicação). No entanto, a base de regras geralmente é composta de mais de uma regra e faz se necessário utilizar um método para inferir resultados a partir de todo o conjunto de regras.

Assim, a máquina de inferência de um controlador fuzzy é responsável por processar as regras de acordo com as entradas fuzzificadas gerando uma conclusão fuzzy, que será posteriormente defuzzificada.

Em [2] são descritas algumas máquinas de inferência tendo como destaque a máquina de inferência Produto e a máquina de inferência Mamdani, que estão descritas com detalhes no apêndice.

Defuzzificador

Essa etapa do controlador fuzzy é responsável por mapear a saída fuzzy resultante da máquina de inferência para um valor numérico. O processo de defuzzificação pode ser executado de várias formas, sendo citado em [wang] os métodos Centro de Gravidade (COG) e Centro Médio (COA) como os mais utilizados (vide anexo para descrição dos métodos).

Em [8] é apresentado o processo de defuzzificação Centro de Gravidade para Singletons (COGS) dado por:

$$u = \frac{\sum_i u(s_i) \cdot s_i}{\sum_i u(s_i)} \tag{10}$$

, onde u é a saída defuzzificada, $u(s_i)$ são os valores inferidos das regras (ou o valor resultante da avaliação das premissas) e s_i é a posição do valor singleton das variáveis de saída de cada regra.

Esse método é apresentado como um caso singular do método generalizado COG e também pode ser considerado uma forma singular do método COA.

2.1.2 Projeto de um Controlador Fuzzy

O projeto de um controlador fuzzy é baseado na definição de cada um dos processos descritos anteriormente. No entanto, não existe nenhum procedimento sistemático para o projeto de tais etapas, como afirma [5]. Ainda, percebe-se que são inúmeros os parâmetros envolvidos na criação de cada um dos processos do controlador. Por exemplo, apenas a escolha das funções de pertinência apresenta como opções funções triangulares, gaussianas, trapezoidais e várias outras como mostra a figura 7.

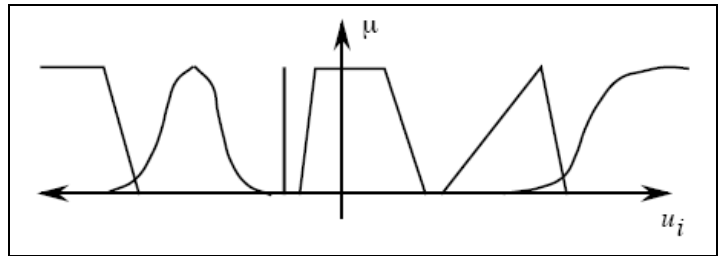

figura 7 : Exemplos de funções de pertinência

No entanto, apesar dos inúmeros parâmetros, existem características que devem ser consideradas quando forem criadas as partes de um controlador fuzzy e que auxiliam no projeto do mesmo. Em [4] é descrito de maneira simplificada os passos que são tomados no projeto de controladores fuzzy sugerindo como melhor forma de desenvolver tais controladores a prática e estudo de controladores já projetados. As etapas a serem seguidas são: Definição de variáveis e base de regras, quantificação do conhecimento e fuzzificação, inferência e defuzzificação.

1º Passo: Definição de variáveis e Base de Regras

Para inicia o projeto de um controlador fuzzy, deve-se inicialmente abordar o problema de forma intuitiva, como ilustra a figura 4. Ou seja, deve-se identificar quais as variáveis, do ponto de vista fuzzy, são necessárias para controlar o sistema.

Tomando como exemplo o controle de temperatura em uma sala, pode-se considerar como variáveis de entrado do controlador a temperatura e o número de pessoas na sala, e como variável de saída a potência do ar condicionado.

Ainda, deve ser decidido como qualificar essas variáveis de forma que suas características possam identificar ações de controle. Para tanto, podem ser usados termos como f**rio**, **morno e quente** relativos a temperatura e **muitas, poucas e nenhuma** para definir a variável pessoas. A ação do ar condicionada pode ser caracterizada como potência **nula**, **baixa**, **média, alta e muito alta**.

A partir desse ponto, a base de regras do sistema é definida de forma que as regras representem o comportamento do controlador. Uma forma de organizar a base de regras é a representação em

tabelas, onde são exploradas todas as combinações das entradas. Vê-se então que ao utilizar mais termos lingüísticos na descrição das variáveis, a combinação dos termos cresce exponencialmente, e assim o número de regras, o que pode tornar-se incompatível com o poder computacional em aplicações reais.

Base de Regras		Número de Pessoas		
		Nenhuma	Poucas	Muitas
Temperatura	Frio	Nula	Baixa	Média
	Morno	Baixa	Média	Alta
	Quente	Média	Alta	Muito Alta

figura 8 : Representação da base de regras para controle de temperatura (3x3 combinações)

Alguns métodos mais avançados como controle fuzzy em modo deslizante [13] e controle fuzzy hierárquico [3] administram esse problema diminuindo consideravelmente o número de regras utilizada.

2º Passo: Quantificação do Conhecimento e Fuzzificação

Definidos as variáveis e os termos utilizados, é preciso quantificá-las matematicamente para que a informação possa ser processada pelo controlador. Ou seja, devem ser definidas as funções de pertinência que irão representar os termos fuzzy.

São inúmeras as formas para definição de conjuntos fuzzy e nem sempre é claro qual deve ser usada. Alguns critérios a ser considerados na sua escolha são a complexidade computacional imposta pela curva e a sua continuidade, que é exigida em alguns casos pelo processo a ser controlado.

Funções triangulares e trapezoidais representam funções computacionalmente simples, porém apresentam um comportamento um tanto descontinuo em seus pontos de quebra. Já a classe de funções mais suaves, como a gaussiana e senoidal, por exemplo, apresentam uma maior continuidade em troca de se exigir maior poder computacional por parte do controlador. A escolha

das funções de pertinência também irá influenciar no processo de fuzzificação como será visto posteriormente.

Para o exemplo do ar condicionado, os termos que devem ser associados a conjuntos fuzzy são: {frio, morno, quente} para a temperatura e {nenhuma, poucas, muitas} para pessoas. Nesse ponto, algumas questões levantadas ao quantificar tais termos são como representar a forma que melhor representa a transição de graus de verdade entre os termos **poucas** pessoas e **nenhuma** pessoa, como escolher o valor do centro da funções que representam o valor máximo de verdade de um conjunto fuzzy etc.

Projeto da Fuzzificação

O processo de fuzzificação das entradas também deve ser estudado cuidadosamente em um projeto de um controlador fuzzy. Assim como a escolha das funções de pertinência, o processo de fuzzificação deve levar em conta a complexidade computacional de sua realização. Ainda, essa etapa pode ser definida de modo a suprimir ruídos sobre os sinais de entrada.

Combinando as escolhas {função de pertinência triangular, fuzzificação triangular} ou {função de pertinência gaussiana, fuzzificação gaussiana} é possível evitar o ruído sobre as medidas de entrada [2], enquanto a fuzzificação singleton não é capaz de tal ação. Em situações onde a filtragem de sinal é necessária, mas pode comprometer a dinâmica do sistema, a fuzzificação gaussiana ou triangular podem representar uma boa escolha para o processo de fuzzificação, já que o controlador fuzzy não influi na dinâmica do sistema, pois se apresenta apenas como um mapeamento não linear entre as variáveis de entrada e variáveis de saída [4].

3º Passo: Máquina de Inferência

O projeto da máquina consiste na escolha das operações s-nroma, t-norma, e implicação, e na forma como as regras são combinadas para formar uma única saída inferida. Neste processo de escolha, as operações min ou produto, max ou soma e implicação produto são consideradas boas escolhas já que são simples de serem descritas computacionalmente. Ainda, essas operações

possuem a propriedade de causa/efeito e não-causa/nenhum-efeito interessante em aplicações de engenharia.

A forma de combinação das regras está associada com a relação de independência entre as regras propostas. No caso, uma combinação de regras com intersecção afirma que todas as regras são dependentes entre si, e resultado individuais irão influenciar na saída fuzziy do processo de inferência. Do mesmo modo, a combinação união entre regras evita que resultados individiuais influenciem no resultado final da inferência. Assim, deve-se considerar como a base de regras está relacionada entre si para escolher o processo adequado para a máquina de inferência.

4º Passo: Defuzzificação

A escolha do processo de defuzzificação em um controlador fuzzy segundo [2] deve considerar alguns critérios como a plausibilidade – a saída defuzzificada deve representar de forma intuitiva os valores fuzzy resultantes da máquina de inferência -, a simplicidade computacional – a etapa deve ponderar o poder computacional a ser utilizado – e a continuidade – o processo de defuzzificação deve-se apresentar geralmente de forma contínua sem grandes variações na saída para pequenas variações da entrada.

Ao analisar o método COG (vide anexo), percebe-se que, caso sejam utilizadas funções de pertinência gaussianas, o processo de defuzzificação seria relativamente complexo, ao ser necessário calcular a área fuzzy inferida. Já para o método COA, as formas da função de pertinência das saídas podem ser de certa forma ignorada, pois esse processo utiliza apenas a informação do centro da função de pertinência para realizar seus cálculos. Dessa forma, é possível notar que a escolha feita sobre as funções de pertinência influencia a escolha do processo de defuzzificação.

Ainda, para os métodos de defuzzificação citados anteriormente e descritos no anexo, deve-se impor condições às regras em certos casos. Estas condições são traduzidas na necessidade de se evitar que os termos () () , para a defuzzificação COG e COA, não sejam nulos, o que seria uma singularidade sobre o controlador podendo levá-lo a instabilidade. Uma forma de se evitar tal situação é explorar todas as combinações possíveis das variáveis de entrada na base de regras.

2.1.3 Sintonia de um Controlador Fuzzy

Devido a grande quantidade de variáveis envolvidos no projeto de um controlador fuzzy, a sintonização do mesmo se torna uma tarefa bastante trabalhosa para o projetista. No entanto, existem formas de ajuste recomendadas que direcionam o projetista na sintonização do controlador. Em [4] sugere-se evitar a sintonização do controlador avaliando diferentes formas de operações fuzzy, de fuzzificação e defuzzificação, sendo proposto a introdução de ganhos de escalonamento sobre a entrada e saída do controlador fuzzy, como descrito a seguir.

Escalonamento das Entradas

A partir do uso de ganhos sobre as variáveis a serem fuzzificadas é possível estender ou contrair as funções de pertinência, como ilustra a figura 9. Esse ajuste permite atribuir novos significados as entradas. No caso, ao multiplicar as entradas por um ganho $0 < G < 1$ (estendo a função de pertinência) torna seus valores mais incertos. O mesmo acontece ao utilizar uma ganho $Gg > 1$ (figura 9), onde os valores que antes não possuíam significativo grau de certeza, passaram a ter maior valor de pertinência.

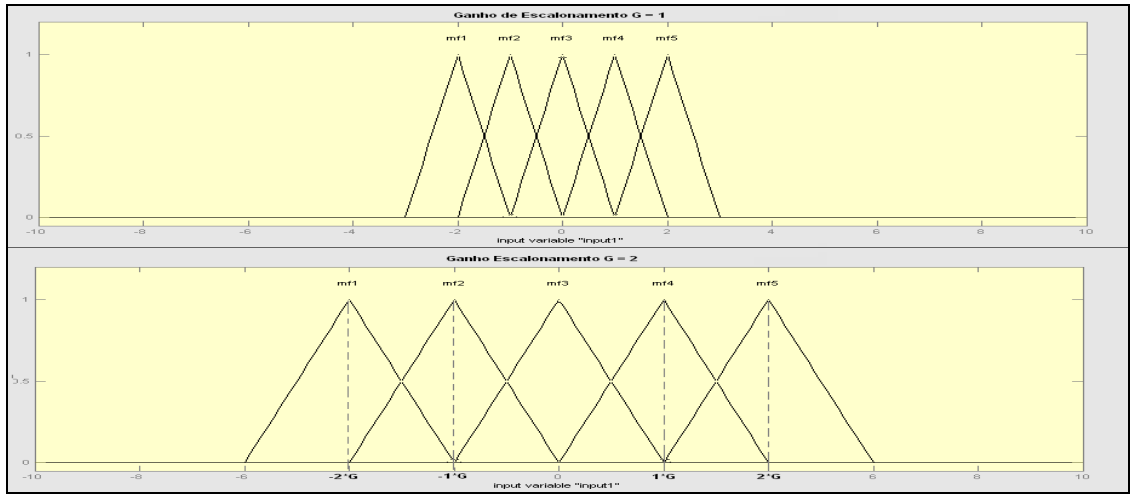

figura 9 : Escalonamento das entradas

Escalonamento da Saída

A aplicação de ganhos sobre as variáveis de saída do controlador fuzzy podem ter dois efeitos. O primeiro efeito é semelhante ao efeito de aplicação do ganho sobre as entradas, onde as variáveis de saída serão contraídas ou estendidas aumentando globalmente a atuação do sistema.

Outro efeito do escalonamento da saída é o ajuste dos ganhos locais, ou seja, ao deslocar de forma desigual as variáveis de saída é possível sintonizar o ganho do controlador fuzzy localmente. A figura 10 ilustra as duas situações para o escalonamento do ganho da saída.

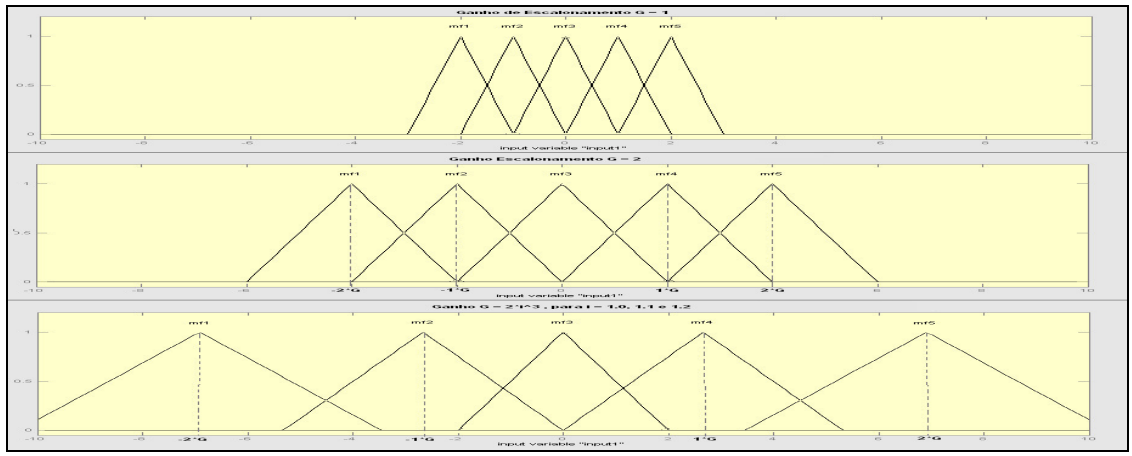

figura 10 : Aplicação de ganhos sobre as variáveis de saída do controlador fuzzy. Os ganhos usados são respectivamente G = 1, G=2 e G = 2*i^3 , para i = 1.0, 1.1 e 1.3

Superfície de Controle

Uma ferramenta que pode ser utilizada para auxiliar na sintonização do escalonamento dos ganhos é a superfície de controle. A partir dela, é possível perceber como o controlador se comporta para determinadas entradas. Assim, podem-se alterar os ganhos de entrada e de saída de forma a obter superfícies de controle desejadas. Nesse ponto, percebe-se a diferença entre um controlador fuzzy, caracterizado por uma superfície não-linear, e um controlador PD, como citado em [4].

Em [9], é estudado de forma prática o efeitos da utilização de algumas superfícies de controle típicas, ilustradas na figura 11.

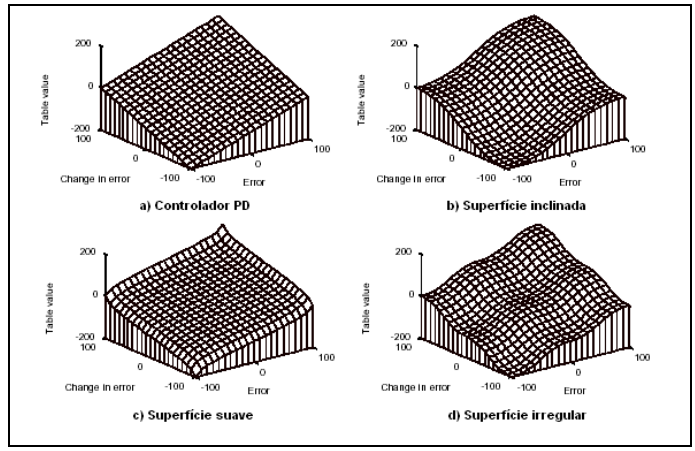

figura 11 : Alguns exemplos de superfícies de controle

3. DESENVOLVIMENTO

3.1 BANCADA EXPERIMENTAL

Como primeiro passo para construir a bancada experimental, foi adquirida a peça que iria constituir a pista onde seria posicionada e medida a posição da bola. Essa peça foi retirada de uma impressora ploter 2d e é composta de uma base de madeira com furos para fixação e um trilho resistivo, como mostra a figura 12.

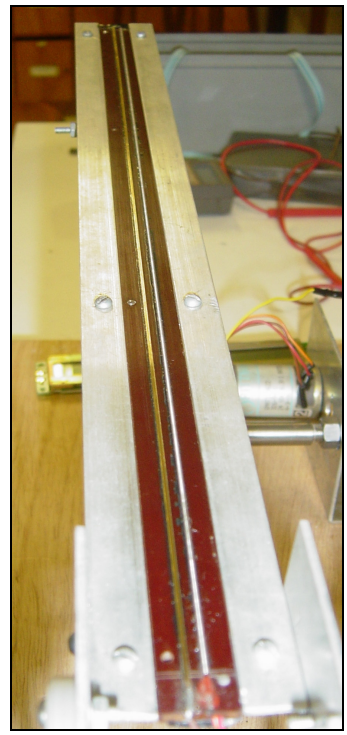

figura 12 : Pista para posicionamento e medição

De posse da pista resistiva, o segundo passo para montagem da bancada foi a aquisição da bola. Para tanto, foram usadas bolas de mouses inutilizados e bilhas de rolamento que eram feitas de metal condutivo, permitindo aproveitar o trilho resistivo para leitura da posição da bola. As bolas estão ilustradas na figura 13.

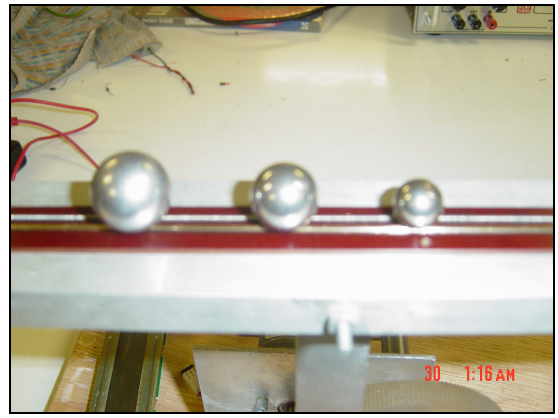

figura 13 : Bolas metálicas de mouse

Ainda, outra peça chave que seria usada como base para o projeto da estrutura da bancada seria o atuador que iria movimentar a pista. O atuador usado em questão foi um motor DC com redução 6.3:1, disponibilizado pelo prof.Lélio Ribeiro, do Departamento de Engenharia Elétrica da Universidade de Brasília.

De posse dessas três peças chave, foi feito o projeto da estrutura que iria compor a bancada experimental. O modelo esquemático que serviu de base para o projeto da estrutura está representado na figura 14.

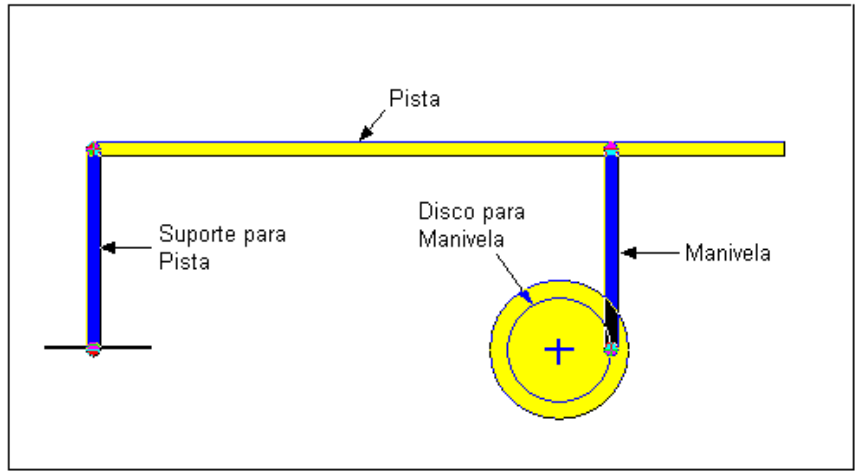

figura 14 : Representação esquemática da bancada experimental

Como indicado na figura 14, a estrutura pode ser divida em três partes principais: manivela, disco para manivela e suporte para pista. Foi verificado que todas essas partes influem na excursão do ângulo da pista, e por sua vez na dinâmica do sistema.

Assim, teve-se o cuidado de calcular as dimensões dessas partes que resultariam em uma excursão adequada do ângulo da pista.

Para calcular a excursão máxima do ângulo em função das dimensões do suporte, foram desenvolvidas as seguintes equações ((11), (12) e (13)), tomando como base a figura 15.

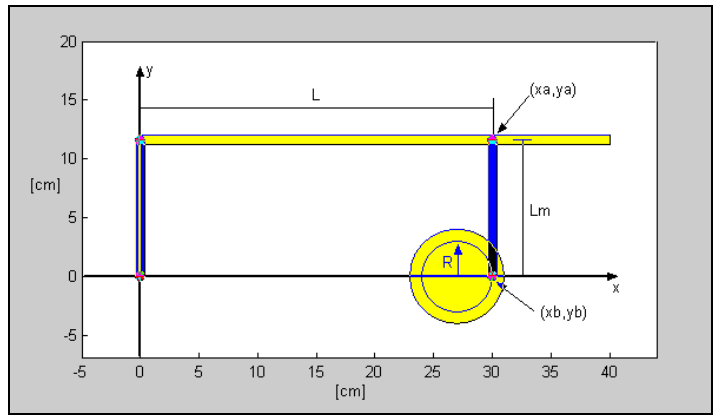

figura 15 : Equacionamento da excursão do ângulo

$$xa^2 + (ya - Lm)^2 = L^2 \tag{11}$$

$$xb = L - R + R \cdot \cos(\theta) \quad \text{e} \quad yb = R \cdot \sin(ang) \tag{12}$$

$$Lm^2 = (xa - xb)^2 + (ya - yb)^2) \tag{13}$$

Para resolver as equações (11), (12) e (13) foi feito um script em matlab que apresenta graficamente a relação entre o ângulo do disco da manivela e o ângulo da pista. A figura 16 ilustra essa relação para as dimensões L = 30cm, Lman = 12cm, R = 3cm. A excursão do ângulo para as dimensões é aproximadamente [-7.0 , 5.0] graus. Tal excursão foi considerada boa observando empiricamente a velocidade atingida pela bola quando colocada a pista sobre o ângulo máximo (positivo e negativo).

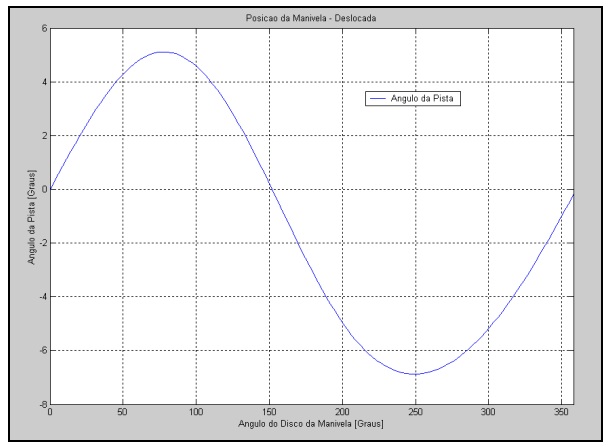

figura 16 : Relação entre ângulo do disco e ângulo da pista

A partir das medidas dimensionadas, foi feito a manivela, o disco para manivela e o suporte para a pista. Essas partes foram construídas na oficina SG9 com o auxílio dos técnicos e está ilustrada na figura 17.

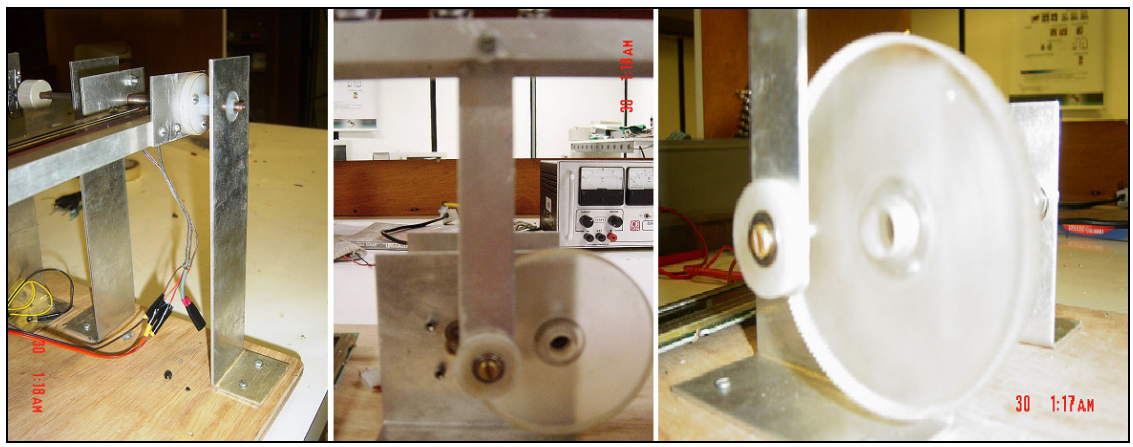

figura 17 : Suporte para pista, manivela e disco da manivela (da esquerda para direita)

Vale observar que o disco da manivela é também uma engrenagem, a fim de que se atingisse uma redução maior para o motor. A redução total atingida, entre motor e disco da manivela, foi de aproximadamente 50:1.

