CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
DEPARTAMENTO DE COMPUTAÇÃO
CURSO DE ENGENHARIA DE COMPUTAÇÃO

SIMULAÇÃO DE CONTROLE MULTIVARIÁVEL NEURO-FUZZY DE UM HELICÓPTERO QUADROTOR:

IMPLEMENTAÇÃO DE UM ALGORITMO PARA ESTABILIZAÇÃO DE UM DRONE

MARCOS FILIPE PARREIRAS

Orientador: Prof. Ms.Ramon da Cunha Lopes
Centro Federal de Educação Tecnológica de Minas Gerais – CEFET-MG

BELO HORIZONTE
OUTUBRO DE 2015

MARCOS FILIPE PARREIRAS

SIMULAÇÃO DE CONTROLE MULTIVARIÁVEL NEURO-FUZZY DE UM HELICÓPTERO QUADROTOR:

IMPLEMENTAÇÃO DE UM ALGORITMO PARA ESTABILIZAÇÃO DE UM DRONE

Monografia apresentada ao Curso de Engenharia de Computação do Centro Federal de Educação Tecnológica de Minas Gerais, como requisito parcial para obtenção do grau de Engenheiro de Computação.

Orientador: Ramon da Cunha Lopes
Centro Federal de Educação Tecnológica de Minas Gerais – CEFET-MG

CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
DEPARTAMENTO DE COMPUTAÇÃO
CURSO DE ENGENHARIA DE COMPUTAÇÃO
BELO HORIZONTE
OUTUBRO DE 2015

Centro Federal de Educação Tecnológica de Minas Gerais

Curso de Engenharia de Computação

Avaliação do Trabalho de Conclusão de Curso

Aluno: Marcos Filipe Parreiras

Título do Trabalho: Simulação de Controle Multivariável Neuro-Fuzzy de um Helicóptero Quadrotor: Implementação de um algoritmo para estabilização de um Drone

Data da defesa: 18/11/2015

Horário: 14:00

Local da defesa: Sala 101, Prédio 17 do CEFET-MG - Campus II

O presente Trabalho de Conclusão de Curso foi avaliado pela seguinte banca:

Professor Ramon da Cunha Lopes - Orientador
Departamento de Computação
Centro Federal de Educação Tecnológica de Minas Gerais

Professor Paulo Eduardo Maciel de Almeida - Membro da banca de avaliação
Departamento de Computação
Centro Federal de Educação Tecnológica de Minas Gerais

Professor Tales Argolo Jesus - Membro da banca de avaliação
Departamento de Computação
Centro Federal de Educação Tecnológica de Minas Gerais

Para um irmão sempre presente que recentemente teve uma perda irreparável. Para você, Goiaba.

Agradecimentos

Agradeço primeiramente e intensamente à minha família. Principalmente ao meu pai, minha mãe e meu irmão. Aos meus pais, Ronaldo e Carmelita, por estarem sempre ao meu lado me apoiando em todos os meus projetos e por terem sido eles os principais responsáveis pela formação do meu caráter. Ao meu irmão, Paulo, pelo companheirismo constante ao longo desses quase vinte e quatro anos, além de ser um exemplo para mim em vários aspectos, como não poderia ser diferente, tendo em vista que ele é o primogênito.

Agradeço também profundamente aos meus amigos. Eles, que caminharam ao meu lado ao longo desta jornada e de tantas outras; que me apoiaram quando precisei de apoio; que dividiram comigo momentos memoráveis e inesquecíveis; que compartilharam diferentes experiências que me levaram a ser exatamente como sou hoje.

Não poderia deixar de agradecer também à minha namorada, Laís, que foi minha principal fonte de inspiração quando esta mais foi necessária, além de ter me apoiado num momento decisivo, principalmente relacionado a este trabalho . Após tantos anos de amizade, só há uma coisa a dizer sobre esse novo nós: "veio como o vento".

Sou imensamente grato à instituição CEFET-MG, que faz parte da minha vida há quase nove anos e que teve substancial importância na minha formação acadêmica, profissional e, claro, pessoal. Foi nela que fiz algumas das grandes amizades que tenho hoje. E foi ainda graças a esta instituição que pude ter uma das experiências mais marcantes da minha vida: residir por um ano em território francês na minha tão querida Grenoble. Por tudo isso, muito obrigado, CEFET-MG.