Terminadas essas partes, foram feitas peças auxiliares que iriam completar a bancada. Essas foram: eixo para disco da manivela, suporte para motor e eixo para disco da manivela, eixo para rotação da pista e suportes, base para pista, eixo para rotação da manivela.

A figura 18 ilustra a bancada experimental montada.

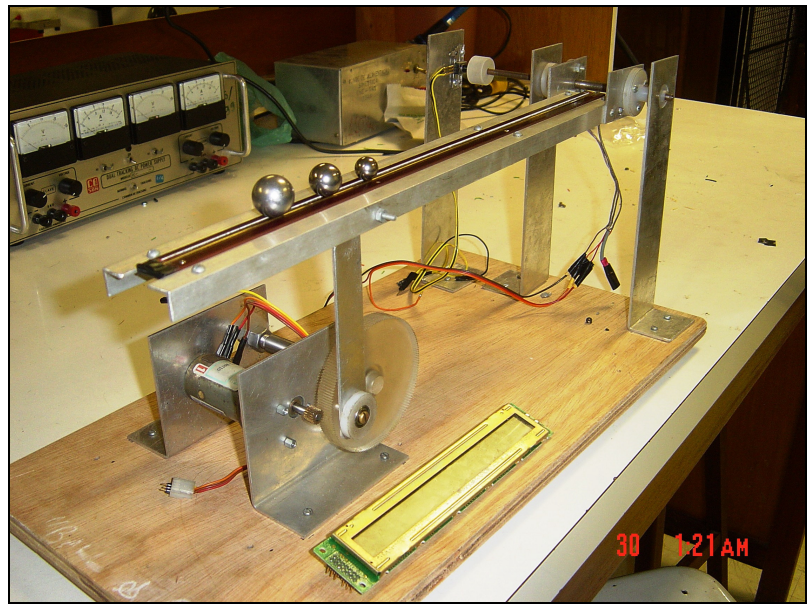

figura 18 : Bancada Experimental

3.2 INSTRUMENTAÇÃO

Realizada a montagem da bancada experimental, foi preciso instrumenta - lá de forma que pudesse ser implementado um sistema de controle. A instrumentação pode ser divida em 2 partes: sensoriamento e acionamento. A seguir é descrita cada uma dessas duas partes (todos os circuitos seguintes foram montados em uma protoboard).

3.2.1 Sensoriamento

O sensoriamento necessário para realizar o controle do sistema exige a medição de duas variáveis: a angulação da pista e a posição da bola sobre a pista.

Para a medição do ângulo, foi necessário acoplar ao eixo de rotação da pista um potenciômetro. Assim, podia-se obter sinais de tensão do potenciômetro proporcionais a angulação da mesma. A figura 19 ilustra como foi feito o acoplamento entre o potenciômetro e o eixo de rotação da pista.

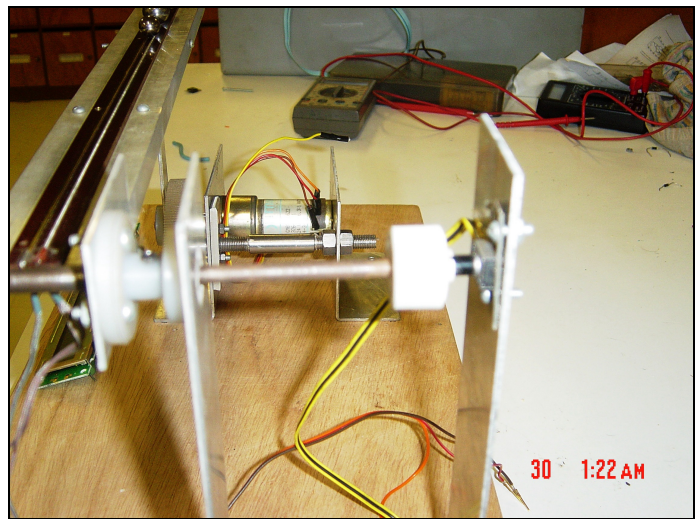

figura 19 : Medição do ângulo da pista

Para a medição da posição da bola, foi utilizada a própria pista como instrumento de medição. Como já citado na descrição da bancada experimental, a pista possui um trilho resistivo de forma que ao utilizar uma bola metálica sobre a bancada, obtinham-se sinais de tensão proporcionais a posição da bola.

Em ambos os sistemas de sensoriamento, foi necessário condicionar os sinais para que o sistema de controle, o microcontrolador PIC, pudesse realizar a aquisição. A seguir é descrito a implementação de tais sistemas de condicionamento para as medições de ângulo e de posição.

3.2.1.1 Medição do Ângulo

Como ilustrado na figura 19, foi utilizado um potenciômetro para realizar as medidas do ângulo da pista. No entanto, como a excursão do ângulo era de aproximadamente ±5° em torno da linha horizontal, a variação da tensão sobre o potenciômetro era muito pequena o que exigiu um condicionamento desse sinal.

A estratégia de condicionamento para realizar a medição foi a seguinte. Inicialmente era aplicada uma tensão de offset ao sistema, para que o intervalo de medição assumisse apenas tensões positivas. Ajustado esse intervalo, foi aplicado um ganho sobre o mesmo para que o sinal assumisse o intervalo de conversão analógico digital do microcontrolador PIC (0 a 5V). Ainda, foi necessário aplicar um filtro de primeira ordem ao sistema para eliminar pequenas variações sobre o sinal de saída, transmitidas para a conversão analógica digital. A figura 20 indica em um sistema de blocos a estratégica de condicionamento descrita.

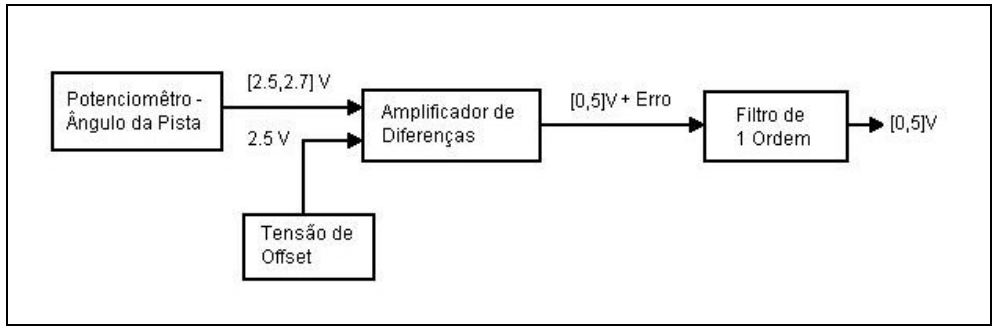

figura 20 : Estratégia de condicionamento do ângulo

A introdução de offset e ganho a medição do ângulo foi feita por meio de um amplificador de diferenças apresentado em [1] e ilustrada na figura 21.

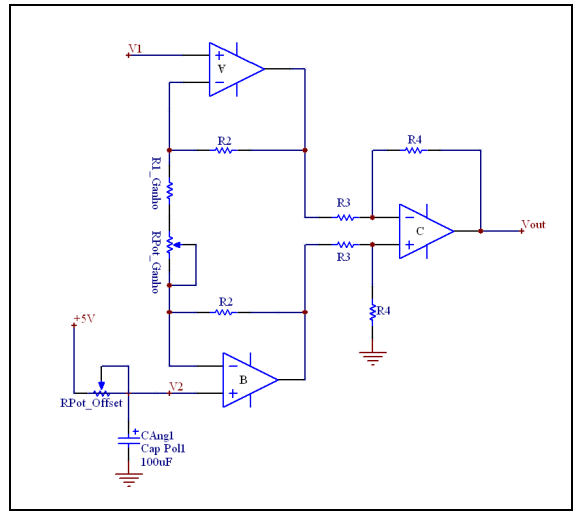

figura 21 : Amplificador de diferenças

A equação da tensão de saída para o amplificador de diferença é dada por:

$$Vout = -\frac{R_4}{R_3}\left(1+\frac{2\cdot R_2}{R_1\_Ganho + RPot\_Ganho}\right)\cdot(V_1 - V_2) \qquad (14)$$

Desse modo, os ajustes de offset e ganho eram feitos manualmente pelos potenciômetros RPot_Offset e RPot_Ganho respectivamente. Assim, independentemente da alimentação usada sobre o potenciômetro de medição do ângulo, e usando $R_3 = R_4$, era possível ajustar o offset e ganho de forma que o sinal de saída variasse sobre o intervalo desejado.

A implementação do filtro passa-baixa de 1ª ordem sobre a saída do amplificador de diferenças está ilustrado na figura 22.

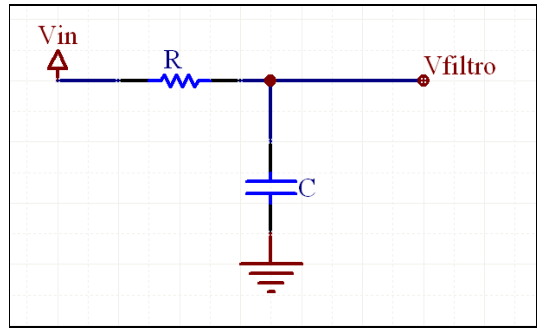

figura 22 : Filtro RC de 1ª ordem

A função de transferência que relaciona *Vin* e *Vfiltro* é dada por:

$$\frac{Vfiltro(s)}{Vin(s)} = \frac{1/RC}{s+1/RC} \qquad (15)$$

O valor da freqüência de corte (*f = 1/(2.pi.RC)*) escolhida para o filtro foi de 100 Hz. Assim, foram escolhidos valores adequados de *R e* C que fornecessem tal freqüência.

Apesar do sistema de condicionamento, as medições ainda não eram ideais já que eram limitadas pela resolução do potenciômetro.

3.2.1.2 Medição da Posição

Assim como foi medida o ângulo da pista, a posição da bola também foi medida por meio de um potenciômetro. No caso, esse potenciômetro consistia na própria pista como ilustrado na figura 12.

Foi verificado por meio de observações sobre um osciloscópio que o sinal fornecido pelo potenciômetro apresentava-se altamente ruidoso com a movimentação da bola sobre o trilho. Em certos momentos, a bola perdia o contato levemente com o trilho gerando picos sobre a medição, sendo esta a principal fonte de ruídos.

Assim, para evitar esse efeito físico gerador de ruídos sobre a medida, foi implementado um filtro butherworthy de 2ª ordem sobre o sinal de saída do potenciômetro. Esse filtro garantia que ao perder o contato com a pista, a tensão de saída se mantivesse constante, sem provocar picos indesejados sobre a medida com o deslocamento da bola. O circuito montado está ilustrado na figura 23 junto com a inclusão da pista como um potenciômetro.

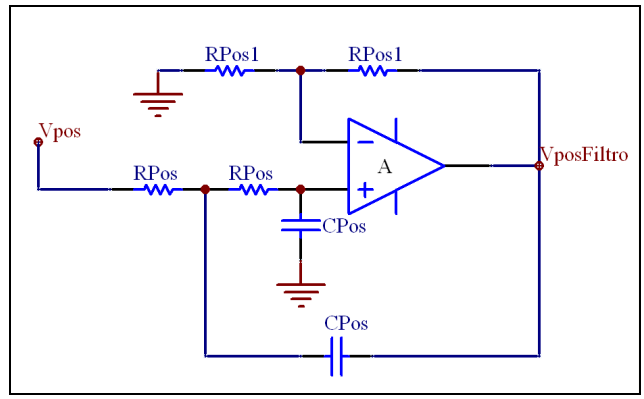

figura 23 : Medição da posição e filtro de 2ª butherworthy

A equação do filtro butherworthy de 2ª é dada por:

$$\frac{\text{VposFiltro}(s)}{\text{Vpos}(s)} = \frac{K \cdot w_c^{\,2}}{(s^2 + s \cdot w_c/Q + w_c^{\,2})} \qquad (16)$$

, onde w_c = 1/RPos.CPos é a freqüência de corte, K = (1+RPos2/RPos1) é o ganho da função transferência e Q = 1/(3 – K) é o fator de qualidade do filtro.

3.2.2 Acionamento

Para realizar o acionamento do motor foram seguidas duas abordagens: acionamento por corrente e acionamento por pwm.

A primeira abordagem, acionamento do motor por corrente, é vantajosa na medida em que a relação torque/corrente é conhecida e assumida linear para o motor DC em questão.

A segunda abordagem, acionamento do motor por pwm, apesar de trabalhar com uma modelagem complexa para o motor, é facilitada na medida em que o próprio microcontrolador possui canais que geram sinais pwm com ciclos de trabalho controlados por software.

Dessa forma, para se trabalhar com maior flexibilidade e mais opções para o sistema, ambas as abordagens foram implementadas e estão descritas a seguir.

3.2.2.1 Acionamento por Corrente

Seguindo a sugestão do prof. Geovany Borges, o circuito montado para que fosse feito o acionamento por corrente está ilustrado na figura 24.

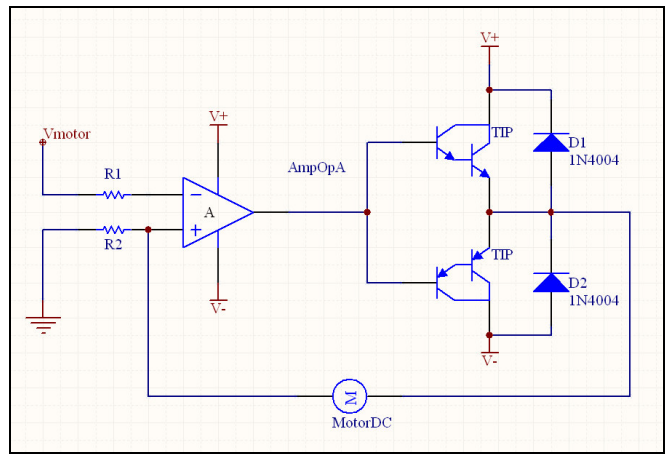

figura 24 : Acionamento do motor por corrente

Por meio desse circuito, a corrente que irá acionar o motor será dada segundo a equação.

$$Imotor = Vmotor \ / R2 \quad (17)$$

Assim, esse circuito trabalha com corrente controlada por tensão por meio da realimentação do amplificador diferencial. Para gerar tal sinal de tensão a ser controlado pelo microcontrolador PIC foi utilizado um conversor digital-analógico. Este conversor trabalha com a conversão de 10 bits para uma faixa de tensão de 0V a 5V. A transmissão dos dados para conversão é feita utilizando a interface SPI, suportada pelo microcontrolador PIC. Dessa forma, foi possível gerar tensões na faixa de 0V a 5V, dada uma entrada de 0 a 1023, que é a faixa de trabalho do conversor digital analógico.

No entanto, para acionar o motor em ambos os sentidos de rotação, era necessário gerar correntes negativas, ou seja, tensões negativas. Desse modo, foi implementado um amplificador de diferenças para aplicar uma tensão de offset ao sinal do conversor digital analógico e um ganho que ajustasse a faixa de operação de corrente sobre o motor, assim como foi aplicado à medição do ângulo da pista.

Para observar a faixa de operação de corrente foi feito o seguinte procedimento. Usando uma alimentação de ±15V para o circuito de acionamento, foi aumentada a tensão de entrada sobre o circuito até que a corrente atingisse um valor máximo, que foi de 0.2 A com a saturação do amplificador diferencial. Então, o ganho do amplificador de diferenças foi ajustado de forma que as tensões na faixa de 0V a 5V geradas pelo conversor digital analógico trabalhassem na faixa de ± (0.2 A)*(15 Ohms). = ± 3.0 Volts.

O diagrama de blocos que resume o esquema de acionamento por corrente do motor está ilustrado na figura 25.

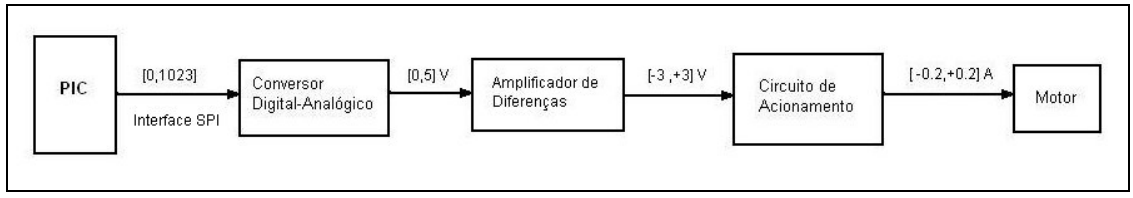

figura 25 : Esquema de acionamento por corrente

3.2.2.2 Acionamento por PWM

O acionamento por PWM do motor foi feito utilizando a ponte H incorporada no CI L298. Para acionar o motor em ambos os sentidos, foram usados os dois canais de PWM do microcontrolador PIC, de acordo com a figura 26.

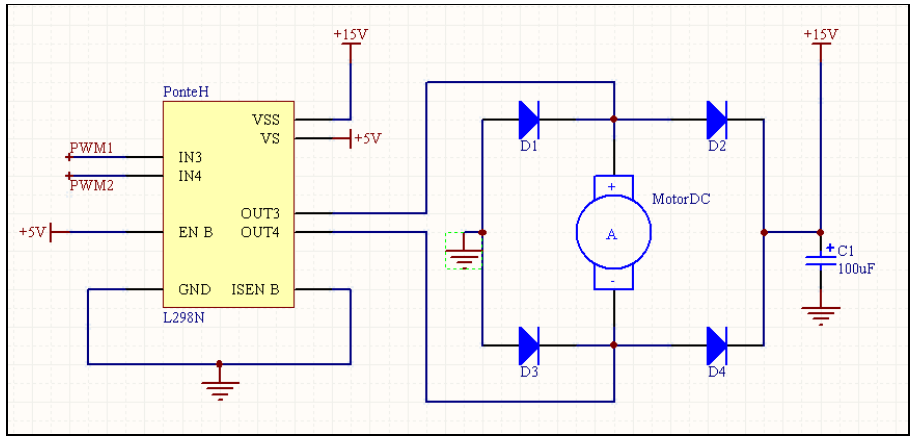

figura 26 : Acionamento por PWM

A freqüência e o ciclo de trabalho de trabalho do canal PWM1 e PWM2 eram ajustados por software pelo microcontrolador PIC. Na implementação, o valor do PWM para os dois canais foi ajustado para aproximadamente 40KHz (visualizado sobre um osciloscópio).

3.3 ABORDAGEM PARA CONTROLE

Antes de iniciar o projeto de um controlador, deve ser abordado a forma como o controlador será incluído no sistema. Até o momento, a malha do sistema está representada na figura 27.

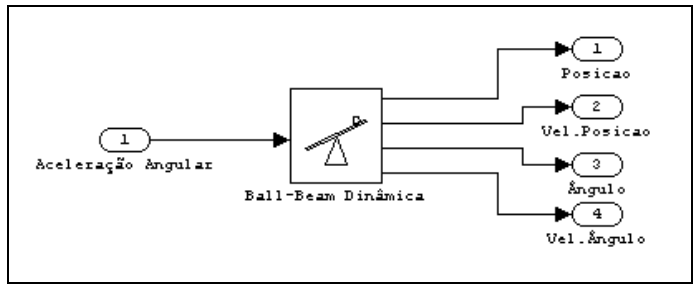

figura 27 : Malha aberta do sistema

Em [11], o projeto de controle é abordado usando linearizações locais. No caso, são determinados alguns pontos de operação para o sistema, e para cada um desses pontos um controlador de realimentação de estados é projeto, pois sobre estas condições locais o sistema é aproximadamente linear. Um controlador fuzzy atua então sobre o sistema julgando em quais pontos linearizados o sistema se encontra. A figura 28 ilustra como a base de regras desse controlador fuzzy é composta.

R^1: IF x_1 is about 0, x_3 is about 0, x_4 is about 0, THEN $\dot{x} = A_1x + b_1u$

R^2: IF x_1 is about 0, x_3 is about $\pm \pi/4$, x_4 is about 0, THEN $\dot{x} = A_2x + b_2u$

R^3: IF x_1 is about 3, x_3 is about $\pm \pi/4$, x_4 is about ± 0.4, THEN $\dot{x} = A_3x + b_3u$

R^4: IF x_1 is about -3, x_3 is about $\pm \pi/4$, x_4 is about ± 0.4, THEN $\dot{x} = A_4x + b_4u$

R^5: IF x_1 is about 3, x_3 is about 0, x_4 is about ± 0.4, THEN $\dot{x} = A_5x + b_5u$

R^6: IF x_1 is about -3, x_3 is about 0, x_4 is about ± 0.4, THEN $\dot{x} = A_6x + b_6u$.

figura 28 : Controlador fuzzy linearizações locais

Já em [12] é feita uma abordagem diferente. O sistema de controle é dividido em duas malhas, como mostra a figura 29. Essa forma é conhecida como controle em cascata. Nesse sistema, um controlador PID foi introduzido na malha interna para controlar a angulação da pista enquanto o controlador fuzzy, posicionado sobre a malha externa do sistema, controla a posição da bola sobre a pista.

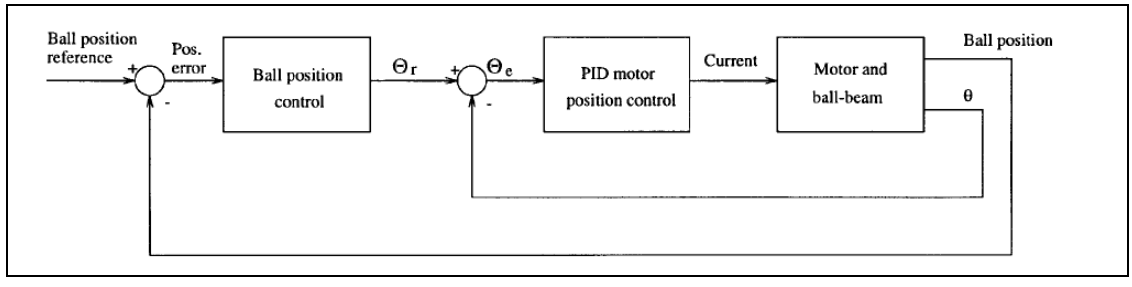

figura 29 : Controle em cascata

Para a presente implementação, foi decidido atuar sobre o sistema de forma mais direta, para observar o comportamento do controlador fuzzy atuando independente de outros controladores. Então, o sistema de controle que foi estudado está representado na figura 30, onde o controlador fuzzy recebe como entrada as variáveis que intuitivamente contém informações necessárias para controlar o sistema.

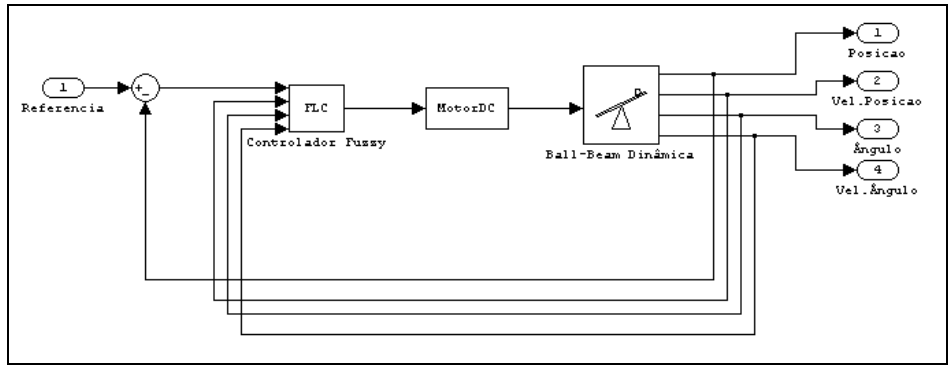

figura 30 : Malha de controle a ser estudada

3.4 CONTROLADOR FUZZY

O projeto do controlador fuzzy, para a abordagem escolhida, foi feito de forma intuitiva. Para tanto, foram seguidos os passos descritos na seção 2.1.2. Durante esse processo, foi lembrado que a implementação do controlador seria feita em um microcontrolador PIC com linguagem de programação em alto nível.

1º Passo : O primeiro passo para sua construção foi a identificação das variáveis do sistema. No caso, observando a figura 30, as variáveis de entrada para o controlador fuzzy são o erro da posição da bola, a velocidade da bola (ou apenas variação do erro), o ângulo da pista e a velocidade do ângulo (ou apenas a variação do ângulo), enquanto a variável de saída é uma tensão ou corrente para acionamento do motor.

Nesse momento, foi necessário caracterizar as variáveis, ou seja, atribuir termos fuzzy, atribuir significado, a cada uma delas. Essa etapa influi bastante na construção do controlador, pois é ela que introduz o raciocínio intuitivo (ou especialista) ao controlador. Ainda, o número de termos

atribuídos as variáveis definem a forma como é controlado o sistema, já que define o número de regras pela combinação de todos os termos.

Considerando esses aspectos, decidiu-se a príncipio utilizar os termos POSITIVO (**POS**), NEGATIVO (**NEG**) e ZERO (**ZE**) para as variáveis *erro da posição da bola*, *velocidade da bola*, *ângulo da pista* e *velocidade angular da pista*. Com o uso desses termos, obtém-se um total de 3x3x3x3 = 81 situações possíveis para o sistema, ou seja, 81 regras para o controlador fuzzy. Apesar do número elevado de regras, a quantidade de 3 termos torna mais específica cada situação o que facilita identificar qual ação tomar para cada regra.

Com relação às ações do controlador, estas também devem ser caracterizadas de maneira fuzzy. Lembrando que a ação do controlador é uma força que irá acionar o motor de forma que a pista se incline para positivamente ou negativamente, os termos atribuídos foram: Força – NEGATIVA ALTA (**NA**), NEGATIVA MEDIA ALTA (**NMA**), NEGATIVA MEDIA BAIXA (**NMB**), NEGATIVA BAIXA (**NB**), ZERO (**ZE**), POSITIVA BAIXA (**PB**), POSITIVA MEDIA BAIXA (**PMB**), POSITIVA MEDIA ALTA (**PMA**), POSITIVA ALTA (**PA**).

Assim, procurou-se obter uma forma organizada de avaliar as situações geradas pelas combinações dos termos citados. Dessa avaliação, foram criadas as regras para o controlador fuzzy a partir do relacionamento entre cada situação e a correspondente ação necessária. A figura 31 ilustra a forma como foi organizadas e definidas as 81 regras para o controlador fuzzy utilizando 3 termos para cada variável.

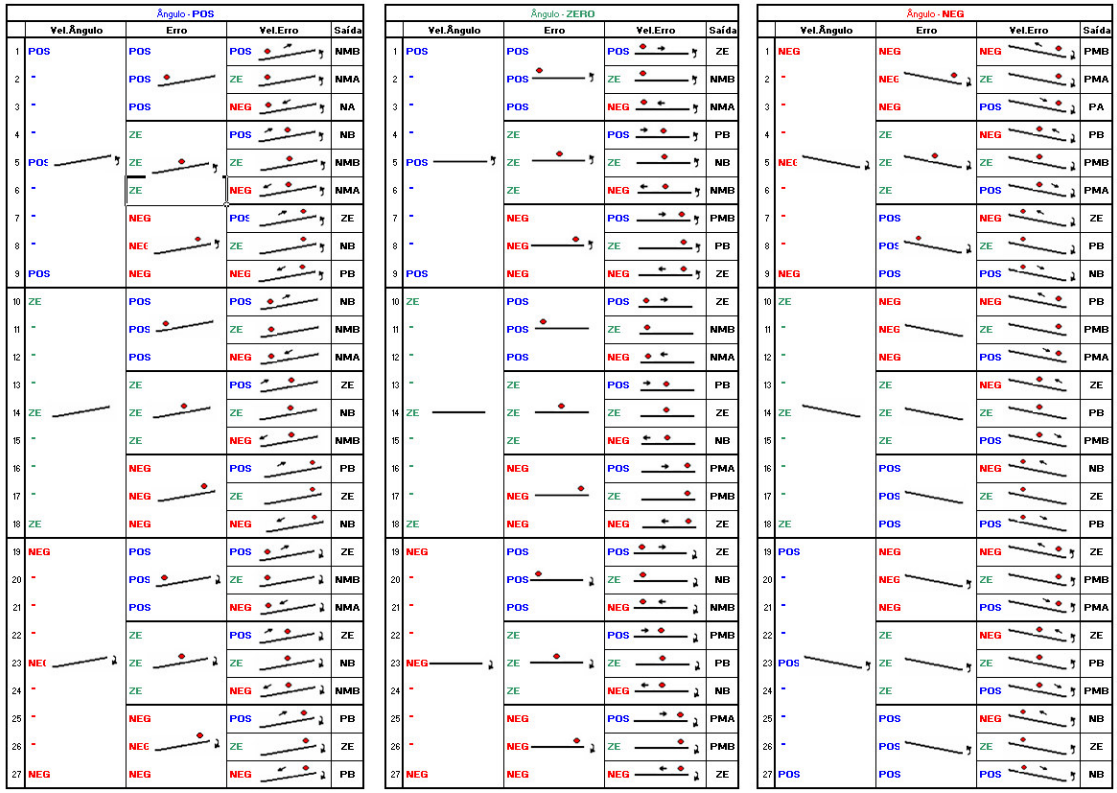

figura 31 : Base de raciocínio para controlador fuzzy com 81 regras

A interpretação da figura 31 é feita da seguinte forma. Tomando a linha 1 como exemplo tem-se que a regra definida é dada por:

SE **Ângulo é POS *E*** **Vel.Ângulo é POS *E*** **Erro é POS *E*** **Vel.Erro é POS ,**

ENTÃO **força é NMB**

A partir dessa base de raciocínio foram definidas as regras para um controlador fuzzy mais simples com 16 regras. Essa base de regras foi feita excluindo as regras que incluíam algum termo **ZE** em sua composição. Assim, a representação da base de regras para um controlador fuzzy com 2 termos por variável (**POS** e **NEG**) está ilustrado na .

Ângulo - POS				
	Vel.Ângulo	Erro	Vel.Erro	Saída
1	POS	POS	POS	NMB
2	''	POS	NEG	NA
3	''	NEG	POS	ZE
4	POS	NEG	NEG	PB
5	NEG	POS	POS	ZE
6	''	POS	NEG	NMA
7	''	NEG	POS	PB
8	NEG	NEG	NEG	PB

Ângulo - NEG				
	Vel.Ângulo	Erro	Vel.Erro	Saída
9	NEG	NEG	NEG	PMB
10	''	NEG	POS	PA
11	''	POS	NEG	ZE
12	NEG	POS	POS	NB
13	POS	NEG	NEG	ZE
14	''	NEG	POS	PMA
15	''	POS	NEG	NB
16	POS	POS	POS	NB

Figura 32 : Base de regras para controlador fuzzy com 16 regras

2º Passo: Outra escolha a ser feita com relação as variáveis é a representação de seus termos, ou seja, quais funções de pertinência serão usadas para as variáveis. O formato das funções de pertinência deve levar em consideração o processo de fuzzificação e defuzzificação a ser usado, pois a partir do momento que curvas mais complexas são definidas, maior o esforço computacional demandado pelo controlador.

Assim, as funções de pertinência para o controlador fuzzy foram definidas por funções triangulares para as variáveis de entrada. Para as variáveis de saída, os termos fuzzy foram definidos por números escalares (singletons), o que irá facilitar o método de defuzzificação adiante. A figura 33 e figura 34 ilustram a representação das funções de pertinência.

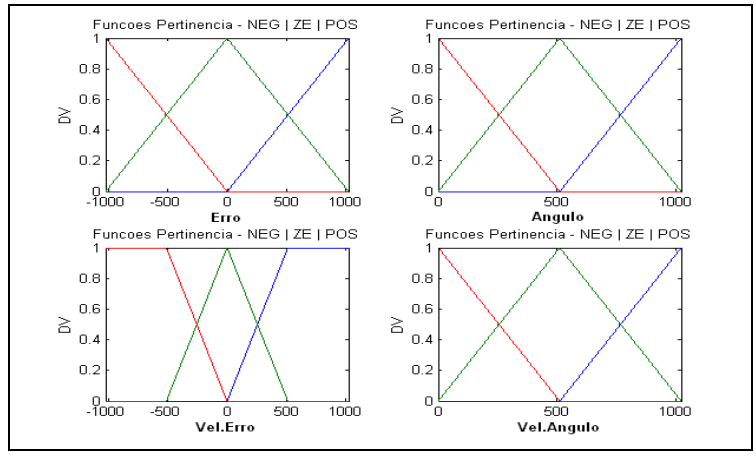

figura 33 : Funções de pertinência – termos POSITIVO, ZERO e NEGATIVO

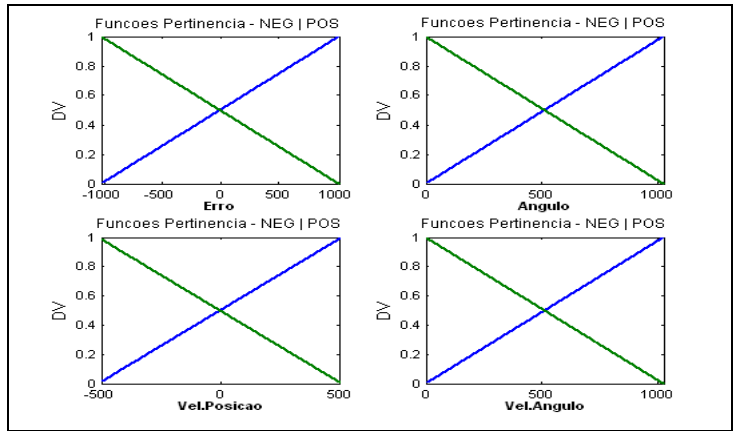

figura 34 : Funções de pertinência – termos POSITIVO e NEGATIVO

O universo de discurso das funções de pertinência está definido sobre o intervalo [-1024,1024] sendo este o intervalo de conversão Analógico/Digital que o microcontrolador PIC trabalha.

Com relação ao processo de fuzzificação, é escolhida a fuzzificação singleton, adotada em todas as aplicações citadas em [passino], que irá simplificar o processo de inferência a ser definido.

3º Passo : A definição do processo do inferência foi feita escolhendo o produto e soma como operações t-norma e s-norma respectivamente. Ainda, foi escolhida a implicação Mamdani produto e o modo de combinação das regras foi a inferência baseada em regra individuais com combinação união (vide-anexo para as definições).

Está máquina de inferência é bastante compatível com um ambiente de programação, considerando ainda que o modo de combinação de regras e as escolhas das operações t-norma e s-norma podem ser feitas de forma simples algebricamente.

Tal máquina de inferência foi definida tanto para o controlador fuzzy com 16 regras como para o controlador fuzzy com 81 regras.

4º Passo: Seguindo a escolha de singletons para representar as funções de pertinência de saída, e ainda considerando as escolhas de operação soma como operações t-norma e a combinação união para a máquina de inferência, o processo de defuzzificação escolhido foi Centro de Gravidade de

Singletons (COGS). Dessa forma, a saída do controlador fuzzy passa a ser uma média ponderada para das implicações das regras.

Ainda, a combinação desse método de defuzzificação com a escolha das funções de pertinência triangulares para o controlador fuzzy com 16 regras possui o denominador de () bem definido. De fato, tal denominador é um valor constante unitário, o que evita um armazenamento dos valores de implicações das regras em um ambiente de programação.

Esse processo de defuzzificação foi adotado tanto para o controlador fuzzy de 16 regras com para o controlador fuzzy com 81 regras.

Definidos então todos os processos do controlador, foram obtidas as superfícies de controle para os ganhos de escalonamento unitários, mostrados na figura 35 e na figura 36.

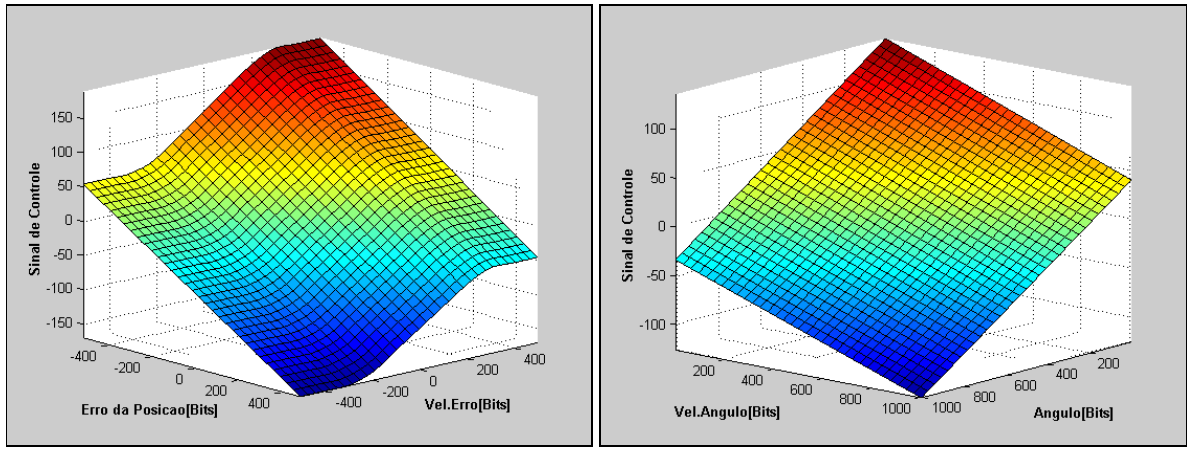

figura 35 : Superfícies de controle para controlador fuzzy com 16 regras e ganhos de escala unitários

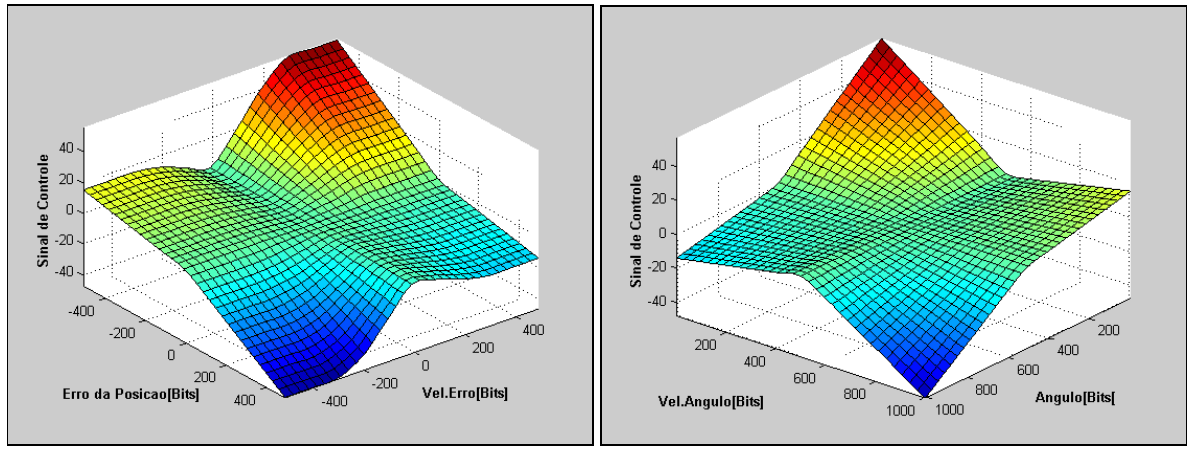

figura 36 : Superfícies de controle para controlador fuzzy com 81 regras e ganhos unitários

Comparando as superfícies entre os controladores, pode-se notar que o controlador fuzzy com 81 regras possui um comportamento mais suave em torno da posição de equiíbrio, enquanto o controlador com 16 regras possui uma inclinação mais acentuada neste ponto.

3.5 SIMULAÇÃO

No momento em que um sistema físico pode ser representado por equações matemáticas, é possível simular seu comportamento dinâmico em um ambiente computacional. Em diversas situações, tais simulações representam importante papel no projeto de controladores, pois permite testar parâmetros de projeto e obter resultados rapidamente.

Nesse sentido, a partir das equações dinâmicas do sistema barra-esfera, foi tomada como primeira etapa do projeto do controlador fuzzy a implementação da simulação do sistema. Além dos aspectos positivos citados anteriormente, a simulação permitiu implementar de forma simplificada, porém com semântica semelhante, o controlador fuzzy a ser embarcado no microcontrolador que irá controlar o sistema real.

Deve-se lembrar que a simulação apenas representa de forma aproximada o comportamento do sistema dinâmico devido a simplificações e considerações normalmente associadas ao modelamento matemático.

Para realizar a simulação, certos aspectos foram considerados. No caso, tais aspectos foram a escolha do modelo do sistema (aproximação linear ou sistema não linear), a inclusão do motor na malha de controle, modelo dinâmico do motor utilizado e as variáveis usadas pelo controlador fuzzy. Para todos os casos foram utilizados os controladores fuzzy de 16 e 81 regras definidos no seção 3.4.

Assim, de acordo com essas características foram implementadas as seguintes simulações: **simulação 1** - utilização de um modelo linearizado sem inclusão do motor; **simulação 2** - utilização do modelo não linear sem inclusão do motor na malha de controle; **simulação 3** - inclusão do motor na malha de controle e nova representação para variáveis dos controladores fuzzy; **simulação 4** - inclusão de ruídos na simulação e utilização de filtragem digital. A partir da **simulação 3**, foi feita uma animação gráfica do sistema para melhor visualizar os resultados da simulação e sintonizar o controlador fuzzy.

A seguir serão discutidos os pontos observados em cada uma das simulações citadas. Os resultados das simulações serão descritos na posteriormente.

Simulação 1: Utilização do modelo linearizado sem inclusão do motor na malha de controle:

Inicialmente, a malha de controle utilizada para a simulação está representada na figura 37.

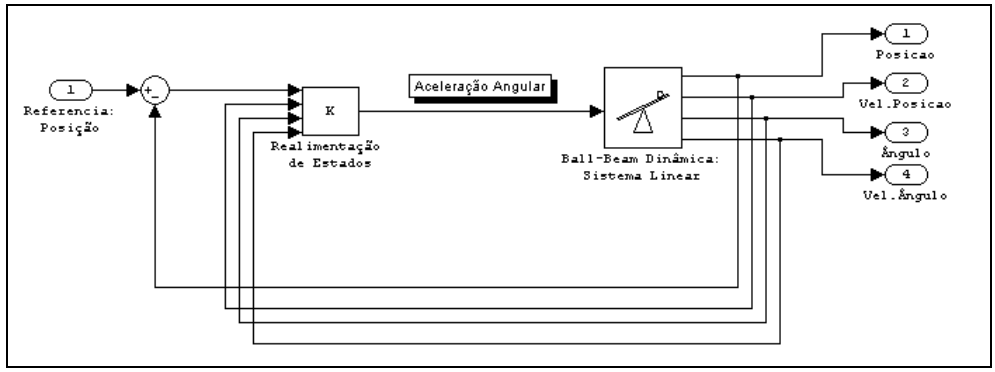

figura 37 : Malha de controle inicial

O objetivo inicial dessa abordagem era construir um modo de simular o sistema gerando resultados confiáveis e usar os resultados para construir o controlador fuzzy. Para tanto foi utilizada a malha

ilustrada na figura 37, que poderia ser comparada com uma implementação demonstrada em [15], cujos resultados obtidos eram baseados na mesma malha de controle, porém utilizava funções internas do matlab para implementar a simulação. A representação esquemática de tal simulação está apresentada na figura 38.

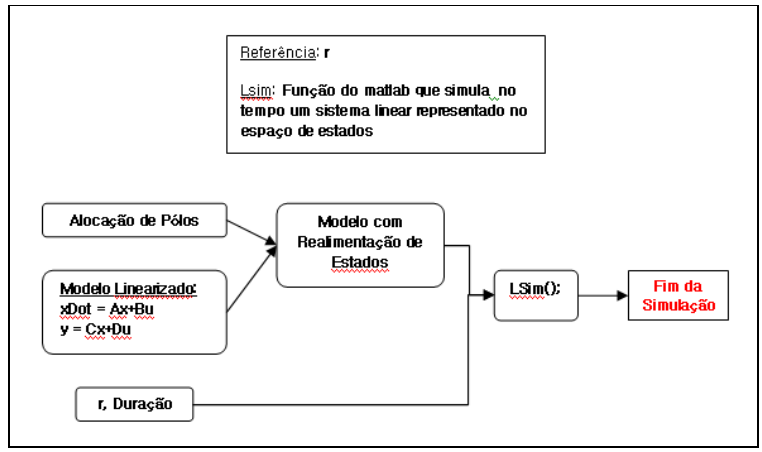

figura 38 : Referência para simulação – Universidade Carnegie Mellon

Assim, foi realizado o modo para simular o sistema usando os mesmos parâmetros adotados por [15], porém sem utilizar as funções internas do matlab para sua execução. Assim, as respostas de ambas as simulações puderam ser comparadas. O modo adotado para a simulação está representado pelo fluxograma ilustrado na figura 39.

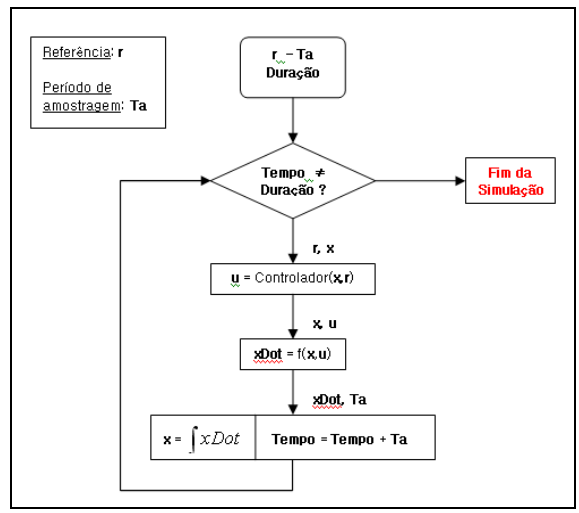

figura 39 : Metodologia de simulação

A partir dessa abordagem inicial, foi possível validar a metodologia de simulação, ambientar com o ambiente de programação do matlab e obter resultados que iriam compor a base inicial para a construção do controlador fuzzy.

Aproveitando os resultados da simulação, foi possível identificar os limites de excursão para as variáveis correspondentes as velocidades de posição e ângulo. Tais limites foram importantes para a simulação da conversão Analógica/Digital das medidas a seguir.

Os arquivos utilizados na realização dessa simulação, que estão incluídos em [17] foram: **rscale.m** e **SimulacaoEspEstados.m** (arquivo que chama a simulação).

Simulação 2: Utilização do modelo não-linear sem inclusão do motor na malha de controle:

Considerando os resultados obtidos pela **simulação 1**, partiu-se para a implementação inicial do controlador fuzzy em matlab, de forma que sua estrutura pudesse ser implementada sobre o microcontrolador pic. Ainda, definido um método para executar a simulação (figura 39) foi possível considerar as não-linearidades do sistema na simulação e incluir o controlador fuzzy na malha de controle.

Observa-se pela figura 37 que o controlador por realimentação de estados utiliza como variáveis de entrada as variáveis de estado obtidas diretamente do laço de simulação. Apesar de a abordagem ser correta, ela não representa adequadamente a situação real que os controladores fuzzy irão encontrar. No caso, as variáveis de entrada deveriam estar escalonadas de acordo com a faixa de valores do conversor Analógico/Digital usado, pois será com este universo de valores que o controlador fuzzy irá trabalhar. Para tanto, foi simulada uma conversão analógica digital sobre as variáveis de estado que serviriam como entrada para os controladores fuzzy, de forma que as medidas de eram escalonadas sobre a faixa de valores de [-1023 , 1023].

Assim, a nova malha de controle está ilustrada na figura 40.

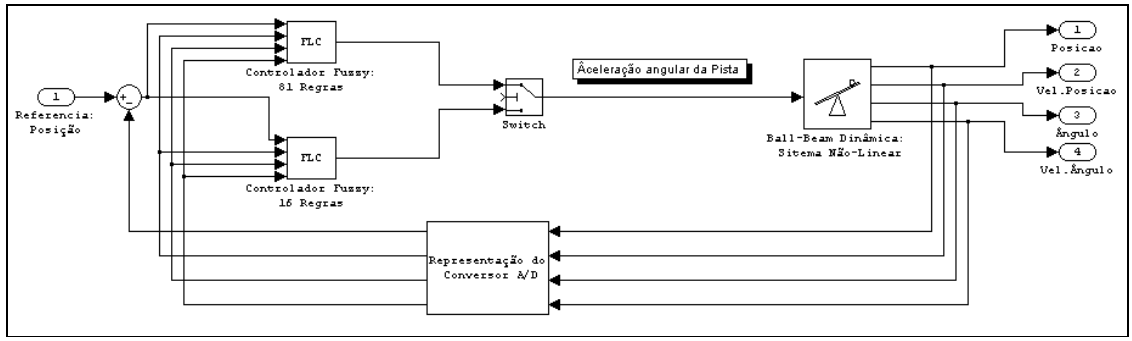

figura 40 : Simulação do sistema não-linear

A partir desse ponto, foi possível gerar resultados baseados em 2 controladores, o controlador fuzzy com 81 regras e o controlador fuzzy com 16 regras.

Os arquivos utilizados na realização dessa simulação, que estão incluídos em [17] foram: **SimCF16_SemMotor.m** (arquivo que chama a simulação); **CF16_Declaracao.m, FuncoesPertinencia16_SM.m, CF16_PreProFuzzificacaoSM.m, CF16_InfeDefuzzificacaoSM.m, CF81_Declaracao.m, FuncoesPertinencia81_SM.m, CF81_PreProFuzzificacaoSM.m e CF81_InfeDefuzzificacaoSM.m** (arquivos que representam as ações do controlador fuzzy).

Simulação 3: Inclusão do motor sobre a malha de controle e mudança nas variáveis do controlador fuzzy

Foi percebido na **simulação 2** que deveria ser feito uma mudança na forma como eram utilizadas as variáveis de entrada do controlador fuzzy. Para tanto, foi simulada uma conversão analógica digital sobre as variáveis de estado que serviriam como entrada para os controladores fuzzy, de modo que as medidas de posição e ângulo eram escalonadas sobre a faixa de valores de [-1023 , 1023].

Além dessas mudanças foi adotada outra modificação com relação ao uso das variáveis de velocidade da posição e ângulo do sistema. Tais variáveis não são medidas diretamente do sistema como sugerido na figura 40, tendo estas que serem calculadas por meio de um processo de derivação, utilizando o período de amostragem.

Para evitar o processo de derivação, que pode ser considerado uma fonte de amplificação de ruídos sobre a medição, foi utilizada a variação das medidas de posição e de ângulo. Assim, as velocidades eram representadas apenas pela diferença entre os valores de posição em dois instantes distintos.

Assim, com essa nova representação das variáveis, conseguiu-se preparar o controlador fuzzy para receber os valores com os quais iria trabalhar na implementação física do sistema.

No entanto, ainda restava incluir o motor sobre a simulação, já que nas abordagens anteriores a ação dos controladores era representada por uma aceleração introduzida diretamente ao sistema. A inclusão do motor, a partir de um modelamento prévio do mesmo (9), permite considerar sobre o sistema a dinâmica de acionamento e a saturação do sinal de controle, que para um motor DC pode ser representada pelos limites de tensão de alimentação.

Ao incluir o motor na simulação, a forma como é transmitido o movimento fisicamente ao sistema deve ser considerada, sendo necessário obter matematicamente a descrição desse movimento. De acordo com a figura 14, a forma de transmissão do movimento era feita por meio de uma manivela acoplada tanto ao disco de engrenagem conectada ao motor quanto à pista. As equações (11), (12) e (13) relacionam a angulação da pista com a angulação do disco da manivela, no entanto são relativamente complexas e suas variáveis de entrada são posições tomadas sobre um eixo cartesiano colocado sobre o sistema como indica a figura 15, o que não seria simples de incluir na simulação.

A forma encontrada para descrever a transmissão do movimento ao sistema se baseia no resultado obtido das equações (11), (12) e (13). Lembrando que o resultado das equações com base na figura 16 formam uma curva relacionando ângulo do disco da manivela e ângulo da pista, foi possível obter uma equação que aproximava essa curva usando o matlab. Dessa maneira a relação ângulo do disco da manivela vs ângulo da pista pode ser representada pela equação polinomial (17). A figura 41 ilustra a curva obtida pelas equações, a aproximação feita e o erro resultante da mesma.