Muito obrigado também à *Université Joseph Fourier* (UJF) e ao *Institut Laue-Langevin* (ILL) que foram, respectivamente, a universidade onde pude estudar durante meu intercâmbio e a empresa onde fiz estágio durante ele. *À vous, merci bien. Vous me manquez beaucoup*.

Por fim, mas não menos importante, agradeço a todos os professores que tive ao longo desse caminho, cada qual fundamental em diferentes aspectos da minha formação, indo muito além de formadores de profissionais e alcançando o status de formadores de pessoas.

"O ontem é história, o amanhã é um mistério, mas o hoje é uma dádiva. É por isso que se chama presente." (Mestre Oogway, no filme Kung Fu Panda)

Resumo

A gama de aplicação de helicópteros quadrotores tem crescido substancialmente nos últimos anos, sendo eles utilizados inclusive para fins militares. Entretanto, para que o seu uso seja possível, se faz necessário o desenvolvimento de controladores que permitam seu funcionamento adequado de forma a garantir sua estabilidade. Para tanto, foram desenvolvidos ao longo deste trabalho, para estabilizar a atitude e altitude de um drone , dois controladores que aplicam técnicas diferente de IC: fuzzy e neuro-fuzzy. Os resultados obtidos por eles são comparados e mostra-se que ambos estabilizam o sistema de forma eficiente e robusta e, ainda, que o controlador neuro-fuzzy obteve melhores resultados, como é mostrado em diversos casos na literatura. Com isto, mostra-se que técnicas de IC podem ser aplicadas para a construção de controladores eficientes para sistemas não-lineares complexos e que o poder de aprendizado das RNAs é, de fato, capaz de melhorar a performance de um um sistema fuzzy, o que pode ser visto pelo melhor desempenho obtido pelo controlador neuro-fuzzy.

Palavras-chave: quadrotor. controle neuro-fuzzy. inteligência computacional.

Abstract

The range of applications of drones have grown substantially in the last years, being used even for military services. However, in order to make their use possible, it's needed the development of controllers which allow their suitable operation to guarantee their stability. Therefore, in this paper were developed, to stabilize a drone's attitude and altitude, two controllers using CI techniques: fuzzy and neuro-fuzzy. The results that they obtained were compared and it's shown that both were able to stabilize the system with efficiency and robustness and yet that the neuro-fuzzy got better results what is also shown in several cases in literature. From it, one can see that CI techniques may be applied in the construction of efficient controller for complex non linear systems and that the Neural Networks' power of learnig it's indeed able to increase a fuzzy system's performance, what can be seen from the better results obtained using the neuro-fuzzy controller.

Keywords: quadcopter. neuro-fuzzy control. computational intelligence.

Lista de Figuras

Lista de Quadros

Lista de Abreviaturas e Siglas

ANFIS	*Adaptive Neuro-Fuzzy Inference Systems*
ANP	*Adaptive Neuro PID Controller*
CC	Corrente Contínua
CEFET-MG	Centro Federal de Educação Tecnológica de Minas Gerais
CVNF	*Complex-Valued Neuro-Fuzzy*
DECOM	Departamento de Computação
FIBS	*Fuzzy Integral Backstepping*
FIS	Fuzzy Inference System
FLC	Fuzzy Logic Controller
FSBC	*Fuzzy Supervisory Backstepping Controller*
GMP	*Generalized Modus Ponens*
IA	Inteligência Artificial
IC	Inteligência Computacional
IBC	*Integral Backstepping Controller*
IBS	*Integral Backstepping*
LMI	*Linear Matrix Inequalities*
LQ	Linear-Quadrático
LQR	*Linear-Quadratic-Regulator*
MLP	*Multilayer Perceptron*
NF	Neuro-Fuzzy
PD	Proporcional-Derivativo
PDC	*Parallel Distributed Compensation*
PID	Proporcional-Integral-Derivativo
PSO	*Particle Swarm Optimization*