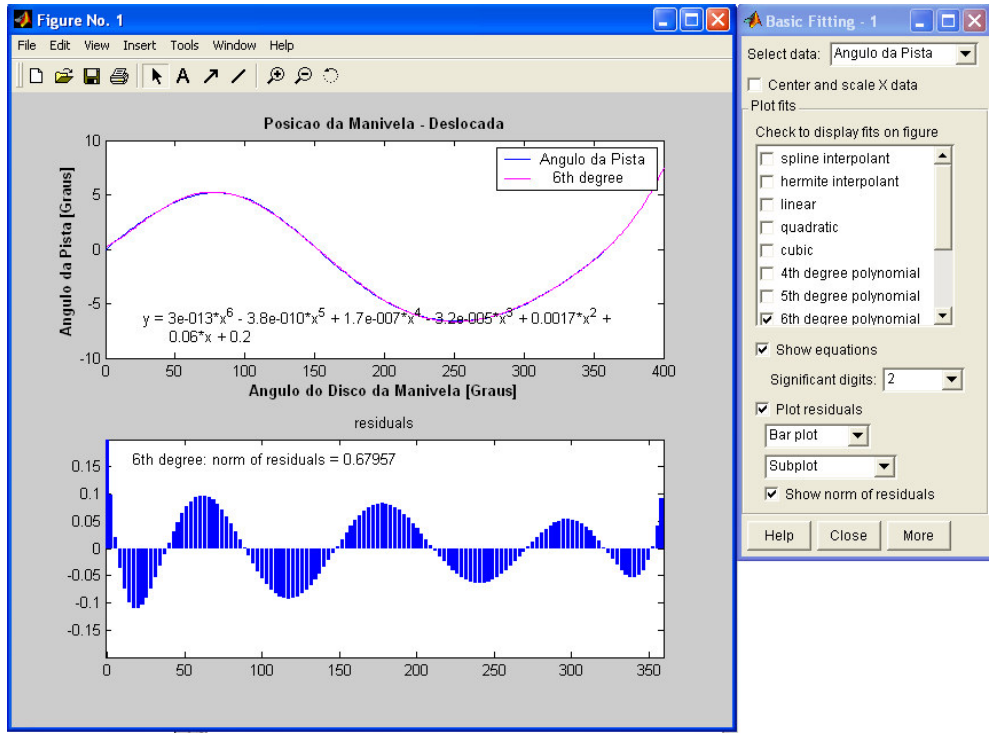

figura 41 : Aproximação para transmissão do movimento do sistema

$$y = 3e-013 \cdot x^6 - 3.8e-010 \cdot x^5 + 1.7e-007 \cdot x^4 - 3.2e-005 \cdot x^3 + 0.0017 \cdot x^2 + 0.06 \cdot x + 0.2 \quad (17)$$

, onde y é o valor do ângulo da pista e x é o valor do ângulo do disco da manivela.

Dessa forma, a partir da equação (9) é possível calcular os valores de x, e em seguida calcular os valores resultantes de y.

Assim, com a modificação nas variáveis de entrada do sistema e a inclusão do motor sobre a simulação, a nova malha de controle para o sistema está representada pela figura 42. A mudança nas variáveis de entrada é indicada por meio do bloco derivativo sobre as medições de posição e ângulo, ao invés da realimentação direta da velocidade da posição e da velocidade do ângulo.

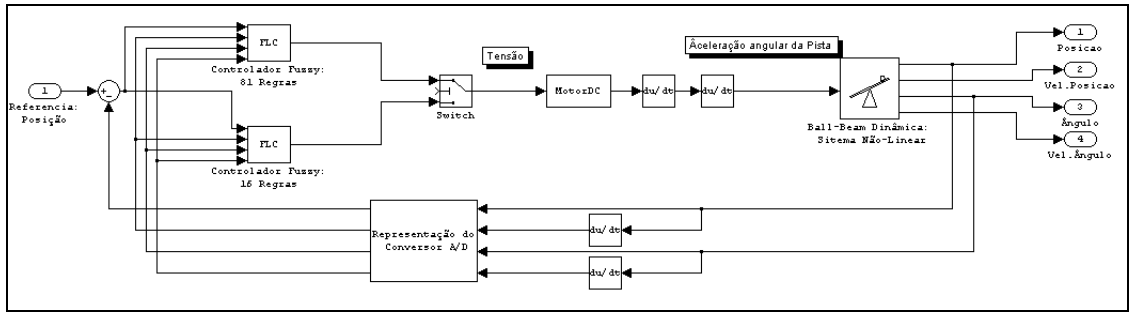

figura 42 : Malha de controle com inclusão do motor

Para avaliar de maneira mais intuitiva os resultados da simulação, foi implementada uma animação gráfica do sistema.

Os arquivos utilizados para a simulação 3, e também para a simulação 4, incluídos em [17], foram: **Simulação3e4_Barra_Esfera.m** (arquivo que executa a simulação e contém todos os seus parâmetros); **CF16_Declaracao.m, FuncoesPertinencia16PrePro.m, CF16_PreProFuzzificacao.m, CF16_InfeDefuzzificacao.m, SuperficieDeControleCF16.m, CF81_Declaracao.m, FuncoesPertinencia81PrePro.m, CF81_PreProFuzzificacao.m, CF81_InfeDefuzzificacaoSM.m e SuperficieDeControleCF16.m,** (arquivos que representam as ações do controlador fuzzy).

Simulação 4: Inclusão de ruídos na simulação.

Para tornar a simulação mais próxima do que seria encontrado na implementação foi criada uma maneira de incluir ruído sobre as medidas de posição e ângulo do sistema. O ruído foi inserido somente sobre estas variáveis, pois são as únicas realmente medidas pelo sistema, e durante a simulação, assim como para o sistema real, esse ruído será consequentemente propagado para as variáveis de velocidade.

Posteriormente, houve a necessidade de incluir no sistema um filtro digital sobre a ação de controle, devido aos ruídos na medição, e um canal integral sobre a variável erro da posição, para evitar erros de regime que constaram nos resultados.

O filtro implementado foi um filtro digital de 1ª ordem, discretizado, como mostra a equação (18) onde *a* é a freqüência de corte do filtro em questão, *T* é o período de amostragem, *u(k)* é o sinal de controle filtrado e *e(k)* é o sinal de controle de entrada para o filtro.

$$u(k) = \frac{u(k-1)}{(1+T \cdot a)} + \frac{e(k) \cdot T \cdot a}{(1+T \cdot a)} \tag{18}$$

O canal integral incluído no sistema é dado pela equação (19), onde T_i é o ganho de integração, *T* é o período de amostragem e *uIntegral(k)* é o sinal de controle do integrador.

$$uIntegral(k) = uIntegral(k-1) - erro(k) \cdot Ti \cdot T \tag{19}$$

3.6 IMPLEMENTAÇÃO

A implementação do controlador fuzzy sobre a bancada barra-esfera montada foi feita sobre o microcontrolador PIC18f252. Esse microcontrolador possui uma linguagem de programação em alto nível e apresenta as características exigidas pelo sistema de instrumentação e de processamento do controlador fuzzy.

No caso, o microncontrolador PIC possui canais conversores Analógicos/Digitais, módulos PWM e uma interface RS232 para comunicação com o computador.

Desse modo, a implementação experimental do sistema era realizada conforme mostra a figura 43.

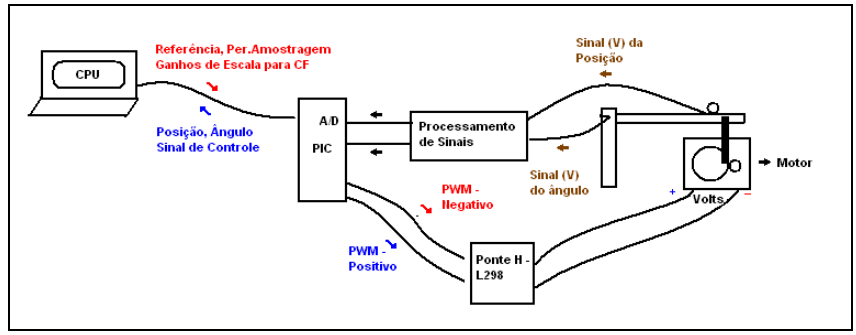

figura 43 : Diagrama da experimentação do sistema

Deve-se citar aqui que os circuitos de condicionamento de sinais foram montados sobre uma protoboard.

Como indica a figura 43, o experimento era conduzido da seguinte forma: por meio do computador era fornecido o período de amostragem da simulação e os valores dos ganhos de escala para as variáveis de entrada e saída. Durante sua execução, os valores dos dados eram enviados para o computador e armazenados para uma análise posterior. Um display LCD era também usado para melhor visualização e entendimento da experimentação.

Ainda, outros parâmetros que facilitavam a experimentação eram também passados como argumento o microcontrolador PIC. Tais parâmetros eram flags que ligavam ou desligavam a ação do filtro digital e do canal integral. O programa que foi utilizado para realizar a comunicação serial entre o computar e o PIC foi o Matlab.

Apesar de ser feito o projeto e simulado resultados para os controladores fuzzy com 16 e 81 regras, somente o controlador de 16 regras foi implementado sobre o PIC inicialmente devido à semelhança do resultado para as simulações entre ambos os controladores e a maior simplicidade computacional do controlador com 16 regras. Após a implementação do controlador fuzzy com 16 regras, verificou-se que o microcontrolador não possuía memória suficiente para comportar as 81 regras do outro controlador fuzzy.

Para a execução experimental do controle sobre a bancada foram usados os ganhos de sintonização do controlador encontrados nas simulações.

4. RESULTADOS

Nesse capítulo serão apresentados os resultados obtidos nas simulações e os resultados da implementação do controlador fuzzy sobre a bancada experimental.

4.1 Resultados sobre Simulações

Como apresentado na seção 3.5, as simulações executadas foram divididas em quatro etapas. Assim, os resultados destas etapas estão mostrados nessa mesma seqüência.

Resultado – Simulação 1:

Como resultado para a **simulação 1** tem-se os gráficos ilustrados na figura 44. Esses gráficos representam a curva de resposta, dada a referência um degrau de 0.20m, para a simulação demonstrada por [15] (gráfico a esquerda) e para a metodologia de simulação adotada (gráfico a direita) de acordo com a figura 39.

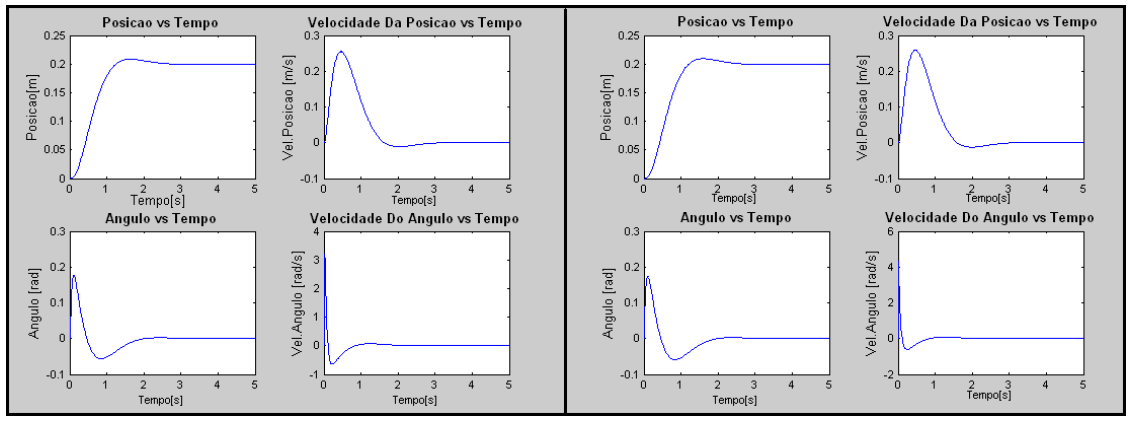

figura 44 : Simulação Carnegie Mellon vs Metodologia de Simulação

Deve-se observar que ao utilizar os mesmos parâmetros para ambas as simulações os resultados são bastante próximos, o que validou a metodologia de simulação implementada.

Com o método para simular o sistema definido, foram identificados os limites de excursão para as velocidades consideradas no sistema.

A identificação dos limites da velocidade da posição da bola foi feita da seguinte maneira: a condição inicial do sistema foi ajustada de modo que a bola estivesse colocada sobre a extremidade positiva da pista e então foi desativado o sinal de controle sobre o sistema. Com isso, a velocidade máxima atingida pela bola foi registrada no momento em que a bola passasse pelo limite oposto da pista. A figura 45 ilustra a situação descrita anteriormente.

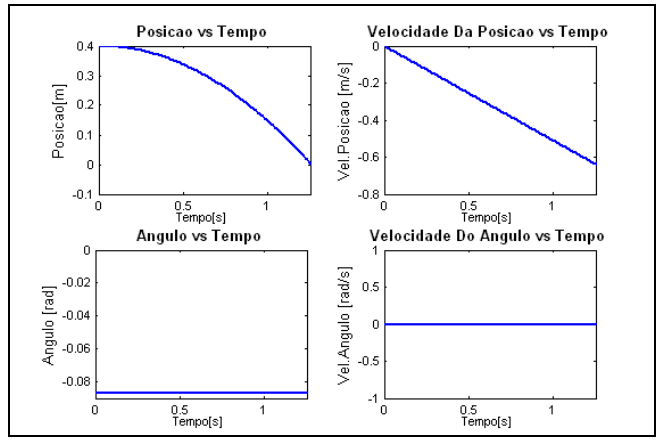

figura 45 : Condução da bola para máxima velocidade

A identificação da velocidade do ângulo foi feito observando os resultados da figura 45. Como nessa simulação não foi considerada nenhuma saturação do controlador, supostamente não há limites para tal variável.

Assim, as escalas de conversão Analógica/Digital para as variáveis foram definidas conforme mostra a figura 46.

Variáveis	Limites de excursão	Escala Digital
Posição	[0 , 0.40] m	[0 , 1023]
Vel.Posição	[-0.64 , +0.64] m/s	[0 , 1023]
Ângulo	[-0.087 , + 0.087] rad	[0 , 1023]
Vel.Ângulo	[-5 , +5] rad/s	[0 , 1023]

figura 46 : Escalas de conversão Analógica/Digital

Resultado - Simulação 2:

Os resultados da simulação estão ilustrados em figura 47 e figura 48, onde as variáveis de estado, a referência e o sinal de controle estão indicados.

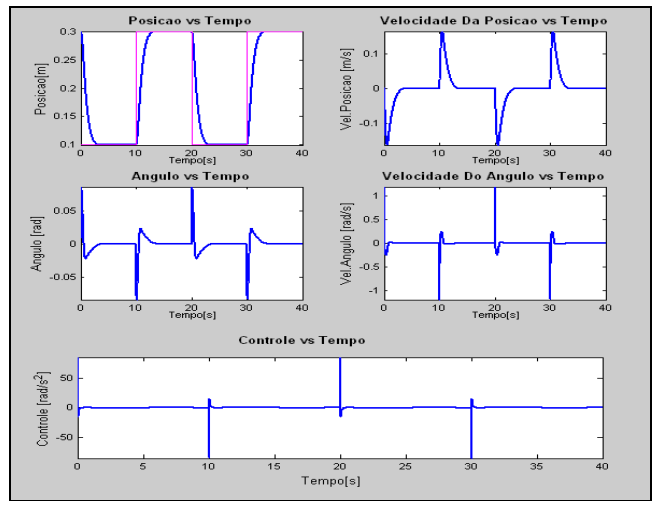

figura 47 : Resultados da simulação 2 para controlador fuzzy com 16 regras

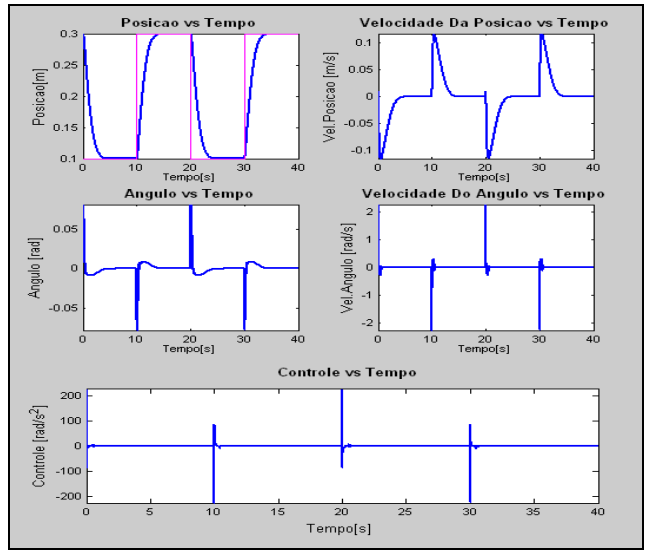

figura 48 : Resultados da simulação 2 para controlador fuzzy com 81 regras

Tais curvas foram obtidas após a definição do limite de excursão das variáveis de acordo com a figura 46.

Os resultados mostram que o controlador fuzzy (de 16 ou 81 regras) é capaz de controlar o sistema teoricamente, usando como entrada os estados do sistema. A curva de resposta para ambos os

controladores é semelhante diferindo apenas sobre a atuação do controlador. No caso, para o controlador fuzzy com 16 regras o valor do sinal de controle atinge valores sobre a faixa de [-200, 200]rad/s^2 enquanto o controlador com 81 regras atua na faixa de [-60, 60] rad/s^2 . Deve-se considerar que nessa simulação nenhum tipo de saturação é imposto ao controlador, o que é fisicamente inviável.

A sintonização dos controladores fuzzy era baseada nos o ajuste de ganhos sobre as variáveis de entrada, estendendo ou comprimindo as funções de pertinência. Essa forma de sintonização consiste em suavizar a ação (estender a função de pertinência) sobre a variável ou intensificá-la (comprimir a função de pertinência) de acordo com a observação da resposta.

Assim, os resultados foram obtidos usando as funções de pertinência ilustradas sobre a figura 49 e figura 50 para os controladores fuzzy com 16 e 81 regras respectivamente.

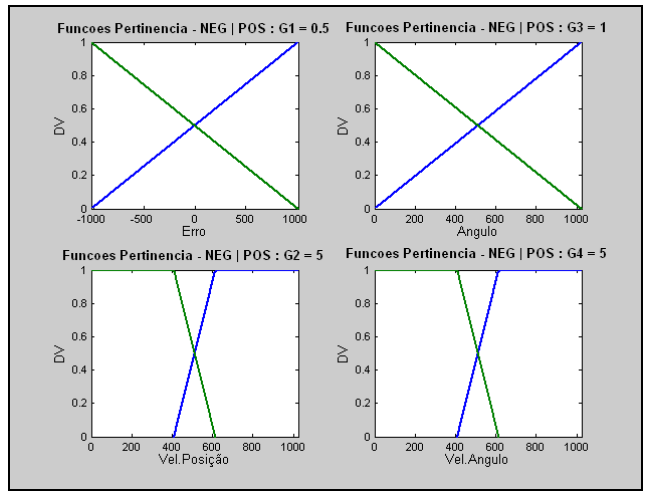

figura 49 : Funções de Pertinência usadas com ganhos G1 = 0.5, G2 = 5, G3 = 1, G4 = 5

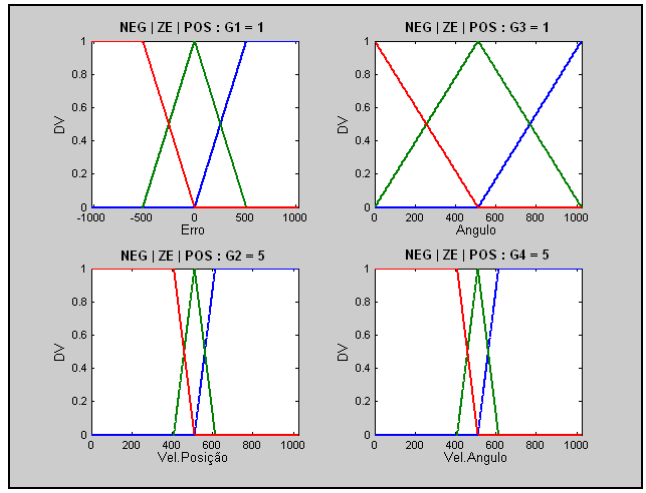

figura 50 : Funções de Pertinência usadas com ganhos G1 = 1, G2 = 5, G3 = 1, G4 = 5

Resultado – Simulação 3:

Como resultado para a **simulação 3** tem-se os gráficos ilustrados na figura 51 e na figura 52, obtidas respectivamente usando os controladores fuzzy com 16 regras e 81 regras. O sistema foi simulado durante 40 segundos tomando como período de amostragem 0.03 s. A curva de referência está ilustrada juntamente com a curva de posição vs tempo.

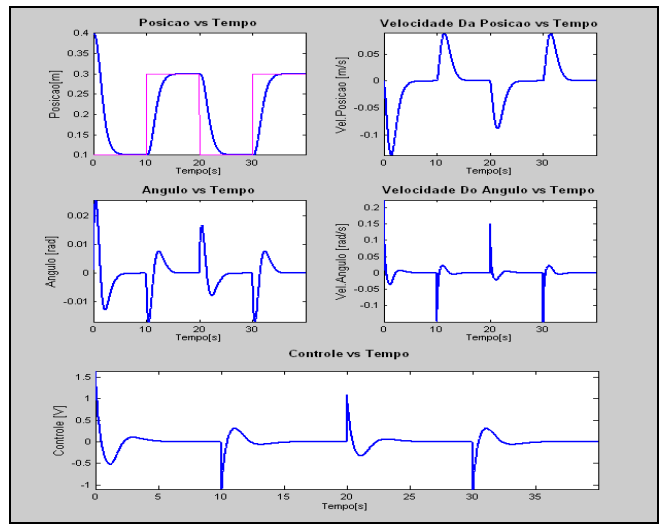

figura 51 : Simulação do sistema para controlador fuzzy com 16 regras

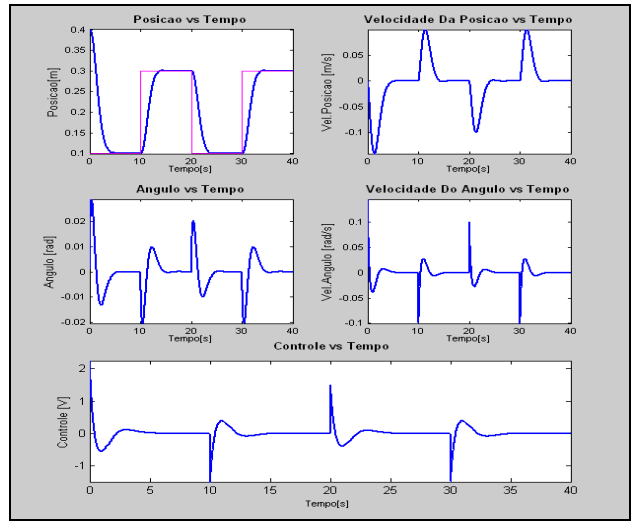

figura 52 : Simulação do sistema para controlador fuzzy com 81 regras

A sintonização dos ganhos foi feita de forma mais intuitiva, já que esta simulação apresentava uma animação gráfica.

As funções de pertinência utilizadas pelos controladores fuzzy de 16 e 81 regras estão ilustradas na figura 53 e na figura 54 respectivamente, com os ajustes de ganhos indicados.

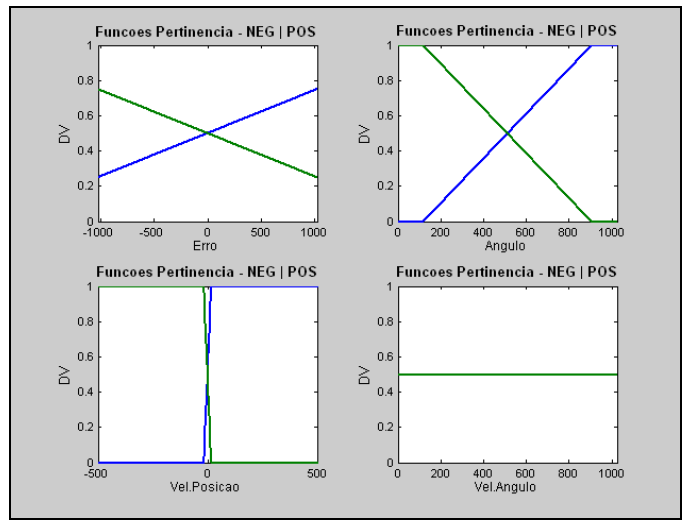

figura 53 : Funções de Pertinência usadas com ganhos G1 = 0.5, G2 = 30, G3 = 13, G4 = 0

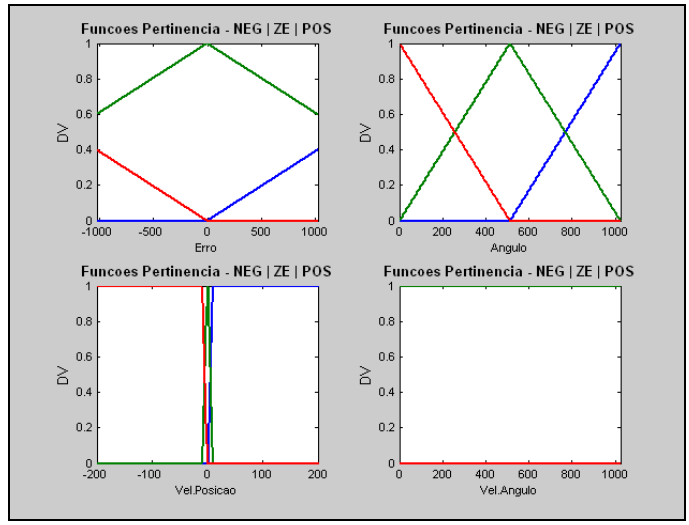

figura 54 : Funções de Pertinência usadas com ganhos G1 = 0.4, G2 = 58, G3 = 1, G4 = 0

As superfícies de controle para ambos os controladores são ilustradas na figura 55.