PVTOL	*Planar Vertical Take-Off and Landing*
RNA	Rede Neural Artificial
SC	*Soft Computing*
SE	*Square Error*
SMC	*Sliding Mode Control*
TSK	Takagi-Sugeno-Kang
UAV	*Unmanned Aerial Vehicle*
UGV	*Unmanned Ground Vehicle*
VTOL	*Vertical Take-Off and Landing*

Lista de Símbolos

g	Aceleração da gravidade
m	Massa do pêndulo invertido; Massa do quadrotor
M	Massa do carro no sistema do pêndulo invertido
I_{xx}	Momento de inércia do quadrotor ao longo do eixo x
I_{yy}	Momento de inércia do quadrotor ao longo do eixo y
I_{zz}	Momento de inércia do quadrotor ao longo do eixo z
u_1	Empuxo total gerado pelos quatro rotores do quadrotor
u_2	Momento em torno do eixo x do quadrotor
u_3	Momento em torno do eixo y do quadrotor
u_4	Momento em torno do eixo z do quadrotor
V_i	Voltagem aplicada ao i-ésimo rotor do quadrotor
x	Coordenada x do posicionamento do quadrotor
y	Coordenada y do posicionamento do quadrotor
z	Coordenada z do posicionamento do quadrotor
τ	Intervalo de tempo
ϕ	Ângulo de *roll* (de arfagem) do quadrotor (ângulo de Euler)
θ	Ângulo de *pitch* (de rolamento) do quadrotor (ângulo de Euler)
ψ	Ângulo de *yaw* (de guinada) do quadrotor (ângulo de Euler)

Sumário

1 Introdução

Quadrotores ou *drones* são aeronaves cuja propulsão é obtida a partir do uso de quatro rotores. Apesar de não possuir ampla aplicação comercial atualmente para transporte de pessoas, este modelo de aeronave foi um dos primeiros com rotores a obter sucesso em um voo. O primeiro teste de que se tem registro ocorreu em 1921, quando o Quadrotor De Bothezat conseguiu fazer um voo com duração de dois minutos e quarenta e cinco segundos (ORSAG; BOGDAN, 2012).

Os quadrotores são divididos em duas categorias principais: os do tipo *Indoor* são aqueles projetos para serem utilizados em ambientes controlados[1], ao passo que os *Outdoor* são aqueles projetos de forma a estarem aptos à utilização mesmo em ambientes externos e, portanto, sujeitos a fatores naturais não controlados. Quadrotores de ambas as categorias vêm sendo utilizados para diversas finalidades.

Na última década, helicópteros quadrotores vêm recebendo cada vez mais atenção devido a suas aplicações civis e relacionadas à pesquisa científica (AL-YOUNES; JARRAH, 2008), além do uso militar (SENKUL; ALTUG, 2013). Um dos motivos para isto é o princípio de voo usado por eles. Os quadrotores se enquadram na categoria das aeronaves VTOL (*Vertical Take-Off and Landing*[2]) ou, mais especificamente, PVTOL (*Planar Vertical Take-Off and Landing*[3]), possuindo portanto propulsão vertical e tanto decolagem quanto aterrissagem praticadas em baixa velocidade. Por possuir esta característica, as aeronaves desta categoria se mostram úteis nas mais diversas situações, tendo em vista que apenas uma mínima área em terra firme é exigida para permitir que elas decolem ou aterrissem, possibilitando a execução de tarefas que seriam difíceis ou até mesmo impossíveis de outra forma (REZAZADEH et al., 2013).

Ainda pouco usado para transporte de pessoas ou cargas pesadas, a grande maioria das aplicações com quadricópteros envolvem modelos pequenos não tripulados para transporte de equipamentos mais leves ou mesmo para apenas obtenção de informações sobre o terreno, como em caso de aplicações militares, por exemplo. Estes quadrotores não tripulados se enquadram na classe dos *UAVs (Unmanned Aerial Vehicles)*[4]. Apesar da enorme variedade de aplicações, o uso dos quadrotores também envolve diferentes desafios.