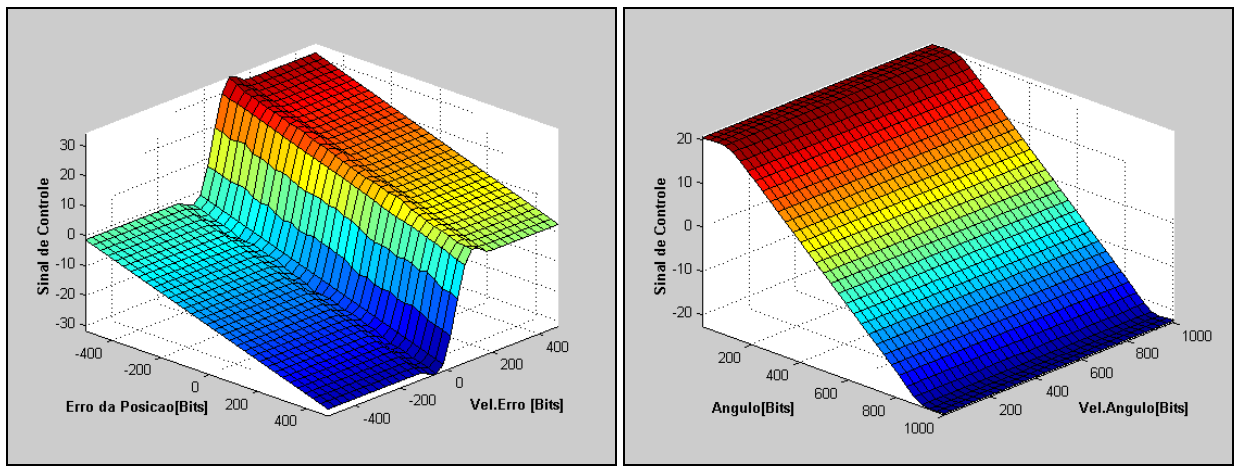

figura 55 : Superfícies de controle para controlador fuzzy com 16 regras e ganhos G1 = 0.5, G2 = 30, G3 = 13, G4 = 0

É possível notar que ambas os controladores apresentam desempenhos muito bons, já que não apresentam erro em regime e conseguem seguir a referência de satisfatória. Percebe-se uma diferença entre os controladores com relação ao transitório, onde o controlador fuzzy com 81 regras alcança mais rapidamente a referência.

Resultado – Simulação 4:

A inclusão de ruídos na simulação afetou bastante o sinal de controle sobre o sistema alterando significativamente a resposta do controlador fuzzy com 16 regras como ilustra a figura 56. Já para o controlador fuzzy com 81 regras, o ruído, apesar de afetar bastante o sinal de controle, não alterou significativamente a resposta do sistema, como mostra a figura 57. Para essa simulação, foram usados os mesmos ganhos da **simulação 3**, representados pelas funções de pertinência ilustradas nas figuras figura 53 e figura 54.

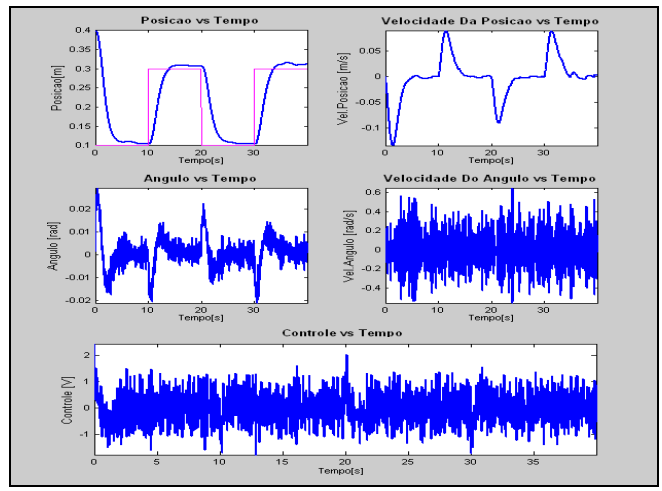

figura 56 : Inclusão de ruído sobre controlador fuzzy com 16 regras

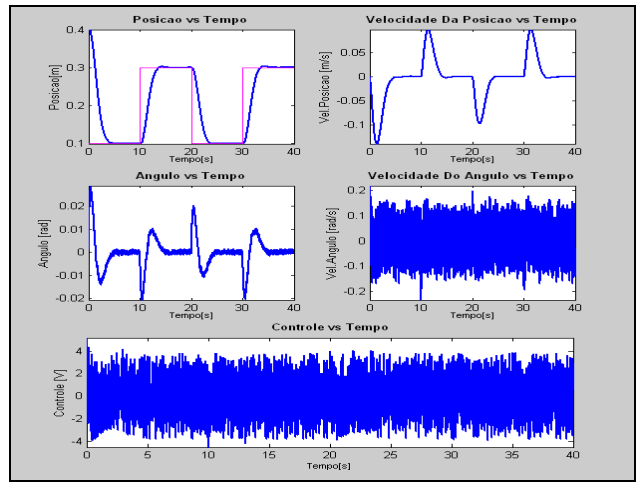

figura 57 : Inclusão de ruído sobre controlador com 81 regras

Assim, uma forma para melhorar a resposta do sistema seria incluir na malha de controle um filtro digital sobre o sinal de controle, principalmente para o controlador fuzzy com 16 regras.

No entanto, antes de implementar um filtro digital, foi acrescentado um integrador sobre o sinal do erro da posição da bola para o controlador fuzzy com 16 regras, para tentar eliminar o erro sem em regime. O controlador fuzzy com 81 regras não apresentou erro em regime permanente, mesmo com o ruído, logo a inclusão de um integrador não é justificada.

O resultado da simulação com ruído para o controlador fuzzy com 16 regras e um integrador é dado pela figura a seguir.

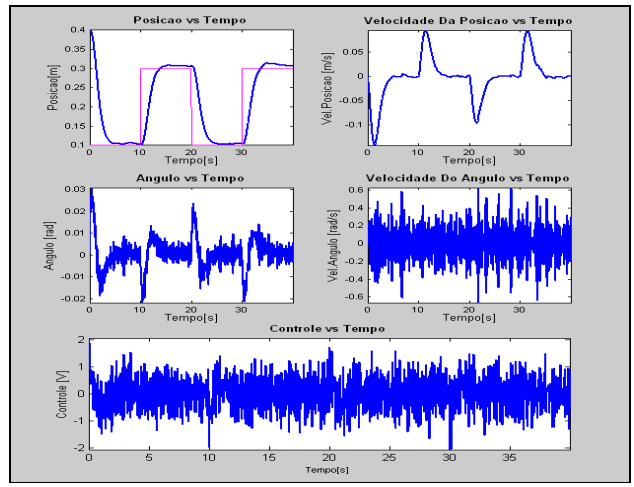

figura 58 : Resposta e sinal de controle para simulação com ruído e canal integral

Como o sistema ainda apresenta um erro em regime permanente, cabe a implementação de um filtro digital para melhorar o sinal de controle sobre o sistema. No caso, o filtro digital foi utilizado para ambos os controladores fuzzy. As respostas do sistema então são apresentadas na figura 59 e figura 60.

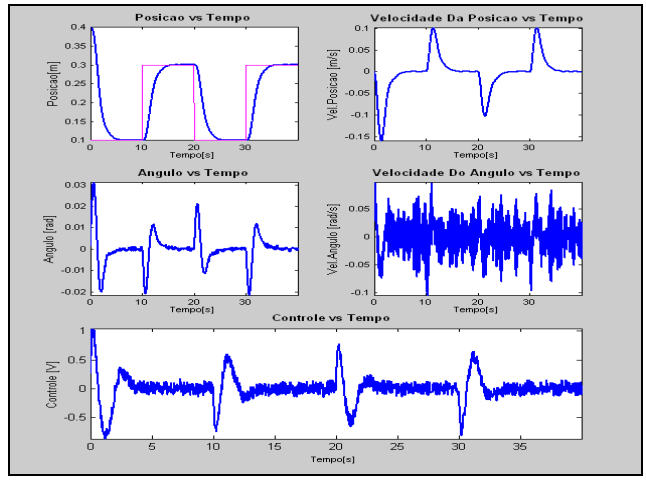

figura 59 : Resposta para simulação com 16 regras utilizando um filtro sobre o sinal de controle

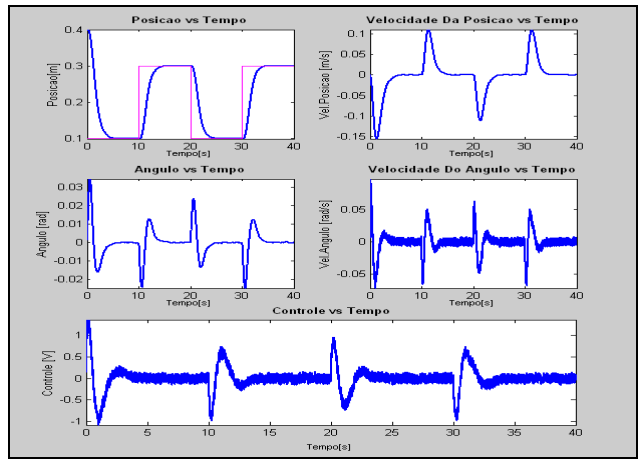

figura 60 : Resposta para simulação com 81 regras utilizando um filtro sobre o sinal de controle

É possível perceber então que a resposta de ambos os controladores apresenta-se bem comportada, com erro nulo em regime permanente. Resta agora avaliar os resultados da implementação do controlador fuzzy com 16 regras sobre a bancada experimental. A sintonização das simulações serviu como base para o ajuste do controlador durante a experimentação.

4.2 Resultados sobre a Bancada Experimental

Os resultados obtidos durante a experimentação utilizaram os mesmos parâmetros da simulação. No caso tais parâmetros eram referentes somente as simulações para o controlador fuzzy com 16 regras, já que o controlador fuzzy com 81 regras não foi implementado no PIC. Ou seja, durante a experimentação foram usados os mesmos ganhos, período de amostragem, referência e outros parâmetros que foram utilizados na simulação.

Assim, as curvas obtidas foram as seguintes:

Simulação inicial, sem canal integral e sem filtragem:

Ganhos G1 = 0.5, G2 = 30, G3 = 1.3, G4 = 0 e GU = 1.3 (figura 53)

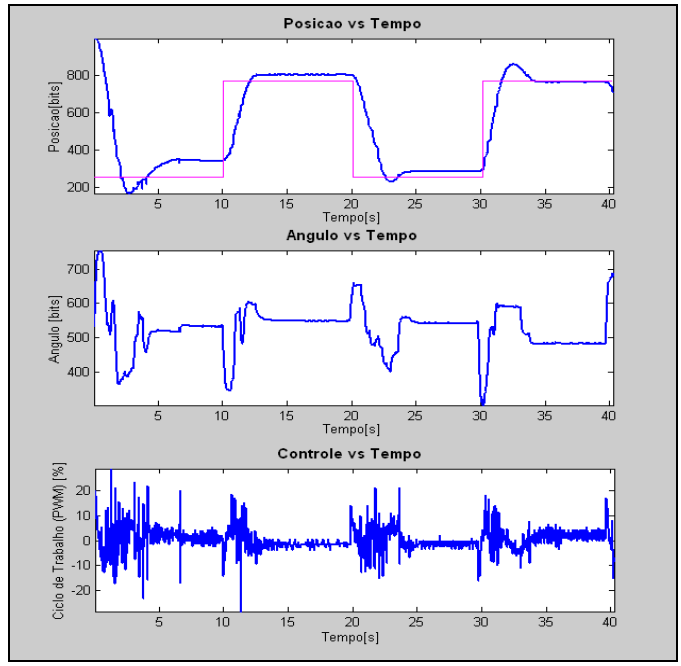

figura 61 : Atuação sem ação integral e sem filtragem

Percebe-se com estes resultados que o sistema apresentou um bom comportamento, semelhante aos resultados da **simulação 4** ilustrados na figura 56, apesar de possuir um leve sobrepasso.

Seguindo a proposta da simulação, foi implementado o canal integral sobre o erro da posição com o intuito de eliminar o erro em regime e o resultado é apresentado na figura 62.

Simulação com canal integral (Ti = 0.0045) e sem filtragem:

Ganhos G1 = 0.5, G2 = 30, G3 = 1.3, G4 = 0 e GU = 1.3 (figura 53)

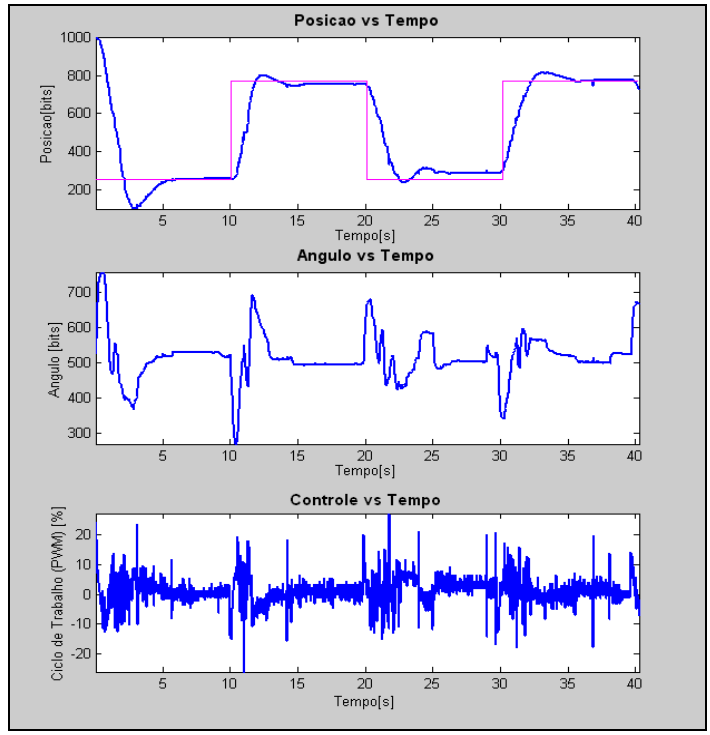

figura 62 : Atuação com ação integral sobre o erro da posição

Apesar da ação do canal integral, o sistema ainda apresentou um erro em regime, o que não aconteceu na simulação realizada, como mostra a figura figura 58. No entanto, esse erro pode ser atribuído a uma característica não linear do sistema, que não está incluída na simulação: o atrito entre a bola e a pista. Em algumas situações, mesmo com uma inclinação positiva ou negativa do ângulo, a bola matinha parada, sem conseguir romper o atrito estático com a pista.

Agora, procurando suavizar a curva de controle, foi acrescentado ao sistema um filtro digital de 1ª sobre o sinal de controle. O resultado é apresentado a seguir.

Simulação sem canal integral e filtragem digital (1hz):

Ganhos G1 = 0.47, G2 = 30, G3 = 1.3, G4 = 0 e GU = 1,3 (figura 53)

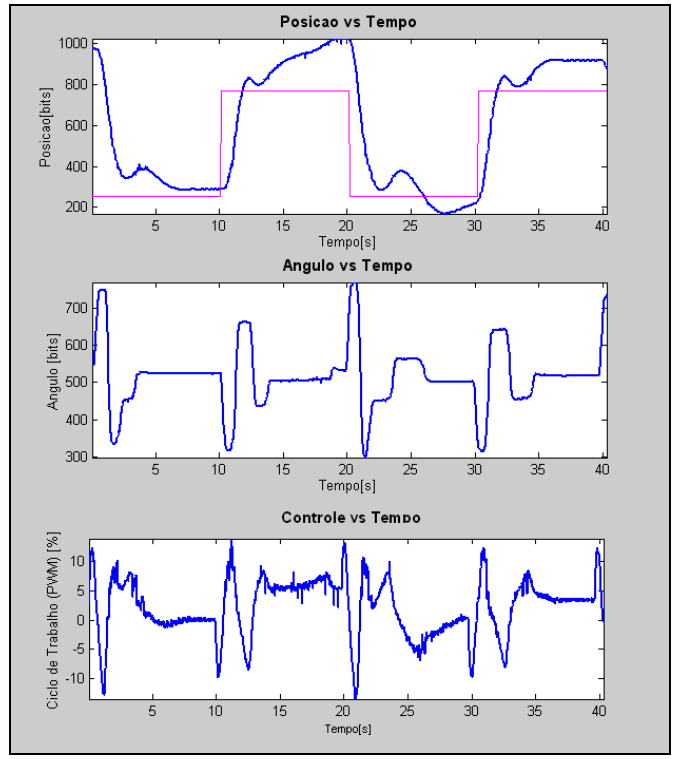

figura 63 : Atuação sem ação integral e com filtragem digital (1hz freqüência de corte)

Com a ação suavizada do controlador o sistema não conseguiu foi possível, com os mesmos ganhos controlar a posição da bola. Percebeu-se durante a experimentação que ao utilizar a filtragem, a zona morta de atuação do motor influenciou bastante o comportamento do sistema. Pode-se observar na figura 63, que entre 15 e 20 segundos o controlador comandava uma ação positiva para levantar o ângulo da pista, porém este permaneceu parado.

Assim como foi feito para a implementação sem filtragem, foi adcionada uma ação integral para tentar compensar esse efeito.

Simulação com canal integral (Ti = 0.0045) e filtragem digital (1hz):

Ganhos G1 = 0.47, G2 = 30, G3 = 1.3, G4 = 0 e GU = 1.3

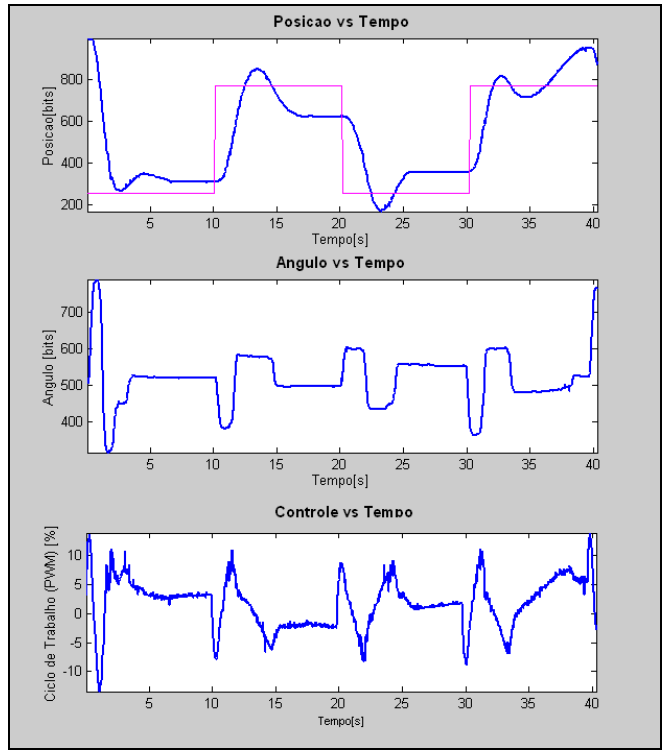

figura 64 : Atuação com ação integral e com filtragem digital (1hz freqüência de corte)

Mesmo com a ação integral, o sistema com a filtragem digital não apresentou bons resultados. Embora tenha sido evitada a zona morta do motor ao estabilizar a posição da bola, esta apresentou um erro significativo.

5. CONCLUSÃO

Apesar das dificuldades inerentes na concepção de qualquer projeto mecânico, a estrutura física resultante apresentou-se como uma boa plataforma experimental. Os elementos sensores usados na pista mostraram-se eficazes na aquisição de informação para os ensaios experimentais e o acionamento apresentou um bom desempenho conseguindo responder de acordo com a dinâmica exigida pelo sistema barra-esfera.

Desse modo, pode-se dizer que o primeiro objetivo proposto para este trabalho , que consistia na construção de uma bancada experimental Barra-Esfera, foi alcançado de forma satisfatória. Assim, com a bancada montada, é possível atualmente estudar de forma prática diferentes técnicas de controle sobre sistema, como era esperado. Ainda, devido à semelhança entre os resultados obtidos pela simulação e os resultados experimentais, é possível também estudar previamente o sistema em simulação com certa confiança de que o controlador ai projetado terá um comportamento semelhante quando implementado experimentalmente.

Com relação à implementação do controle fuzzy sobre o sistema, pode-se dizer que esta foi realizada com êxito. De acordo com as curvas obtidas na seção 4.2, a abordagem de controle usado sobre o sistema (figura 30) obteve um desempenho satisfatório, de modo que o controlador fuzzy com 16 regras usado conseguiu estabilizar a esfera sobre a barra com um erro aceitável. Desse modo, o segundo objetivo proposto, que era o controle do sistema por meio de controle fuzzy foi alcançado.

Avaliando o projeto do controlador fuzzy, pode-se dizer que, como esperado, ele apresentou-se como uma forma bastante empírica de abordar um problema de controle. Apesar de exigir mais com relação a sua implementação em termos de programação e esforço computacional, a forma de projetar o controle mostrou-se simples e eficaz, devido a sua característica intuitiva.

PROPOSTA PARA TRABALHOS FUTUROS

Para pessoas que tenham interesse em continuar o trabalho aqui apresentado, seguem abaixo algumas recomendações e sugestões para novos trabalhos:

- Sistema físico: como a barra que está sendo usada sobre a bancada experimental foi emprestada pelo prof.Lélio, fica como recomendação adquirir uma nova barra feita de material resistivo para ser incorporada ao sistema, ou utilzar outros métodos para medir a posição da esfera no sistema, como visão computacional por exemplo.

- Instrumentação: considerando que todos os circuitos foram implementados em protoboard e que outras formas de condicionamento ainda podem ser testadas, fica como sugestão para próximos trabalhos, buscar alternativas para os condicionamentos de sinais utilizados e recomenda-se confeccionar os circuitos que venham a ser propostos em placas de circuito impresso.

Ainda, fica como sugestão para prómixos trabalhos utilizar uma memória auxiliar junto com o microcontrolador PIC ou buscar outros microcontroladores com este recurso mais avançado, para que possam ser testados controladores fuzzy mais complexos, como o controlador com 81 regras.

- Modelamento do Sistema e Simulações: uma proposta para continuação do trabalho dentro desse tema seria buscar encontrar as equações dinâmicas que descrevem o sistema incluindo o sistema de manivela usado. Ainda, uma outra sugestão seria realizar a identificação do sistema para comparação com o modelo utilizado atualmente nas simulações.

- Sistema de Controle: aqui vários propostas podem ser implementadas. Fica como sugestão inicial procurar sintonizar o ganho de saído de forma não linear para compesar o efeito da zona morta do motor quando utilizado um filtro digital, como verificado na figura 63. Ainda, outras sugestões seria estudar novas abordagens que podem reduzir o número de regras utilizadas pelo controlador fuzzy como por exemplo controle fuzzy em modo deslizante[13], controle fuzzy hierárquico[3] ou a abordagem do controle em cascata[12].

REFERÊNCIAS BIBLIOGRÁFICAS

[1] R. Dettmer,"Fuzzy control: engineering for empiricists,"*IEEE Review,* vol.40, nº1, pg.17-20, 20 Jan. 1994.

[2] L.X. Wang, *A course in Fuzzy Systems and Control*, 1ªed. New Jersey,EUA: Prentice Hall PTR, 1997.

[3] L.X. Wang, "Analysis and design of hierarchical fuzzy systems," *IEEE Trans. Fuzzy Syst.*, vol.7, nº5, outubro 1999.

[4] K.M.Passino, S.Yurkovich, *Fuzzy Control*, 1ªed. Califórnia,EUA: Addison Wesley Longman Inc.1998.

[5] C.C. Lee, "Fuzzy logic in control systems: fuzzy logic controller – Part I," *IEEE Trans. Syst. Man Cybern.,* vol.20, nº2, março/abril 1990.

[6] F.O. Rodriguez, "Modelado y control PD-difuso em tiempo real para el sistema barra-esfera," 2004, 141 pg., Dissertação (Mestrado em Controle Automático), Centro de envestigación y de estudios avanzados del instituto politécnico nacional, México, D.F.

[7] M. Virseda, "Modeling and control of the ball and beam process," março 2004, Dissertação (Mestrado em Controle Automático), Lund Institute of Technology, Lund-Suécia.

[8] J. Jantzen, "Design of fuzzy controllers," Technical University of Denmark, Department of Automation, Dinamarca, 30 set. 1999.

[9] J. Jantzen, "A robustness study of fuzzy control rules," Technical University of Denmark, Department of Automation, Dinamarca.

[10] J. Hauser, S. Sastry, P. Kokotovic, "Nonlinear control via approximate input-output lineartization: the ball and beam example," *IEEE Trans. Autom. Control,* vol.37, nº3, março 1992.

[11] J.M. Zhang, R.H. Li, P.A. Zhang, "Stability analysis and systematic design of fuzzy control systems," *Elsevier - Fuzzy sets and syst.,* vol.120, pg.65-72, 2001.

[12] R. Ordonez, J. Zumberge, J.T. Spooner, K.M. Passino, "Adaptive fuzzy control: experiments and comparative analyses," *IEEE Trans. Fuzzy Syst.,* vol.5, nº2, maio 1997.

[13] J.C. Lo, Y.H. Kuo, "Decoupled fuzzy sliding-mode control," *IEEE Trans. Fuzzy Syst.,* vol.6, nº6, agosto 1998.

[14] K. Ogata, *Engenharia de controle moderno,* 3ªed. Rio de Janeiro, RJ: Livros Técnicos e Científicos Ed. S.A., 2000.

[15] B.Messner, D. Tilbury, "Control Tutorials for Matlab," Disponível em: http://www.engin.umich.edu/group/ctm/

[16] A.S. Sedra, K.C. Smith, *Microeletrônica,* 4ªed. São Paulo, Brasil: Peason Education do Brasil, 2000.

[17] J.M.F.Cunha, "Projeto de Graduação: Ball and Beam", CD-ROOM

6. APENDICE

6.1 Lógica fuzzy

6.1.1 Conjuntos Fuzzy e Operações

A base de toda a lógica fuzzy está relacionada à generalização da visão binária sobre a teórica clássica de conjuntos. Essa teoria define os conjuntos a partir de propriedades restritivas que agregam apenas duas possibilidades aos elementos, ou um elemento atende a propriedade ou não atende.