Apesar da grande gama de possibilidades que os quadrotores oferecem, o controle necessário para mantê-los estáticos ou em movimento no ar não é trivial, principal-

[1] e.g. sem a presença de vento
[2] Decolagem e Aterrissagem Verticais, tradução nossa
[3] Planadores com Decolagem e Aterrissagem Verticais, tradução nossa
[4] Veículos Aéreos não Tripulados, tradução nossa

mente se tratando dos modelos *Outdoor*. A complexidade do controle a ser implementado deve-se ao fato de que ele deve atuar sobre mais de uma variável, caracterizando, portanto, um problema de controle multivariável. As múltiplas variáveis a serem controladas são referentes às diferentes movimentações que os quadrotores permitem, que são em três dimensões. Com isto, são seis as variáveis a serem controladas. Três delas são referentes à posição do quadrotor em cada dimensão: x, y e z. As outras três representam o ângulo do quadrotor em relação a cada um dos eixos: ao eixo x é chamado de ângulo de *roll* (ou de arfagem); ao eixo y, de ângulo de *pitch* (ou de rolamento); e ao eixo z, de ângulo de *yaw* (ou de guinada).

Este controle vem sendo objetivo de extensivas pesquisas no campo de sistemas de controle autônomos. Vários algoritmos para estabilização e controle utilizando diferentes paradigmas vêm sendo propostos. Bouabdallah et al. (2004) faz uma comparação entre controladores PID (Proporcional-Integral-Diferencial) e LQ (Linear-Quadrático) aplicados ao controle de quadrotores e conclui que o controlador PID alcança resutados mais robustos. Ainda, Adigbli (2007) mostra que o controle PID não é eficaz como controlador de rastreamento de *check point*. Paralelamente às técnicas convencionais de controle, as inteligentes também têm sido alvo de diversos estudos.

A IA (Inteligência Artificial) e a IC (Inteligência Computacional) vêm ganhando espaço na construção de controladores para quadrotores, como é feito em Boudjedir (2012), em que é usada uma RNA (Rede Neural Artificial) para que o quadrotor lide de forma eficaz contra distúrbios e turbulências, como os gerados pela presença de vento.

A proposta deste trabalho é implementar controladores neuro-fuzzy para garantir a estabilidade em altitude e atitude do quadrotor. Um controlador neuro-fuzzy vai além de um sistema baseado na lógica fuzzy, tendo em vista que o primeiro alia a capacidade de aprendizado de uma RNA bem como sua capacidade de generalização à teoria de conjuntos fuzzy.

1.1 Relevância

Muitos dos quadrotores disponíveis atualmente no mercado são dotados de câmeras filmadoras, para auxiliar em um controle efetuado a longa distância ou mesmo para capturar informações do local sobrevoado por eles. Desta forma, o desenvolvimento de um controlador eficaz para quadrotores poderá possuir diferentes aplicações. Este poderia ser, por exemplo, um meio eficiente para transporte de mantimentos e/ou medicamentos para pessoas que se encontram em áreas de difícil acesso (e.g. após alguma catástrofe natural). Do ponto de vista militar, o uso de quadrotores pode ser aplicado para reconhecimento aéreo de áreas de difícil ou perigoso acesso.

As aplicações dos helicópteros quadrotores vão além das já alcançadas hoje

pelos quadrotores de pequeno porte. Segundo Orsag e Bogdan (2012), alguns *designs*, como *Bell Boeing Quad TiltRotor*, estão sendo projetados para operações de carga pesada. Com isto, novas possibilidades surgiriam como, por exemplo, o próprio transporte de pessoas.

1.2 Objetivo

Este trabalho tem, como objetivo, primeiramente contextualizar a necessidade do desenvolvimento de controladores apropriados para diferentes sistemas não lineares intrinsecamente instáveis e, então, propor diferentes abordagens de Inteligência Computacional para controlar de forma eficiente a estabilidade de atitude e altitude de um quadrotor.

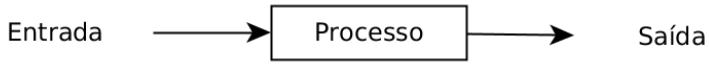
2 Fundamentação Teórica

Neste capítulo, são abordados os temas necessários para a compreensão do trabalho desenvolvido. Na seção 2.1 é feita uma introdução aos sistemas de controle, na 2.2, é discutido o que é uma Rede Neural Artificial (RNA), na 2.3 se trata da lógica fuzzy e, por fim, na seção 2.4 é apresentada a rede neuro-fuzzy.