Os conjuntos fuzzy associam um grau de pertinência aos elementos de um domínio U qualquer. Desse modo, pode-se dizer que um elemento atende a propriedade de um conjunto em certo grau de verdade. A definição de um conjunto fuzzy A [2] é dada por:

$$A = \{(x, u_A(x) \mid x \in U\} \qquad \text{(A.1)}$$

, em que x é um elemento pertencente ao domínio U, chamado de universo de discurso, e $u_A(x)$ é a função de pertinência que associa o elemento x a um grau de pertinência definido no intervalo [0,1].

Alguns exemplos de funções de pertinência apresentados em [3] são mostrados na figura 65.

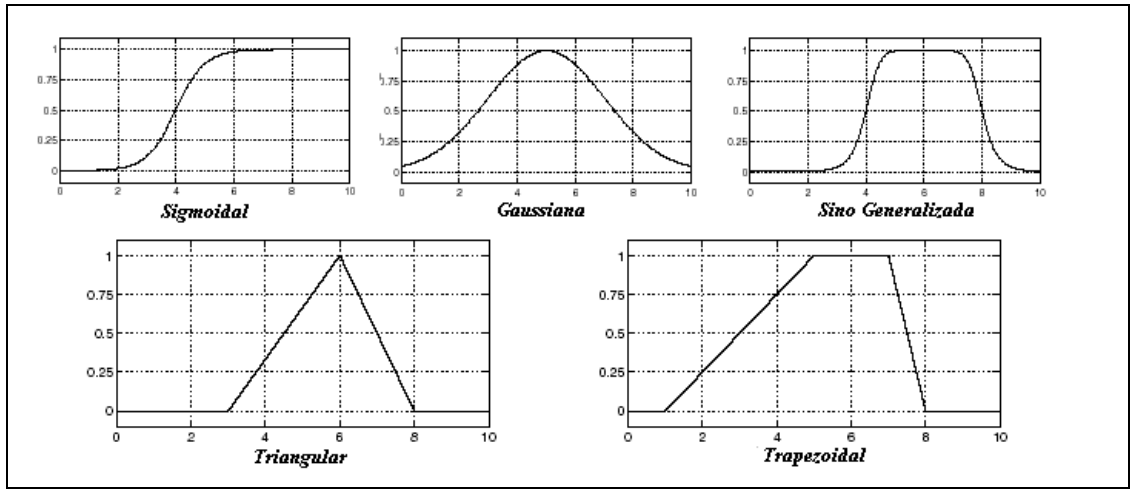

figura 65 : Exemplos de função de pertinência

Os conjuntos fuzzy podem ser separados em dadas categorias [2] de acordo com os valores assumidos por U:

$$U - discreto : A = \sum_U u_A(x) / x \qquad \text{(A.2)}$$

$$U - contínuo : A = \int_U u_A(x) / x \qquad \text{(A.3)}$$

Pode-se tomar como exemplo para ilustrar o conceito de grau de pertinência o conjunto fuzzy dos dias que fazem parte do fim de semana. De acordo coma figura 66, percebe-se que a sexta-feira está associada ao conjunto dos dias do fim de semana pelo grau de pertinência 0.8 tanto no caso discreto quanto no caso contínuo, enquanto o conjunto clássico apenas descarta sexta-feira dos dias do fim de semana.

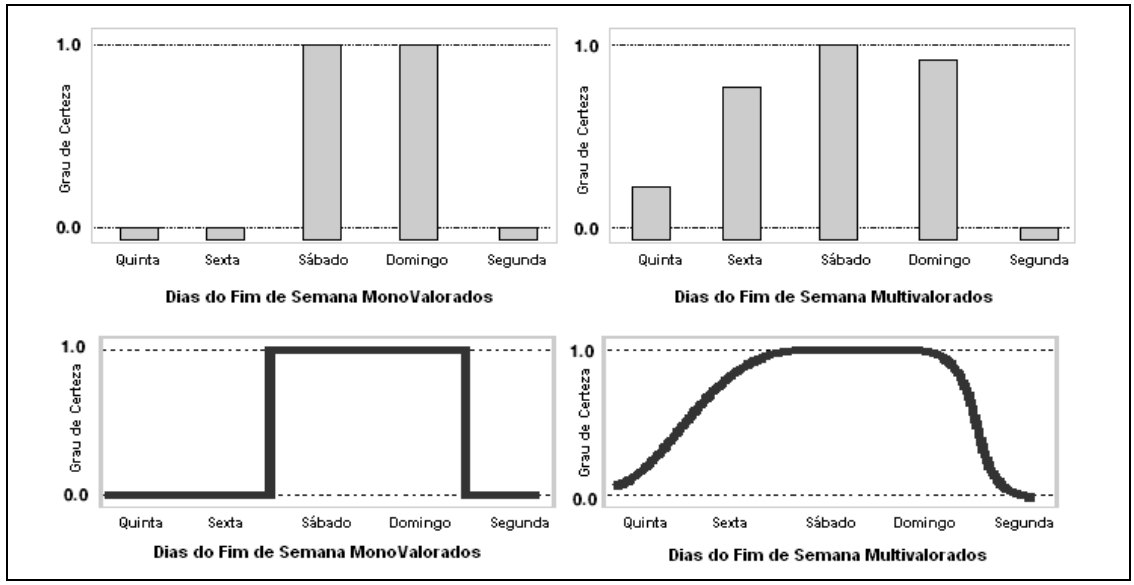

figura 66 : Demonstração de grau de pertinência e conjuntos fuzzy

Algumas características importantes que estão associadas aos conjuntos fuzzy são:

→ suporte: conjunto de todos os valores em U que possuem grau de pertinência na nulo.

$$\sup(A) = \{x \in U \mid u_A(x) > 0\} \qquad \text{(A.4)}$$

→ centro: valor médio entre o conjunto de valores que possui grau de pertinência máximo da função de pertinência. Caso o valor médio seja infinito positivo(ou negativo) o centro é definido como o menor (ou maior) valor dentre os que atingem grau de pertinência máximo.

→ altura: é o maior valor de pertinência assumido pela função de pertinência de um conjunto. Caso a altura seja 1, o conjunto é chamado de conjunto fuzzy normal.

→ corte-α: é o conjunto A_α que possui todas os elementos em U que possuem grau de pertinência maior que α.

$$A_\alpha = \{x \in U \mid u_A(x) \geq \alpha\} \qquad \text{(A.5)}$$

Geralmente, em sistemas fuzzy, as funções de pertinência estão associadas a variáveis lingüísticas. Para o exemplo citado anteriormente,

As operações entre conjuntos fuzzy são as mesmas usadas na teoria clássica de conjuntos. União, intersecção e complemento são as operações herdadas pelos conjuntos fuzzy, porém elas são definidas com base em alguns axiomas [2] que devem ser satisfeitos para que uma operação possa ser definida.

Complemento Fuzzy

A operação de complemento *c* sobre a função de pertinência de um conjunto fuzzy A é apresentada como:

$$c[u_A(x)] = u_{\bar{A}}(x) \qquad \text{(A.6)}$$

Os axiomas que o complemento fuzzy deve satisfazer são:

Axioma c1: *c(0) = 1, c(1)=0* (condição de borda)

Axioma c2: para todo $a,b \in [0,1]$, se $a<b$ então $c(a) \geq c(b)$ (condição não-crescente)

Exemplos de operações de complemento[2]:

i. complemento básico:

$$c[a] = 1 - a \qquad \text{(A.7)}$$

ii. complemento Sungeno:

$$c_\lambda[a] = (1-a)/(1+\lambda \cdot a) \qquad \text{(A.8)}$$

iii. complemento Yager:

$$c_w[a] = (1 - a^w)^{1/w} \qquad \text{(A.9)}$$

União Fuzzy ou S-Norma

A operação de união que define uma nova função de pertinência entre os conjuntos fuzzy *a* e *b* é apresentada como:

$$s[a,b] = u_{a \cup b}(x)\,\text{s} \qquad \text{(A.10)}$$

Os axiomas que a operação deve satisfazer para ser considerado uma união fuzzy são:

Axioma s1: *s(1,1)=1, s(0,a)=s(a,0)=a* (condição de borda)

Axioma s2: *s(a,b)=s(b,a)* (condição comutativa)

Axioma s3: se $a \leq a'$ e $b \leq b'$ então $s[a,b] \leq s[a',b']$(condição não-decrescente)

Axioma s4: *s[s[a,b],c] = s[a,s[b,c]]* (condição associativa)

Exemplos de união fuzzy[2]:

i. união max:

$$u_{a \cup b}(x) = \max(a,b) \qquad \text{(A.11)}$$

ii. Soma drástica:

$$s_{sd}[a,b] = \begin{Bmatrix} a,'se'b = 0 \\ b,'se'a = 0 \\ 0,'outros' \end{Bmatrix} \qquad \text{(A.12)}$$

iii. soma algébrica:

$$s_{sa}[a,b] = a + b - a \cdot b \qquad \text{(A.13)}$$

Intersecção Fuzzy ou T-Norma

A operação de intersecção que define uma nova função de pertinência entre os conjuntos fuzzy *a* e *b* é apresentada como:

$$t[a,b] = u_{a \cap b}(x) \qquad \text{(A.14)}$$

Os axiomas que essa operação deve satisfazer para ser considerada uma intersecção fuzzy são:

Axioma s1: $t(0,0)=0,\ t(1,a)=s(a,1)=a$ (condição de borda)

Axioma s2: $s(a,b)=s(b,a)$ (condição comutativa)

Axioma s3: se $a \leq a'$ e $b \leq b'$ então $s[a,b] \leq s[a',b']$ (condição não-decrescente)

Axioma s4: $t[t[a,b],c] = t[a,t[b,c]]$ (condição associativa)

Exemplos de intersecção fuzzy[2]:

i. Operação min:

$$t[a,b] = \min[a,b] \qquad \text{(A.15)}$$

ii. produto drástico:

$$t_{pd}[a,b] = \begin{cases} a, 'se' b = 1 \\ b, 'se' a = 1 \\ 0, 'se' outros \end{cases} \qquad \text{(A.16)}$$

iii. produto algébrico:

$$t_{pa}[a,b] = a \cdot b \qquad \text{(A.17)}$$

Segundo [2] as operações mais usadas nas aplicações de engenharia são o complemento básico, a t-norma mínimo ou produto algébrico e a s-norma max ou soma algébrica.. A figura 3 ilustra a aplicação de alguns dos operados citados anteriormente.

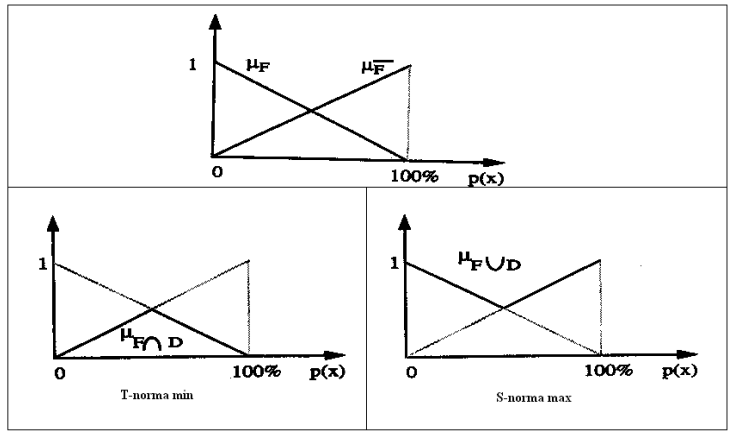

figura 67 : de operações t-norma min(15) e s-norma max (11)

Antes de finalizar esse tópico, devem ser introduzidos os conceitos de relações e composições fuzzy.

Uma relação fuzzy representa o grau de associação entre os elementos de dois ou mais conjuntos fuzzy e é definida sobre o produto cartesiano entre os universos de discursos dos conjuntos. A definição formal apresentada por [2] é dada por:

$$Q = \{((u_1, u_2, ..., u_n), u_Q(u_1, u_2, ..., u_n)) \mid (u_1, u_2, ..., u_n) \in U_1 \times U_2 \times ... \times U_n) \qquad \text{(A.18)}$$

,onde $u_Q: (u_1, u_2, ..., u_n) \rightarrow [0,1]$

Uma forma mais intuitiva de representar relações entre conjuntos fuzzy é a representação lingüística. Por exemplo, "*a* está perto que *b*" e "*x* é aproximadamente igual a *y*" são relações que associam elementos entre os conjuntos fuzzy indicados. As relações podem ser tanto definidas sobre universos discretos como contínuos.

Vale lembrar que as relações também são conjuntos fuzzy e todas as operações definidas anteriormente podem ser aplicadas a elas.

Existindo a necessidade de associar duas relações entre si, uma composição deve ser criada. Dessa forma, dada as relações *P*, definida sobre $(x,y) \in UxV$, e *Q* definida sobre $(y,z) \in VxW$, a composição $P \circ Q$ é definida por:

$$u_{P \circ Q}(x,z) = \max_{Y \in V} t[u_P(x,y), u_Q(y,z)] \qquad \text{(A.19)}$$

,em que *t* é uma t-norma.

A composição, sendo uma associação entre relações, é também uma relação e está definida também sobre os domínios discreto e contínuo.

A execução da composição entre relações em domínio discreto pode ser executada por meio de uma simbólica multiplicação de matrizes onde a multiplicação resultante entre os elementos é substituída pela t-norma usada na composição.

A composição em domínio contínuo exige um maior esforço matemático. Tomando as relações citadas anteriormente e realizando a composição com uma t-norma produto algébrico, percebe-se que é necessário utilizar técnicas de derivação para executar a operação $\max_{Y \in V}$.

6.1.2 Regras SE-ENTÃO

A formação do controle fuzzy inicia-se a partir da criação das regras fuzzy baseadas na lógica proposicional SE-ENTÃO como apresenta a tabela 1.

Tabela 1: Lógica clássica proposicional

p	q	$p \wedge q$
T	T	T
T	F	F
F	T	F
F	F	F

O primeiro passo para descrever as regras é a descrição das proposições.

Existem dois tipos de proposições: a proposição atômica, formada por apenas uma frase possuindo uma única variável lingüística, e a proposição composta, formada por pelo menos duas proposições atômicas unidas por conectivos *e*, *ou* e *negação(não).*

A interpretação dos conectivos é feita respectivamente pelos operadores *intersecção (t-norma), união (s-norma)* e *complemento*. Dessa forma, as proposições são definidas como relações fuzzy.

Tomando como exemplo a proposição *erro(e) é grande e delta-erro(de) é pequeno* a respectiva função de pertinência é dada por:

$$u_{grabde \cap pequeno}(e, de) = t[u_{grande}(e), u_{pequeno}(de)] \qquad \text{(A.20)}$$

Como seria então definida a operação de implicação que relaciona a proposição *se* (antecedente) com a proposição *então* (conseqüente). Seguindo a lógica clássica, as implicações podem ser interpretadas conforme as tautologias.

$$(p \rightarrow q) \leftrightarrow (\overline{p} \vee q) \qquad \text{(A.21)}$$

$$(p \rightarrow q) \leftrightarrow ((p \wedge q) \vee \overline{q}) \qquad \text{(A.22)}$$

A partir das tautologias (A.21) e (A.22) o operador de implicação é definido usando as operações já conhecidos t-norma, s-norma e complemento. Como são vários os tipos de operações fuzzy existem inúmeras formas de interpretar as regras fuzzy, que serão escolhidas de acordo com a aplicação.

Para aplicações de engenharia, a implicação mais usada é a implicação Mamdani, e forma que a regra *se $p_1(x)$ então $p_2(y)$* é dada por:

$$u_{p1\to p2}(x,y) = \min[u_{p1}(x), u_{p2}(y)] \qquad \text{(A.23)}$$

ou $$u_{p1\to p2}(x,y) = u_{p1}(x)\cdot u_{p2}(y) \qquad \text{(A.24)}$$

Essa implicação na é interpretada pelas tautologias (A.21) ou (A.22) e sim a partir de $p \to q = p \wedge q$ e preserva a relação de causa-efeito que é exigida em uma aplicação de engenharia, já que a ausência de uma entrada (não-causa) deveria acarretar em nenhuma ação. Essa regra de implicação é proposta por Mamdani é apresentada em [2] como uma implicação de engenharia onde também são mostrados os casos em que a relação causa-efeito não é preservada para outras regras de implicação.

Inferência de Regras

Considerando que para uma dada regra *se x é A* $\to$ *y é B* onde é conhecido que *x* é *A'* (*x* é *A* em um certo grau de verdade) o que se pode inferir a respeito de *y* ? Nesse sentido, a lógica clássica possui regras de inferência baseadas em premissas e conseqüências chamadas Modus Ponens, Modus Tolens e Silogismo Hipotético. A lógica fuzzy estende esses conceitos para formas generalizadas que agreguem sua lógica multivalorada. As definições destas formas são:

Moduns Pones Generalizado: dada duas proposições *p* e *q* que compõem a regra $p \to q$ a inferência feita pelo Modus Pones clássico é:

$$\underbrace{(p}_{\textit{premissa}} \wedge (p \rightarrow q)) \rightarrow \underbrace{q}_{\textit{consequência}} \qquad \text{(A.25)}$$

A forma generalizada que agrega os valores fuzzy é dada por:

Premissa:	*x é A'*
Regra:	***se*** *x é A* ***então*** *y é B*
Conseqüência :	***y*** *é B'*

, em que o grau de verdade de *A'* irá definir o grau de verdade da conseqüência *B'*.

Moduns Tolens Generalizado: dada duas proposições *p* e *q* q eu compõem a regra $p \rightarrow q$ a inferência feita pelo Modus Tolens clássico é:

$$\underbrace{(q}_{\textit{premissa}} \wedge (p \rightarrow q)) \rightarrow \underbrace{p}_{\textit{conseqüênica}} \qquad \text{(A.26)}$$

A forma generalizada que agrega os valores fuzzy é dada por:

Premissa:	*y é B'*
Regra:	***se*** *x é A* ***então*** *y é B*
Conseqüência :	*x é A'*

, em que o grau de verdade de *B'* irá definir o grau de verdade da conseqüência *A'*.

Silogismo Hipotético Generalizado: dada duas proposições p e q que compõem a regra $p \rightarrow q$ a inferência feita pelo Silogismo Hipotético clássico é:

$$\underbrace{((p \rightarrow q)}_{\textbf{premissa1}} \wedge \underbrace{(q \rightarrow z))}_{\textbf{premissa2}} \rightarrow \underbrace{(p \rightarrow r)}_{\textbf{conseqüência}} \qquad \text{(A.27)}$$

A forma generalizada que agrega os valores fuzzy é dada por:

Premissa 1:	***se*** *x é A* ***então*** *y é B*
Premissa 2:	***se*** *y é B'* ***então*** *z é C*
Conseqüência:	***se*** *x é A* ***então*** *z é C'*

, em que o grau de verdade de *B'* irá definir o grau de verdade da conseqüência *C'*.

Assim, é possível definir para cada regra de inferência a função de pertinência que representa o resultado inferido de acordo com as premissas dadas. As definições são dadas por:

. MPG: $u_{B'}(y) = \max_{x \in U} t[u_{A'}(x), u_{A \rightarrow B}(x, y)]$ (A.28)

. MTG: $u_{A'}(x) = \max_{y \in V} t[u_{B'}(y), u_{A \rightarrow B}(x, y)]$ (A.29)

. SHG: $u_{A \rightarrow C'}(x, z) = \max_{y \in V} t[u_{A \rightarrow B}(x, y), u_{B' \rightarrow C}(y, z)]$ (A.30)

6.1.3 Sistema Fuzzy

Um sistema fuzzy é composto das seguintes partes como ilustra a figura 68.

1. Fuzzificador: etapa onde uma entrada na fuzzy é associada a um conjunto fuzzy por meio de um grau de pertinência.
2. Base de Regras: conjunto de todas as regras que descrevem o sistema fuzzy.
3. Máquina de Inferência: etapa em que as entradas fuzzificadas são avaliadas de acordo com a Base de Regras e são inferidas as saídas fuzzy.
4. Defuzzificador: realiza a associação inversa de um fuzzificador usando a saída fuzzy inferida para gerar uma saída numérica.

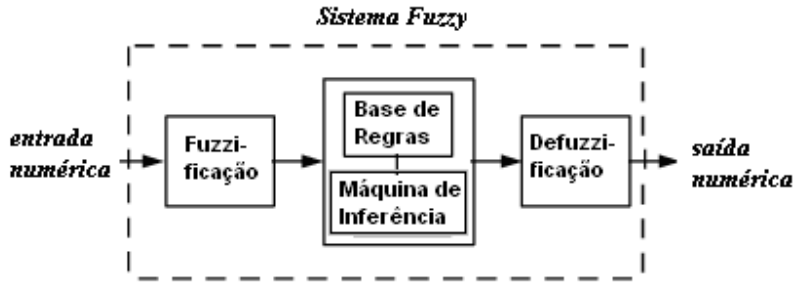

figura 68 : Esquema de um sistema fuzzy

Fuzzificação

Considerando uma definição mais formal para esta etapa, pode-se dizer que a fuzzificação é um mapeamento de um número real $\boldsymbol{x}^* \in U \subset R^n$ para um conjunto fuzzy *A'* em *U*. São apresentadas a seguir 3 fuzzificadores propostos por [2] que possuem propriedades distintas.

2. Fuzzificador Singleton: possui a característica de simplificar a complexidade computacional no processo de inferência.

$$u_{A'}(x) = \begin{cases} 1, 'se' \, x = x^* \\ 0, 'se' \, outros \end{cases} \qquad \text{(A.31)}$$

3. Fuzzificador Gausssiano: quando as funções de pertinência são gaussianas, o uso desse fuzzificador reduz a complexidade dos cálculos e reduz a influência de ruídos sobre *x*.

$$u_{A'}(x) = t[e^{-\left(\frac{x1-x1^*}{a1}\right)^2},...,e^{-\left(\frac{xn-xn^*}{an}\right)^2}] \qquad \text{(A.32)}$$

, onde a_n são parâmetros positivos.

4. Fuzzificador Triangular: quando usado sobre funções de pertinência triangulares a complexidade computacional é reduzida e a imunidade a distúrbios sobre $\boldsymbol{x}^*$ é aumentada.

$$u_{A'}(x) = \begin{cases} t[(1-|x_1 - x^*_1|)/b_1,...,(1-|x_n - x^*_n|)/b_n, 'se' |x_i - x^*_i| \le b_i \\ 0, 'se' outros \end{cases} \qquad \text{(A.33)}$$

, em que $i=1,2,...,n$ e b_i são parâmetros positivos

Máquina de Inferência

São responsáveis pela avaliação do conjunto de regras sobre uma entrada fuzzy pela inferência gerando uma saída fuzzy que será posteriormente defuzzificada. Ou seja, na máquina de inferência é feito um mapeamento de um conjunto fuzzy *A'* em *U* para um conjunto fuzzy *B'* em *V*.

Como foi visto anteriormente, a inferência de uma única regra é feita por meio das regras de inferência (A.28), (A.29) e (A.30), onde seria feita escolhas sobre qual operador t-norma e qual método de implicação usar. Ao avaliar um conjunto de regras a máquina de inferência deve considerar ainda o modo como irá manipular a inferência de todas as regras em conjunto. A referência [2] cita dois métodos:

i. Inferência baseada em composição, onde todas as regras são combinadas em uma relação para depois serem avaliadas como uma única regra.
ii. Inferência baseada em regras individuais, onde todas as regras serão primeiramente avaliadas para depois ser feita uma inferência baseada na composição de todos os resultados individuais.

Em ambos os casos as combinações, ou agregações como cita a referência [2], podem ser realizadas usando os operadores s-norma ou t-norma.

Duas máquinas de inferência e suas características são apresentadas a seguir baseadas no conjunto de regras (34).

$$Ru^{(l)} : \textbf{se } x_1 \text{ é } A_1^{\,l} \textbf{ e } x_2 \text{ é } A_2^{\,l} \textbf{ e } \ldots \textbf{ e } x_m \text{ é } A_m^{\,l}, \textbf{ então } y \text{ é } B^l \qquad \text{(A.35)}$$

. Máquina de Inferência Produto:

i. Inferência baseada em regras individuais / combinação s-norma
ii. Implicação Mamdani Produto(24)
iii. Operador produto algébrico (17) para todas as t-normas e max (11) para todas s-normas

Resultado: $u_{B'}(y) = \max_{l=1}^{M}[\sup_{x\in U}(u_{A'}(x)\prod_{i=1}^{n} u_{A_i^l}(x_i)u_{B^l}(y))]$ (A.36)

.Máquina de Inferência Mamdani:

i. Inferência baseada em regras individuais / combinação s-norma
ii. Implicação Mamdani Mínimo (A.23)
iii. Operador min (15) para todas t-normas e max (11) para todas s-normas

Resultado: $u_{B'}(y) = \max_{l=1}^{M}[\sup_{x\in U}\min(u_{A'}(x), u_{A_1^l}(x_1),\ldots, u_{A_n^l}(x_n), u_{B^l}(y))]$ (A.37)

Um exemplo gráfico que representa o processo de inferência de uma Máquina de Inferência Mamdani (36) é ilustrado na figura 5**,** adaptada de [3].

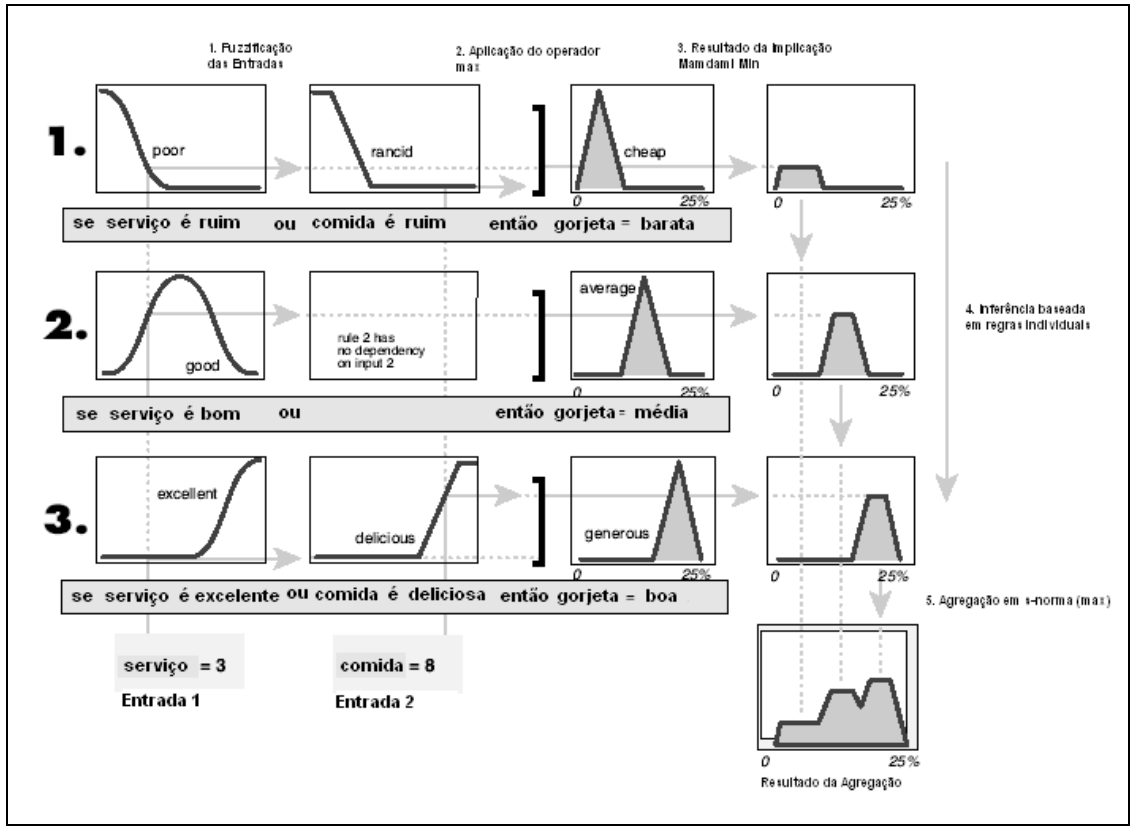

Figura 69: Atuação de uma Máquina de Inferência Mamdani

Defuzzificação

A defuzzificação em um sistema fuzzy consiste no mapeamento de um conjunto fuzzy *B'* em $V \subset R$ em um número real $y^* \in V$. Ou seja, a defuzzificação tem o papel de definir uma saída numérica que melhor represente o conjunto fuzzy resultante máquina de inferência.

A seguir são apresentados 3 defuzzificadores propostos em [2]

→ Defuzzificador por Centro de Gravidade:

Usa como saída da defuzzificação o centro de gravidade da área formada pelo conjunto fuzzy *B'*, calculado como:

$$y^* = \frac{\int_V y \cdot u_{B'}(y)dy}{\int_V u_{B'}(y)dy} \qquad \text{(A.38)}$$

A representação deste defuzzificador é mostrada na figura 70.

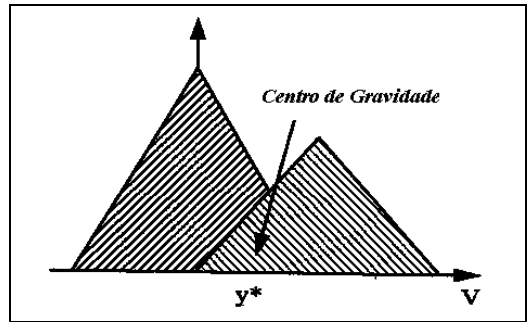

figura 70 : Defuzzificador Centro de Gravidade

→ Defuzzificador por Centro Médio:

A saída deste defuzzificador é a média dos centros dos conjuntos fuzzy que formam a saída da máquina de inferência ponderada pelas alturas dos mesmos. O resultado dessa operação é dado por:

$$y^* = \frac{\sum_{l=1}^{M} \bar{y}^l \cdot w_l}{\sum_{l=1}^{M} w_l} \qquad \text{(A.39)}$$

, onde $l = 1,2,...,M$ é o índice para o conjunto de regras e w_l as respectivas alturas dos centros $\bar{y}^l$.
A representação deste defuzzificador é mostrada na figura 71.

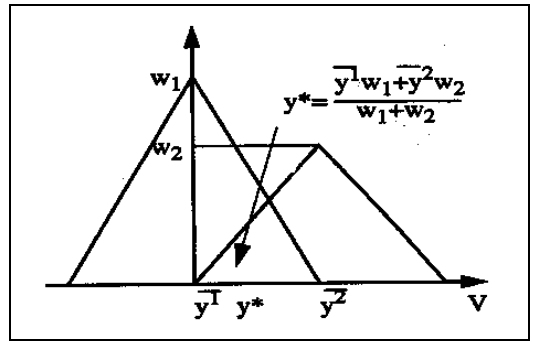

figura 71 : Defuzzificar Centro Médio

→ Defuzzificador Máximo:

O nome deste defuzzificador já apresenta seu principio. O valor calculado para saída será o valor máximo do conjunto fuzzy apresentado na saída da máquina de inferência fuzzy.

Porém, existem casos onde há mais de um valor máximo e para tanto são definidas maneiras de escolher um valor máximo dentre os disponíveis. No caso, algumas dessas maneiras são: menor do máximo – escolhe o menor valor de *y* que está associado a um valor máximo de *B'*; maior do máximo – escolhe o maior valor de *y* que está associado a um valor máximo de *B'*; média do máximo – calcula a média entre os valores de *y* que estão associados a um valor máximo de *B'*. A representação deste defuzzificador é mostrada na figura 72.

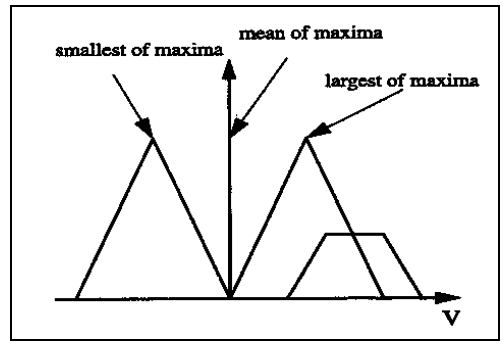

figura 72 : Defuzzificador Máximo

6.2 Programas da simulação

Aqui é apresentado o código fonte usado na implementação do controlador fuzzy com 16 regras na **simulação 3 e 4**, junto com o código da implementação da simulação. O restante dos programas desenvolvidos estão incluídos em []. Assim, os arquivos apresentados são:

6.2.1 CF16_Declaracao.m

```
% Declaracao da Base de regras e variaveis globais do Controlador Fuzzy

% regras - Configuracao das Saidas
global Output
Output = [0 0 ;0 0 ];
Output(:,:,2)=0;
Output(:,:,:,2)=0;


% Indice usado para identificacao das regras
global POS ZOE NEG
 POS = 1;
 ZOE = 2;
 NEG = 3;

% Valores singletons para defuzzificacao
NA = -100*4;
NMA = -75*2;
NMB = -50*1;
NB = -25*1;
ZE = 0;
PA = 100*4;
PMA = 75*2;
PMB = 50*1;
PB = 25*1;


%%%%%%%%%%% DEFINICAO DAS REGRAS %%%%%%%

%%%%%%%%%%% ANGULO POSITIVO %%%%%%%%%%%%

Output(POS,POS,POS,POS) = NMB-60;
Output(POS,POS,POS,NEG) = NA-100;
Output(POS,POS,NEG,POS) = ZE ;
Output(POS,POS,NEG,NEG) = PB + 50;
Output(POS,NEG,POS,POS) = ZE;
Output(POS,NEG,POS,NEG) = NMA-50;
Output(POS,NEG,NEG,POS) = PB;
Output(POS,NEG,NEG,NEG) = PB;

%%%%%%%%%%% ANGULO NEGATIVO %%%%%%%%%%%%

Output(NEG,NEG,NEG,NEG) = PMB+60;
Output(NEG,NEG,NEG,POS) = PA+100;
```

```
Output(NEG,NEG,POS,NEG) = ZE ;
Output(NEG,NEG,POS,POS) = NB - 50;
Output(NEG,POS,NEG,NEG) = ZE;
Output(NEG,POS,NEG,POS) = PMA+50;
Output(NEG,POS,POS,NEG) = NB;
Output(NEG,POS,POS,POS) = NB;

% Indice de Regras ativadas
global indiceAngulo1 indiceAngulo2 indiceVelAng1 indiceVelAng2 indiceErro1
global indiceErro2 indiceVelErro1 indiceVelErro2

% Armazena Valores de Pertinencia das Variaveis Erro,
% VelErro, Angulo e VelAngulo global erroPosDV velErroDV anguloDV velAnguloDV

% Armazena valores pre-processados para escala [-100,100]
global angInput erroPosInput velAngInput velPosInput
```

6.2.2 CF16_PreProFuzzificacao.m

```
function CF16_Fuzzificacao(angulo,velAngulo,erroPos,velPos)

% Variaveis globais usadas pelas funções:
% CF16_DeclaracaoDefault e CF16_InfeDefuzzificacao
global POS_ESQ POS_CE POS_DIR ZE_ESQ ZE_CE ZE_DIR NEG_ESQ NEG_CE NEG_DIR
global POS ZOE NEG
global indiceAngulo1 indiceAngulo2 indiceVelAng1 indiceVelAng2 indiceErro1
global indiceErro2 indiceVelErro1 indiceVelErro2
global erroPosDV velErroDV anguloDV velAnguloDV
global angInput erroPosInput velAngInput velPosInput
% Ganhos das entradas a serem ajustados apos o uso de CF16_Declaracao
global G1 G2 G3 G4

angInput = ((angulo*(100/512))-100)*G3;
if (angInput > 100) angInput = 100; end
if (angInput <-100) angInput = -100; end

erroPosInput = ((erroPos*(100/1024)))*G1;
if (erroPosInput > 100) erroPosInput = 100; end
if (erroPosInput <-100) erroPosInput = -100; end

velAngInput = ((velAngulo*(100/512))-100)*G4;
if (velAngInput > 100) velAngInput = 100; end
if (velAngInput <-100) velAngInput = -100; end

velPosInput = (velPos*(100/512))*G2;
if (velPosInput > 100) velPosInput = 100; end
if (velPosInput <-100) velPosInput = -100; end

% Fuzzificacao das Inputs:

%% Fuzzificacao do Angulo

anguloDV(NEG) = (-1/200)*angInput + 0.5;
anguloDV(POS) = (1/200)*angInput + 0.5;
indiceAngulo1 = NEG; indiceAngulo2 = POS;

%% Fuzzificacao da Velocidade do Angulo
```

```
velAnguloDV(NEG) = (-1/200)*velAngInput + 0.5;
velAnguloDV(POS) = (1/200)*velAngInput + 0.5;
indiceVelAng1 = NEG; indiceVelAng2 = POS;

%% Fuzzificacao do erro da Posicao da Bolinha
erroPosDV(NEG) = (-1/200)*erroPosInput + 0.5;
erroPosDV(POS) = (1/200)*erroPosInput + 0.5;
indiceErro1 = NEG; indiceErro2 = POS;

%% Fuzzificacao da velocidade do erro da Posicao da Bolinha
velErroDV(NEG) = (-1/200)*velPosInput + 0.5;
velErroDV(POS) = (1/200)*velPosInput + 0.5;
indiceVelErro1 = NEG; indiceVelErro2 = POS;
```

6.2.3 CF16_InfeDefuzzificacao.m

```
function CONTROLE = CF16_InfeDefuzzificacao();

% Funcao que realiza inferencia e defuzzificacao usando o metodo COGS

global Output
global POS ZOE NEG
global indiceAngulo1 indiceAngulo2 indiceVelAng1 indiceVelAng2 indiceErro1
global indiceErro2 indiceVelErro1 indiceVelErro2
global erroPosDV velErroDV anguloDV velAnguloDV

% Inferência + Defuzzificacao

% Valores de Implicacao das Regras;
CONTROLE = 0;
Imp1 =
anguloDV(indiceAngulo1)*velAnguloDV(indiceVelAng1)*erroPosDV(indiceErro1)*velErroDV(indiceVelErro1)
Imp2 =
anguloDV(indiceAngulo1)*velAnguloDV(indiceVelAng1)*erroPosDV(indiceErro1)*velErroDV(indiceVelErro2)
Imp3 =
anguloDV(indiceAngulo1)*velAnguloDV(indiceVelAng1)*erroPosDV(indiceErro2)*velErroDV(indiceVelErro1)
Imp4 =
anguloDV(indiceAngulo1)*velAnguloDV(indiceVelAng1)*erroPosDV(indiceErro2)*velErroDV(indiceVelErro2)
Imp5 =
anguloDV(indiceAngulo1)*velAnguloDV(indiceVelAng2)*erroPosDV(indiceErro1)*velErroDV(indiceVelErro1)
Imp6 =
anguloDV(indiceAngulo1)*velAnguloDV(indiceVelAng2)*erroPosDV(indiceErro1)*velErroDV(indiceVelErro2)
Imp7 =
anguloDV(indiceAngulo1)*velAnguloDV(indiceVelAng2)*erroPosDV(indiceErro2)*velErroDV(indiceVelErro1)
Imp8 =
anguloDV(indiceAngulo1)*velAnguloDV(indiceVelAng2)*erroPosDV(indiceErro2)*velErroDV(indiceVelErro2)
Imp9 =
anguloDV(indiceAngulo2)*velAnguloDV(indiceVelAng1)*erroPosDV(indiceErro1)*velErroDV(indiceVelErro1)
Imp10 =
anguloDV(indiceAngulo2)*velAnguloDV(indiceVelAng1)*erroPosDV(indiceErro1)*velErroDV(indiceVelErro2)
Imp11 =
anguloDV(indiceAngulo2)*velAnguloDV(indiceVelAng1)*erroPosDV(indiceErro2)*velErroDV(indiceVelErro1)
Imp12 =
anguloDV(indiceAngulo2)*velAnguloDV(indiceVelAng1)*erroPosDV(indiceErro2)*velErroDV(indiceVelErro2)
Imp13 =
anguloDV(indiceAngulo2)*velAnguloDV(indiceVelAng2)*erroPosDV(indiceErro1)*velErroDV(indiceVelErro1)
Imp14 =
anguloDV(indiceAngulo2)*velAnguloDV(indiceVelAng2)*erroPosDV(indiceErro1)*velErroDV(indiceVelErro2)
Imp15 =
anguloDV(indiceAngulo2)*velAnguloDV(indiceVelAng2)*erroPosDV(indiceErro2)*velErroDV(indiceVelErro1)
Imp16 =
anguloDV(indiceAngulo2)*velAnguloDV(indiceVelAng2)*erroPosDV(indiceErro2)*velErroDV(indiceVelErro2)

CONTROLE =            Imp1*Output(indiceAngulo1,indiceVelAng1,indiceErro1,indiceVelErro1);
```

```
CONTROLE = CONTROLE + Imp2*Output(indiceAngulo1,indiceVelAng1,indiceErro1,indiceVelErro2);
CONTROLE = CONTROLE + Imp3*Output(indiceAngulo1,indiceVelAng1,indiceErro2,indiceVelErro1);
CONTROLE = CONTROLE + Imp4*Output(indiceAngulo1,indiceVelAng1,indiceErro2,indiceVelErro2);
CONTROLE = CONTROLE + Imp5*Output(indiceAngulo1,indiceVelAng2,indiceErro1,indiceVelErro1);
CONTROLE = CONTROLE + Imp6*Output(indiceAngulo1,indiceVelAng2,indiceErro1,indiceVelErro2);
CONTROLE = CONTROLE + Imp7*Output(indiceAngulo1,indiceVelAng2,indiceErro2,indiceVelErro1);
CONTROLE = CONTROLE + Imp8*Output(indiceAngulo1,indiceVelAng2,indiceErro2,indiceVelErro2);
CONTROLE = CONTROLE + Imp9*Output(indiceAngulo2,indiceVelAng1,indiceErro1,indiceVelErro1);
CONTROLE = CONTROLE + Imp10*Output(indiceAngulo2,indiceVelAng1,indiceErro1,indiceVelErro2);
CONTROLE = CONTROLE + Imp11*Output(indiceAngulo2,indiceVelAng1,indiceErro2,indiceVelErro1);
CONTROLE = CONTROLE + Imp12*Output(indiceAngulo2,indiceVelAng1,indiceErro2,indiceVelErro2);
CONTROLE = CONTROLE + Imp13*Output(indiceAngulo2,indiceVelAng2,indiceErro1,indiceVelErro1);
CONTROLE = CONTROLE + Imp14*Output(indiceAngulo2,indiceVelAng2,indiceErro1,indiceVelErro2);
CONTROLE = CONTROLE + Imp15*Output(indiceAngulo2,indiceVelAng2,indiceErro2,indiceVelErro1);
CONTROLE = CONTROLE + Imp16*Output(indiceAngulo2,indiceVelAng2,indiceErro2,indiceVelErro2);
CONTROLE = CONTROLE/(Imp1 + Imp2 + Imp3 + Imp4 + Imp5 + Imp6 + Imp7 + Imp8 + Imp9 + Imp10 + Imp11 +
Imp12 + Imp13 + Imp14 + Imp15 + Imp16);
```