2.1 Sistemas de Controle

Um sistema de controle é, segundo Dorf (2011, p. 2), uma interconexão de componentes formando uma configuração de sistema que vai fornecer uma resposta desejada. Todo sistema de controle tem, como objetivo, atuar sobre um determinado processo, o qual pode ser representado por um bloco que representa a relação entre entrada e saída do sistema, como mostrado na Figura 1.

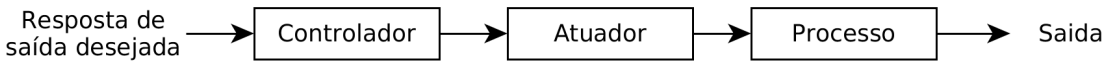
Figura 1 – Diagrama representando um processo



Fonte: Adaptado de Dorf (2011, p. 2)

O controle sobre um processo pode assumir duas formas distintas: malha aberta ou malha fechada sendo que um sistema de controle em malha aberta é composto, além do processo, por um controlador e um atuador para se obter a resposta desejada sem o uso de realimentação (DORF, 2011, p. 2). Um exemplo de sistema deste tipo é mostrado na Figura 2.

Figura 2 – Diagrama representando um sistema de controle em malha aberta

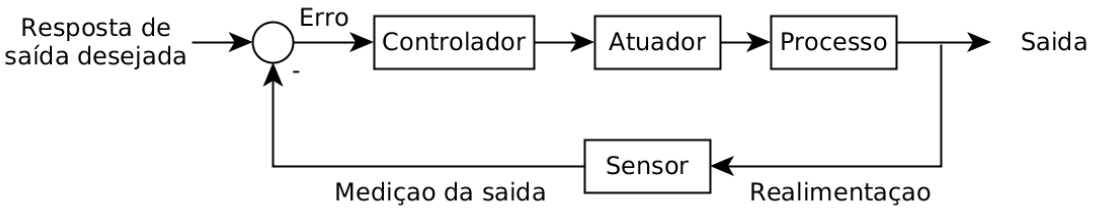


Fonte: Adaptado de Dorf (2011, p. 2)

Em oposição a um sistema de controle em malha aberta, um em malha fechada encorpora, além dos componentes que aquele inclui, uma medição da saída atual para ser comparada com a saída desejada para o processo. Um exemplo de um sistema de controle em malha fechada simples é mostrado na Figura 3. Os sistemas em malha fechada possuem várias vantagens sobre os em malha aberta como, por exemplo, a capacidade de rejeitar distúrbios externos e melhorar a atenuação de ruídos nas

medições, componentes estes que são inevitáveis em aplicações no mundo real (DORF, 2011, p. 3).

Figura 3 – Diagrama representando um sistema de controle em malha fechada



Fonte: Adaptado de Dorf (2011, p. 3)

Como já foi visto, um processo representa a relação entre a entrada e a saída de um sistema sendo que há diferentes formas de fazê-lo. As próximas seções discutem as principais delas: as funções de transferência e a representação no espaço de estados.

2.1.1 Funções de Transferência

A função de transferência de um sistema representa a relação que descreve as dinâmicas do sistema em questão e é definida pela razão entre as transformadas de Laplace das variáveis de saída e de entrada com todas as condições iniciadas definidas como zero (DORF, 2011, p. 65). A forma de uma função de transferência é dada a seguir:

$$G(s) = \frac{Y(s)}{X(s)}$$

em que $G(s)$ é a função de transferência que descreve o sistema, $Y(s)$ é a transformada de Laplace da variável de saída do sistema e $X(s)$ é a transformada de Laplace da variável de entrada, ambas considerando as condições iniciais definidas como zero.

Apesar de ser amplamente utilizada, o uso de funções de transferência tem certas limitações. As transformadas de Laplace só podem ser utilizadas sobre sistemas descritos por equações e diferencias lineares e sem parâmetros variantes no tempo e, portanto, o uso das funções de transferência se restringem aos casos em que estas condições são satisfeitas.