6.2.4 Simulacao3e4_Barra_Esfera.m

```
%%%%%%%%%%%% SCRIPT PARA SIMULACAO DO SISTEMA BARRA ESFERA %%%%%%%%%%%%
%
% Universidade de Brasili, UnB
% Faculdade de Tecnologia
% Departamento de Engenharia Eletrica, ENE
% Autor: Joao Miguel Ferreira Cunha
%
%
% Este script contem a implementacao da simulacao 3 e 4,
% ou seja,ela contem a simulacao do sistema barra-esfera
% com a inclusao do motor na malha de controle, simulacao de ruidos,
% uso do filtro digital e uso do canal integral
%
% Arquivos usados na simulacao>
%  > Controladores fuzzy:
%    - CF16_Declaracao.m         | CF81_Declaracao.m
%    - CF16_PreProFuzzificacao.m | CF81_PreProFuzzificacao.m
%    - CF16_InfeDefuzzificacao.m | CF81_InfeDefuzzificacao.m
%    - FuncoesPertinencia16PrePro.m | FuncoesPertinencia81PrePro.m
%    - SuperficieDeControleCF16FINAL.m | SuperficieDeControleCF81FINAL.m
%
%  > Animacao Grafica:
%     - iniciaAnimacao.m
%     - plotBeam(angulo).m
%     - plotBall(angulo,100*posicao).m
%     - [V3x,V3y,V4x,V4y] = plotManivelaBeam(angulo);
%     - [V1x,V1y,V2x,V2y] = plotManivelaDisco(angulo);
%     - plotManivela([V1x V1y V2x V2y],[V3x V3y V4x V4y])
%
%
%
% Para simular o sistema as seguintes escolhas devem ser feitas:
%
% 1) Qual controlador usar ? Vide a variavel flagControle
% 2) Deseja incluir ruidos ? Vide a variavel flagRuidos
% 3) Deseja mudar a ordem do ruido ?
% Vide as variaveis ordemRuidoPosicao e ordemRuidoAngulo
% 4) Deseja acrescentar acao integral ? Vide a variavel flagIntegral
% 5) Deseja acrescentar filtro digital ? Vide a variavel flagFiltro
% 6) Para mudar outros parametros da simulacao como periodo de amostragem,
```

```
% duracao da simulacao e frequencia do filtro digital vide a secao:
% DEFINICAO DE PARAMETROS DA SIMULACAO
% 7) A sintonia dos ganhos eh feita alterando os valores de G1, G2, G3, G4 e G5
% 8) Para ver as regras implementadas, abra os arquivos CF16_Declaracao.m ou
% CF81_Declaracao.m

clear all
% Variaveis globais que serao usadas pelas funcoes de animacao
global Raio L E RaioManivela RaioDisco LMan

%%%%%%%%%% PARAMETROS FISICOS DO SISTEMA %%%%%%%%%%%%%%%%%%%%%%%%%%
m = 0.028;              % massa da Bola - 0.006kg | 0.017kg | 0.028kg
G = -9.81;              % aceleracao da gravidade [m/s^2]
Raio = 0.015;           % Raio da Bolinha  [m]
L = 0.40;               % Comprimento da pista [m]
d = 0.13;               % Comprimento da Manivela [m]
J = 2*m*Raio^2/3;       % Momento de Inercia da Bola = 2mR^2/3
K = m*G/((J/(Raio^2))+m);     %Constante usada no modelo de Espaco de Estados
M = 0.169;              % Massa da Pista [kg]
% Espessura da Pista [m] - Considerada para um calculo aproximado do momento de
inercia da barra
E = 0.004;
B1 = m/((J/Raio^2)+m);
Jbeam = M*(L^2+E^2)/12 + M*0.20; % Momento de Inercia da Pista

RaioManivela = 3;       % Parametro para animacao grafica
RaioDisco = 4;          % Raio do disco da manivela
LMan = 12;              % Comprimento da manivela

n = 8*6.3;              % Relacao das engrenagens
Jm = 9.87e-7;           % Momento de Inercia do rotor do motor
Jl = 2.5e-7;            % Momento de Inercia da carga(engrenagem) vista pelo
eixo de saida
Jo = Jm+Jl/n ;          % Momento de Inercia do conjunto visto pelo eixo do
motor
bo = 1.03e-6;           % Coeficiente de atrito - Catalogo PITTMAN
Ra = 12;                % Resistencia do motor  - Catalogo PITTMAN
La = 6e-3;              % Indutancia do motor   - Catalogo PITTMAN
K2 = 27.4e-3;           % Constante de torque do motor - Catalogo PITTMAN
K3 = 27.4e-3;           % Constante que relaciona tensao induzida pela bobina
do motor - Catalogo PITTMAN

%%%%%%% CONSTANTES PARA O ESPACO DE ESTADOS APROXIMADO PARA O MOTOR
K4 = K2/n;                %       Theta          K4
a = (Jo*Ra);              %       -----  =  ------------
b = (bo*Ra +K2*K3);       %       Volts      a*s^2 + b*s

% Definicao do Espaco de Estados para modelo do motor sem desprezar a
indutancia
%Amotor = [0 1 0; 0 -bo/Jo (K2)/(Jo*n); 0 (-K3*n)/La -Ra/La];
%Bmotor = [0;0;1/La];
%Cmotor = [1 0 0];

% Definicao do Espaco de Estados para o modelo do motor desprezando a
indutancia
Amotor = [0 1; 0 -b/a];
Bmotor = [0 K4/a];
```

```
Cmotor = [1 0];

% Definicao do Espaco de Estados do sistema Barra-Esfera segundo a equacao
linearizada:
% 0 = (J/R^2 + m)*rDotDot + m*g*sin(ang) - m*r*(angDot)^2, sendo linearizado em
torno de ang= 0 -> 0 = (J/R^2+m)*rDotDot + m*g*ang
A = [0 1 0 0 ; 0 0 K 0  ; 0 0 0 1  ;0 0 0 0  ];
B = [0;0;0;1];
C = [1 0 0 0];
D = 0;

%%%%%%%%% DEFINICAO DE PARAMETROS PARA A SIMULACAO %%%%%%%%%%%%%%


Tcontinuo = 0.03;                   % Periodo de amostragem
freq = 1;                           % Frequencia do filtro digital
duracao = 40;                       % duracao da simulacao em segundos
xdot = zeros(size(0:Tcontinuo:duracao));   % inicializacao do vetor que
representa as variaveis de estado durante a simulacao
xdot = [xdot;xdot;xdot;xdot]';
x = xdot;   % vetor que ira armazenar as variaveis de estado durante a
simulacao
Uss = x(:,1);            % Sinal de controle desenvolvido durante a simulacao
y=x;        % Valores que representam a conversao A/D das variaveis de estados
y(:,5) = xdot(:,1);                 % Coluna onde eh armazenado o erro
thetaDot = [x(:,1) x(:,2)];    % Inicializacao das variaveis de estados do
motor
theta = thetaDot;             % Inicializacao das variaveis de estados do motor
i = 2;                              % Contador usado durante a simulacao
ref = 0.10;               % valor de referencia para a posicao da bolinha em
[m]
ref1 = ref;            % primeiro valor de referencia para a posicao da bolinha
ref2 = 0.3;            % segundo valor de referencia -> Simulando onda quadrada
REF = x(:,1);          % Variavel onde eh armazenada referencia para plotagem
x(1,:) = [0.4 0 0 0];               % Condicao inicial do sistema
flagControlador = 0;  % Define qual controlador vc quer simular ( 1 = 81
regras, 0 = 16 regras)

% Declaracao que inicia variaveis glovais do controlador fuzzy - onde eh
definida base de regras

if (flagControlador == 0)
CF16_Declaracao
end
if (flagControlador == 1)
CF81_Declaracao
end
global G1 G2 G3 G4

% Ganhos para sintonizacao do controlador fuzzy com 16 regras
if (flagControlador == 0)
GUss = 1/4 ;        % Ajuste do ganho de saida do controlador fuzzy
G1 = 0.5; G2 = 30;  % G1 = Ganho para o erro da pos.| G2 = Ganho para vel.pos
G3 = 1.3;  G4 = 0;  % G3 = ganho para ângulo  | G4 = ganho para vel.angulo
end

% Ganhos para sintonizacao do controlador fuzzy com 81 regras
```

```
if (flagControlador == 1)
GUss = 1 ;      % Ajuste do ganho de saida do controlador fuzzy
G1 = 0.4; G2 = 58;
G3 = 1; G4 = 0;
end


%%%%%%%%%%%%%%%% INICIANDO PRIMEIRO QUADRO DA PLOTAGEM %%%%%%%%%%%%%%%%%%%%%%%%
iniciaAnimacao

%%%%%%%%%%%%%%%%%%%%%%%% INICIO DA SIMULACAO %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if (flagControlador == 0)
% Plota funcoes de pertinencia do sistema
FuncoesPertinencia16PrePro(G1,G2,G3,G4);
end
if (flagControlador == 1)
% Plota funcoes de pertinencia do sistema
FuncoesPertinencia81PrePro(G1,G2,G3,G4);
end

%%% Flags que conduzem a simulação
% Flag que insere ruido na simulacao (1 = ruido ligado, 0 = ruido desligado)
flagRuido = 0 ;
% Flag que liga acao do filtro digital (1 = filtro ligado, 0 = filtro
desligado)
flagFiltro = 0;
% Flag que liga canal integral (1 = integral ligado, 0 = integral desligado)
flagIntegral = 0;
Ti = 0.0045;          % Ganho integral
uInte = 0;            % Inicial valor do integral
% Magnitude do ruido em m( 40cm --> 1024 assim como 0.4 cm -->10 bits)

ordemRuidoPosicao = 0.004;
ordemRuidoAngulo = 0.03*pi/180;  % Magnitude do ruido em graus
% Flags que controlam plotagem indicadas - Variaveis de estado e plotagem de
fase
plotEstados = 1; plotPhase = 0;
% Sinal de controle e superficie de controle
plotControle = 1; plotSuperficie = 0;
%%%%%%%%%%%%%%%%%%%%%%%% INICIO DA SIMULACAO %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for j=0:Tcontinuo:duracao


     % mudanca do sinal de referencia
     if (j >(duracao)/4)
         ref = ref2;
     end
     if (j>(duracao/2))
         ref = ref1;
     end
     if (j>(duracao*3/4))
         ref = ref2;
     end
     REF(i) =  ref;

% Simulando conversao A/D do microprocessador
```

```
% As variaveis  r       rDot,   phi    phiDot sao representadas respectivamente
por:
%               y(:,1)  y(:,2)  y(:,3) y(:,4)

% O universode de conversao para r e phi sao definidos para a simulacao de
acordo com
% as especificacoes do modelo fisico : r = [-0.087,0.087](rad) , phi = [-
0.087,0.087](rad)

% As conversoes sao feitas da seguinte forma
%  Posicao = [0.00, 0.40]cm  -> [0,1023] -> PreProcessado e Fuzzificado
% vel.Erro = [-0.64,0.64]m/s -> [0,1023] -> PreProcessado e Fuzzificado
(valores observados na simulacao feita para controlador em Espaco de Estados)
%      Ang = [-.087,.087]rad -> [0,1023] -> PreProcessado e Fuzzificado
%  vel.Ang = [-5.00,5.00]rad/s -> [0,1023] -> PreProcessado e Fuzzificado
(valores observados na simulacao feita para controlador em Espaco de Estados)

% PreProcessamento do erro da posicao
    y(i-1,1) = (x(i-1,1)+ordemRuidoPosicao*rand(1)*flagRuido)*(1024/0.4);
    y11 = y(i-1,1);
    erro = ref*1024/0.4 - y(i-1,1);
    if (i == 2)
        uInte_ant = uInte;
    end
% calculo do sinal de controle integral
    uIntegral = uInte_ant - erro*Ti*Tcontinuo;
    if (erro>1024)  erro = 1024; erroSaturacao = 'Ok';
    end
    if (erro<-1024)  erro = -1024;
    end
    y1 = y(i-1,1);

% PreProcessamento da velocidade do erro da posicao
    if (i==2)     % verifica se esta no primeiro laco da simulacao
        y(i-1,2) =(y(i-1,1) - y(i-1,1));
    else
        y(i-1,2) = (y(i-1,1)-y(i-2,1));
    end
    if ( y(i-1,2) >1024 ) y(i-1,2) = 1024; velErroSaturacao = 'Ok';
    end
    if ( y(i-1,2) <-1024) y(i-1,2) = -1024;
    end
    y2 = y(i-1,2);

% PreProcessamento do angulo
    y(i-1,3) = (x(i-1,3)+(ordemRuidoAngulo)*rand(1)*flagRuido)*(1024/0.174) +
512;
    if (y(i-1,3)>1024) y(i-1,3) = 1024; anguloSaturacao = 'Ok';
    end
    if (y(i-1,3)<0) y(i-1,3) = 0;
    end
     y3 = y(i-1,3);

% PreProcessamento da velocidade do angulo
    y(i-1,4) = x(i-1,4)*(1024/10) + 512;
    y4 = y(i-1,4);
    if (y(i-1,4)>1024) y(i-1,4) = 1024; velAnguloSaturacao = 'Ok';
```

```
    end
    if (y(i-1,4)<0) y(i-1,4) = 0;
    end
     y4 = y(i-1,4);

% Chama o processo de PreProcessamento + Fuzzificacao: escolha entre
controlador fuzzy com 16 oou 81 regras
if (flagControlador == 0)
   CF16_PreProFuzzificacao(y(i-1,3),y(i-1,4), erro,y(i-1,2));
end
if (flagControlador == 1)
   CF81_PreProFuzzificacao(y(i-1,3),y(i-1,4), erro,y(i-1,2));
end

% Chama o processo de Inferencia e Defuzzificacao: escolha entre controlador
fuzzy com 16 oou 81 regras
if (flagControlador == 0)
    CONTROLE = CF16_InfeDefuzzificacao;
end
if (flagControlador == 1)
    CONTROLE = CF81_InfeDefuzzificacao;
end

% Escalonando o sinal de saida de acordo com os limites de tensao sobre o motor
e de acordo com a escala de saida do controlador fuzzy
    Uss(i) = CONTROLE*(15/100)*GUss;

 % Verifica se canal integral deve ser usado
    if (flagIntegral == 1)
        Uss(i) = Uss(i) + uIntegral;
    end

%  Verifica se filtro digital deve ser usado
    if (flagFiltro == 1)
       if (i == 2)
           CONTROLE_ANT = 0;
       end
   %   Uss(i) = (Uss(i)*Tcontinuo*2*pi*freq) +  CONTROLE_ANT*(1-
Tcontinuo*2*pi*freq); % Foward Rectangle Rule
       Uss(i) = (Uss(i)*Tcontinuo*2*pi*freq/(1+Tcontinuo*2*pi*freq) +
CONTROLE_ANT/(1+Tcontinuo*2*pi*freq)); % Backward Rectangle Rule
       CONTROLE_ANT = Uss(i);
    end

 % Calculo dos estados do motor DC
    thetaDot(i,1) = Amotor(1,:)*theta(i-1,:)' + Bmotor(1)*Uss(i);
    thetaDot(i,2) = Amotor(2,:)*theta(i-1,:)' + Bmotor(2)*Uss(i);

    theta(i,2) = thetaDot(i,2)*Tcontinuo + theta(i-1,2);
    theta(i,1) = thetaDot(i,1)*Tcontinuo + theta(i-1,1);


% Calculo que relaciona o angulo de saida do motor DC com o angulo da
barra(Beam). Esse calculo foi baseado numa aproximacao obtida pelo script
PlotExcursaoPosicaoDeslocada.m
    anguloBeam = - 0.014*(theta(i,1))^3 - 0.0054*(theta(i,1))^2 +
0.099*(theta(i,1)) + 0.0011;
```

```
% derivacao para calcular velocidade angular da barra
    anguloBeamDot = (anguloBeam - x(i-1,3))/Tcontinuo;
% derivacao para calcular aceleracao angular da barra
    anguloBeamDotDot = (anguloBeamDot - x(i-1,4))/Tcontinuo;
% Calculo das variaveis de estados segundo Espaco de Estados
Linearizado(A,B,C,D)
%    xdot(i,1) = A(1,:)*x(i-1,:)';
%    xdot(i,2) = A(2,:)*x(i-1,:)';
%    xdot(i,3) = A(3,:)*x(i-1,:)';
%    xdot(i,4) = A(4,:)*x(i-1,:)' + anguloBeamDotDot; %Uss(i);

% Calculo das variaveis de estado segundo Modelo Nao-Linear
    xdot(i,1) = x(i-1,2);
    xdot(i,2) = B1*(x(i-1,1)*x(i-1,4)^2 + G*sin(x(i-1,3)));
    xdot(i,3) = x(i-1,4);
    xdot(i,4) = 0 + anguloBeamDotDot;

% Integracao dos valores para obtencao de r, rDot, ang e angDot
    x(i,4) = (xdot(i,4))*Tcontinuo + x(i-1,4);
    x(i,3) = (x(i,4))*Tcontinuo + x(i-1,3);
    x(i,2) = (xdot(i,2))*Tcontinuo + x(i-1,2);
    x(i,1) = (xdot(i,1))*Tcontinuo + x(i-1,1);


%----- Plotagem do estados do sistemas na animacao ---------------
    for j=1:0.0001:5, %0.00016
    end

    plotBeam(x(i,3))
    plotBall(x(i,3),100*x(i,1))
    [V3x,V3y,V4x,V4y] = plotManivelaBeam(x(i,3));
    [V1x,V1y,V2x,V2y] = plotManivelaDisco(x(i,3));
    plotManivela([V1x V1y V2x V2y],[V3x V3y V4x V4y])

    i = i+1;

end;
x(1,:)=[];
Uss(1,:)=[];
REF(1,:) = [];

%%%%%%%%%%%%%% ROTINAS DE PLOTAGEM DE GRAFICOS %%%%%%%%%%%%%%%%
if (plotEstados==1)
set(figure,'Position',[scnsize(3)/2+8,20,scnsize(3)/2-10,scnsize(4)/2-30],...
    'NumberTitle','off','Name','Variaveis de Estado')
subplot(2,2,1);plot([0:Tcontinuo:duracao],x(:,1),'LineWidth',2);
hold on
plot([0:Tcontinuo:duracao],REF,'Color','m');
title('Posicao vs Tempo');
xlabel('Tempo[s]'); ylabel('Posicao[m]');
axis tight;
%hold on
%figure

subplot(2,2,2);plot([0:Tcontinuo:duracao],x(:,2),'LineWidth',2);
title('Velocidade Da Posicao vs Tempo');
xlabel('Tempo[s]'); ylabel('Vel.Posicao [m/s]');
```

```
axis tight;
%hold on
%figure

subplot(2,2,3);plot([0:Tcontinuo:duracao],x(:,3),'LineWidth',2);
title('Angulo vs Tempo');
xlabel('Tempo[s]'); ylabel('Angulo [rad]');
axis tight;
%hold on
%figure

subplot(2,2,4);plot([0:Tcontinuo:duracao],x(:,4),'LineWidth',2);
title('Velocidade Do Angulo vs Tempo');
xlabel('Tempo[s]'); ylabel('Vel.Angulo [rad/s]');
axis tight;
hold on
end
if (plotControle ==1)

set(figure,'Position',[scnsize(3)/2+8,scnsize(4)/2+75,scnsize(3)/2-
10,scnsize(4)/2-155],...
    'NumberTitle','off','Name','Sinal de Controle [rad/s^2]')
plot([0:Tcontinuo:duracao],Uss,'LineWidth',2);
title('Controle vs Tempo');
xlabel('Tempo[s]'); ylabel('Controle [V]');
axis tight;
hold on
end

%%%%%%%%%%%%% Phase Plots %%%%%%%%%%%%%%%%%%%%

if (plotPhase == 1)
set(figure,'Position',[8,scnsize(4)/2+75,scnsize(3)/2-10,scnsize(4)/2-155],...
    'NumberTitle','off','Name','Phase Plot')
subplot(1,2,1);plot(x(:,1),x(:,2),'LineWidth',2);
%plot(x(:,1),x(:,2));
title('Posicao vs Vel.Posicao');
xlabel('Posicao [m]'); ylabel('Vel.Posicao [m/s]');
axis([0,tempo,-0.1,0.5])
axis tight ;
hold on
%figure

subplot(1,2,2);plot(x(:,3),x(:,4),'LineWidth',2);
title('Velocidade Do Angulo vs Angulo');
xlabel('Angulo[rad]'); ylabel('Vel.Angulo [rad/s]');
axis tight;
hold on
end

% Plotagem da superficie de controle
if (plotSuperficie==1)
    if (flagControlador == 0)
SuperficieDeControleCF16FINAL
end
if (flagControlador == 1)
SuperficieDeControleCF81FINAL
```

```
end
end
```

6.3 Código fonte implementado no PIC18F252

```
#include <18f252.h>

#device adc=10

#use delay(clock=40000000)
#use rs232(baud = 19200, xmit = pin_c6, rcv = pin_c7)

#fuses H4,NOWDT,PUT,NOBROWNOUT,NOLVP,CCP2C1

#include <mod_lcd_p18_para.c>
#include <math.h>
#include <stdlib.h>
#include <usart18f252.c>      // Biblioteca de comunicação assincrona.


 unsigned long int tempo;
 float resto;
 unsigned long int TaDisplay=0,TaInterrupcao=0,ref=256,ref1=256,ref2=768;
 short int flagRef=0, flagIntegral,filtroPos,filtroControle;
 unsigned int16 aux, Ta;
 int amostras;
 char amostraString[4];
 int contadorEscala=0;
 char auxGanhos[6];

 signed int16 bitDoAngulo, bitDaPosicao,posicaoAnterior,
refDoAngulo,bitDaPosicaoAnterior;
 signed int Output[2][2][2][2];
 int POS =0, NEG = 1;
 int controlePWM1,controlePWM2;
 float erroPosDV[2],velErroDV[2],anguloDV[2],velAnguloDV[2];
 float angInput, erroPosInput, velAngInput, velPosInput;
 float CONTROLE,CONTROLE_ANT,erro,uIntegral,uIntegral_anterior,Ti,offset;
 float G1,G2,G3,G4,GU;
 float k1,k2,freq; // Parametros do filtro


#int_timer0 // Interrupçao ocorrendo com periodo de ajustavel
void trata_t0 ()

  {
 // Comandos usados para contar a duracao da interrupcao
      output_high(pin_b1);
      output_high(pin_b0);

       set_timer0(TaInterrupcao);
       set_adc_channel(0);
       delay_us(50);
       bitDoAngulo = read_adc();
```

```
     set_adc_channel(1);
     delay_us(50);
     bitDaPosicao = read_adc();

       if (filtroPos == 1) {
       bitDaPosicao = bitDaPosicaoAnterior*((float)k1) +
bitDaPosicao*((float)k2);
       bitDaPosicaoAnterior = bitDaPosicao;
       }

// PreProcessamento
      angInput =((((float)bitDoAngulo - (float)512.5)*((float)0.1953125)))*G3;

      erro = ((float)ref - (float)bitDaPosicao);
      erroPosInput = ((erro*((float)0.097751710654936461388074291300097 8)))*G1;

      if (flagIntegral == 1) {
      uIntegral = uIntegral_anterior - ((float)0.03)*Ti*erro;
      uIntegral_anterior = uIntegral;
      }
      velPosInput = (((float)bitDaPosicao-
(float)posicaoAnterior)*((float)0.1953125))*G2;
      posicaoAnterior = bitDaPosicao;

      velAngInput = 0; // valor desconsideradp

// Fuzzificacao do Angulo
      anguloDV[NEG] = ((float)-0.005)*angInput + (float)0.5;
      if (anguloDV[NEG] > 1.0) {
            anguloDV[NEG] = 1.0; }
      else if (anguloDV[NEG] < 0.0 ) {
            anguloDV[NEG] = 0; }
      anguloDV[POS] = (float)1.0 - anguloDV[NEG];

// Fuzzificacao da Velocidade do Angulo
      velAnguloDV[NEG] = ((float)-0.005)*velAngInput + (float)0.5;
      velAnguloDV[POS] = (float)1.0 - velAnguloDV[NEG];

// Fuzzificacao do erro da Posicao da Bolinha
      erroPosDV[NEG] = ((float)-0.005)*erroPosInput + (float)0.5;
      if (erroPosDV[NEG] > 1.0) {
            erroPosDV[NEG] = 1.0; }
      else if (erroPosDV[NEG] < 0.0) {
            erroPosDV[NEG] = 0.0; }
      erroPosDV[POS] = (float)1.0 - erroPosDV[NEG];

// Fuzzificacao da velocidade da Posicao da Bolinha
      velErroDV[NEG] = ((float)-0.005)*velPosInput + (float)0.5;
      if (velErroDV[NEG] > 1.0) {
            velErroDV[NEG] = 1.0; }
      else if (velErroDV[NEG] < 0.0) {
            velErroDV[NEG] = 0; }
      velErroDV[POS] = (float)1.0 - velErroDV[NEG];
// INFERENCIA E DEFUZZIFICACAO
```

```
CONTROLE =
(float)anguloDV[POS]*(float)velAnguloDV[POS]*(float)erroPosDV[POS]*(float)velEr
roDV[POS]*((float)Output[POS][POS][POS][POS]);
CONTROLE = (float)CONTROLE +
(float)anguloDV[POS]*(float)velAnguloDV[POS]*(float)erroPosDV[POS]*(float)velEr
roDV[NEG]*((float)Output[POS][POS][POS][NEG]);
CONTROLE = (float)CONTROLE +
(float)anguloDV[POS]*(float)velAnguloDV[POS]*(float)erroPosDV[NEG]*(float)velEr
roDV[POS]*((float)Output[POS][POS][NEG][POS]);
CONTROLE = (float)CONTROLE +
(float)anguloDV[POS]*(float)velAnguloDV[POS]*(float)erroPosDV[NEG]*(float)velEr
roDV[NEG]*((float)Output[POS][POS][NEG][NEG]);
CONTROLE = (float)CONTROLE +
(float)anguloDV[POS]*(float)velAnguloDV[NEG]*(float)erroPosDV[POS]*(float)velEr
roDV[POS]*((float)Output[POS][NEG][POS][POS]);
CONTROLE = (float)CONTROLE +
(float)anguloDV[POS]*(float)velAnguloDV[NEG]*(float)erroPosDV[POS]*(float)velEr
roDV[NEG]*((float)Output[POS][NEG][POS][NEG]);
CONTROLE = (float)CONTROLE +
(float)anguloDV[POS]*(float)velAnguloDV[NEG]*(float)erroPosDV[NEG]*(float)velEr
roDV[POS]*((float)Output[POS][NEG][NEG][POS]);
CONTROLE = (float)CONTROLE +
(float)anguloDV[POS]*(float)velAnguloDV[NEG]*(float)erroPosDV[NEG]*(float)velEr
roDV[NEG]*((float)Output[POS][NEG][NEG][NEG]);
CONTROLE = (float)CONTROLE +
(float)anguloDV[NEG]*(float)velAnguloDV[POS]*(float)erroPosDV[POS]*(float)velEr
roDV[POS]*((float)Output[NEG][POS][POS][POS]);
CONTROLE = (float)CONTROLE +
(float)anguloDV[NEG]*(float)velAnguloDV[POS]*(float)erroPosDV[POS]*(float)velEr
roDV[NEG]*((float)Output[NEG][POS][POS][NEG]);
CONTROLE = (float)CONTROLE +
(float)anguloDV[NEG]*(float)velAnguloDV[POS]*(float)erroPosDV[NEG]*(float)velEr
roDV[POS]*((float)Output[NEG][POS][NEG][POS]);
CONTROLE = (float)CONTROLE +
(float)anguloDV[NEG]*(float)velAnguloDV[POS]*(float)erroPosDV[NEG]*(float)velEr
roDV[NEG]*((float)Output[NEG][POS][NEG][NEG]);
CONTROLE = (float)CONTROLE +
(float)anguloDV[NEG]*(float)velAnguloDV[NEG]*(float)erroPosDV[POS]*(float)velEr
roDV[POS]*((float)Output[NEG][NEG][POS][POS]);
CONTROLE = (float)CONTROLE +
(float)anguloDV[NEG]*(float)velAnguloDV[NEG]*(float)erroPosDV[POS]*(float)velEr
roDV[NEG]*((float)Output[NEG][NEG][POS][NEG]);
CONTROLE = (float)CONTROLE +
(float)anguloDV[NEG]*(float)velAnguloDV[NEG]*(float)erroPosDV[NEG]*(float)velEr
roDV[POS]*((float)Output[NEG][NEG][NEG][POS]);
CONTROLE = (float)CONTROLE +
(float)anguloDV[NEG]*(float)velAnguloDV[NEG]*(float)erroPosDV[NEG]*(float)velEr
roDV[NEG]*((float)Output[NEG][NEG][NEG][NEG]);
CONTROLE = (2.92571428571428571428571428571392*CONTROLE)*GU; //uIntegral; //
(256/100)*(224/256)

if (flagIntegral == 1) {
      CONTROLE = CONTROLE + (float)uIntegral;
}

if (filtroControle == 1) {
CONTROLE = CONTROLE_ANT*((float)k1) + CONTROLE*((float)k2);
```

```
CONTROLE_ANT = CONTROLE;
}


if  (CONTROLE > 0) {
      if (CONTROLE > 255) CONTROLE = (255);
      controlePWM1 = CONTROLE + offset;
      controlePWM2 = 0;
      }
if (CONTROLE < 0) {
      if (CONTROLE <-255) CONTROLE = (-255);
      controlePWM2 = (-1)*CONTROLE + offset;
      controlePWM1 = 0;
      }
if (CONTROLE == 0) {
      controlePWM1 = CONTROLE;
      controlePWM2 = CONTROLE;
      }
set_pwm1_duty(controlePWM1);
set_pwm2_duty(controlePWM2);

output_low(pin_b0);

      // Amostragem dos valores em display em 1 segundo
    if (aux == TaDisplay)  {

      lcd_escreve('\f');
      tempo++;
        resto = fmod(tempo,20.0);
        if (resto < 10.0) ref = ref1;
        else ref = ref2;
        if ((tempo)==40) {
          printf("\X\r");
            printf(lcd_escreve,"Aperte qualquer tecla para encerrar");
            getc();
            }
 // Envio de dados para o PC
printf(lcd_escreve,"A:%ld P:%ld %3.1f B %3.1f %lu
s",bitDoAngulo,bitDaPosicao,CONTROLE,uIntegral,tempo);
        aux = 0;
      }
printf("%ld %ld %3.2f \r",bitDoAngulo,bitDaPosicao,CONTROLE);
      aux++;

      output_low(pin_b1);
      }

main()
{
      delay_ms(100);
      lcd_ini();
      lcd_escreve('\f');
      printf(lcd_escreve,"Esperando a comunicação com matlab ...");

//%%%%%%%%%%% ANGULO POSITIVO %%%%%%%%%%%%
      Output[POS][POS][POS][POS] = -50;   //NMB
      Output[POS][POS][POS][NEG] = -100;  //NA
```

```
      Output[POS][POS][NEG][POS] = 0;          //ZE
      Output[POS][POS][NEG][NEG] = 25;     //PB
      Output[POS][NEG][POS][POS] = 0;          //ZE
      Output[POS][NEG][POS][NEG] = -75;    //NMA
      Output[POS][NEG][NEG][POS] = 25;     //PB
      Output[POS][NEG][NEG][NEG] = 25;     //PB

//%%%%%%%%%%% ANGULO NEGATIVO %%%%%%%%%%%
      Output[NEG][NEG][NEG][NEG] = 50;     //PMB
      Output[NEG][NEG][NEG][POS] = 100;    //PA
      Output[NEG][NEG][POS][NEG] = 0;          //ZE
    Output[NEG][NEG][POS][POS] = -25;      //NB
      Output[NEG][POS][NEG][NEG] = 0;          //ZE
      Output[NEG][POS][NEG][POS] = 75;     //PMA
      Output[NEG][POS][POS][NEG] = -25;    //NB
      Output[NEG][POS][POS][POS] = -25;    //NB

// Recebimento dos Parametros da simulacao
      gets(auxGanhos);
      G1 = atof(auxGanhos);
      gets(auxGanhos);
      G2 = atof(auxGanhos);
      gets(auxGanhos);
      G3 = atof(auxGanhos);
      gets(auxGanhos);
      G4 = atof(auxGanhos);
      gets(auxGanhos);
      GU = atof(auxGanhos);
      gets(auxGanhos);
      offset = atof(auxGanhos);
      gets(auxGanhos);
      Ti = atof(auxGanhos);

      gets(auxGanhos);
      flagIntegral = atoi(auxGanhos);
      gets(auxGanhos);
      filtroPos = atoi(auxGanhos);
      gets(auxGanhos);
      filtroControle = atoi(auxGanhos);
      gets(auxGanhos);
      freq = atof(auxGanhos);

      lcd_escreve('\f');
      printf(lcd_escreve,"%2.1f %2.1f %2.1f %2.1f %2.1f %1.3f %2.0f %i %i %i
%2.1f",G1,G2,G3,G4,GU,Ti,offset,flagIntegral,filtroPos,filtroControle,freq);

      getc();

// Configura portas analógicas e clock do conversor interno
      setup_ADC_ports (ALL_ANALOG);
      setup_adc(ADC_CLOCK_INTERNAL );

// periodo de amostragem fornecido pelo matlab - dado em segundos
      lcd_escreve('\f');
    gets(amostraString);
      TaInterrupcao = atol(amostraString);
      gets(amostraString);
```

```
      TaDisplay = atol(amostraString);
// Aguarda execução do Matlab para iniciar o programa
      printf(lcd_escreve,"'\f' TaInt: %lu TaDis: %lu",TaInterrupcao,TaDisplay);

//    printf(lcd_escreve,"Esperando começar - amostras = %d",amostras);

      getc();

//    Leitura dos valores iniciais
      set_adc_channel(0);
    delay_us(10);
    bitDoAngulo = read_adc();
      delay_us(10);

    set_adc_channel(1);
    delay_us(10);
    bitDaPosicao = read_adc();
      posicaoAnterior = bitDaPosicao;
      bitDaPosicaoAnterior = bitDaPosicao;

// Configura PWM1 e PWM2:
      setup_timer_2(T2_DIV_BY_4,62,1);
      setup_ccp1(CCP_PWM); set_pwm1_duty(0);
      setup_ccp2(CCP_PWM); set_pwm2_duty(0);

// Configura o timer 0 para clock interno e prescaler dividindo por 16
      setup_timer_0(RTCC_INTERNAL | RTCC_DIV_16);

      set_timer0(TaInterrupcao);

// habilita interrupções
    enable_interrupts (global | int_timer0);
      tempo = 0;
      aux = 0;
      uIntegral_anterior = 0.0;
      k1 = ((Float)1.0)/(1.0 + 2.0*3.1416*freq*0.03);
      k2 = ((float)2.0*3.1416*freq*0.03)/(1.0 + 2.0*3.1416*freq*0.03);
      CONTROLE_ANT = 0;

    while (true){
// espera interrupção
      }
}
```