Além da representação a partir de funções de transferência, uma forma alternativa de representar as dinâmicas do sistema é a representação no espaço de estados.

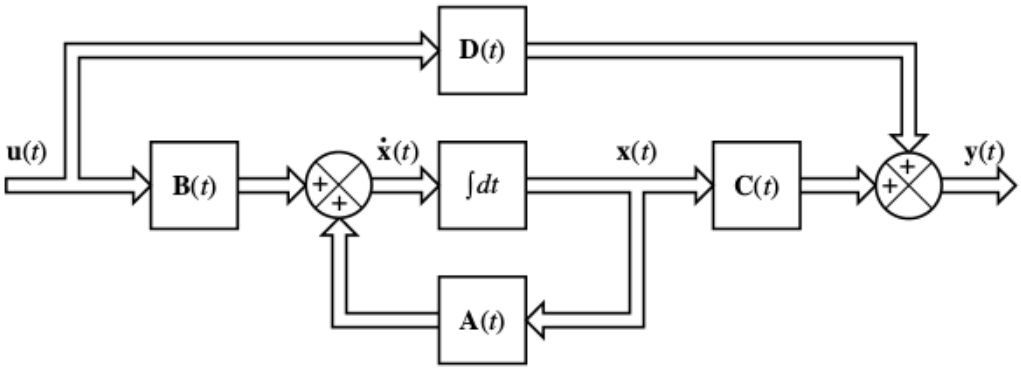
2.1.2 Espaço de Estados

A representação de um sistema dinâmico no espaço de estados descreve um sistema a partir das seguintes equações :

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$

em que A é a matriz de estados, B a matriz de entrada, C a matriz de saída, D a matriz de transmissão direta, x é o vetor de variáveis de estado, u o vetor de entradas e y o vetor de saídas. A Figura 4 exibe um diagrama de blocos para a representação em espaço de estados definida.

Figura 4 – Diagrama de blocos de um sistema linear e contínuo tempo representado no espaço de estados



Fonte: Ogata (2010, p. 32)

A representação no espaço de estados oferece uma ferramenta poderosa para a manipulação da representação do sistema, permitindo que as variáveis de estado sejam representadas de forma independente entre si a partir do processo de desacoplamento delas.

O processo de desacoplamento de variáveis de estado é baseado em ferramentas matemáticas aplicando manipulações sobre frações parciais. A partir deste processo, por exemplo, o sistema mostrado na Figura 5, que representa um sistema de controle de um motor CC com velocidade como saída e é representado pela função de transferência:

$$G(s) = \frac{30(s+1)}{(s+5)(s+2)(s+3)} \tag{1}$$

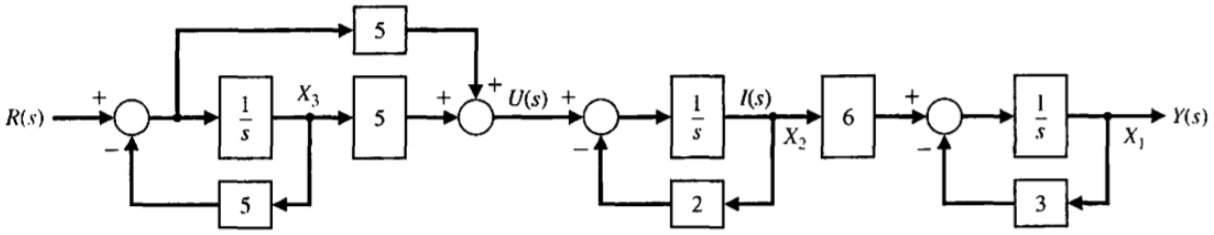
pode ser representado por uma função de transferência do tipo:

$$G(s) = \frac{q(s)}{(s-s_1)(s-s_2)(s-s_3)} \tag{2}$$

cuja resposta é ditada por s_1, s_2 e s_3. Utilizando a expansão de frações parciais, pode-se representar a mesma função de transferência por (DORF, 2011, p. 183):

$$T(s) = \frac{k_1}{s+5} + \frac{k_2}{s+2} + \frac{k_3}{s+3} \tag{3}$$

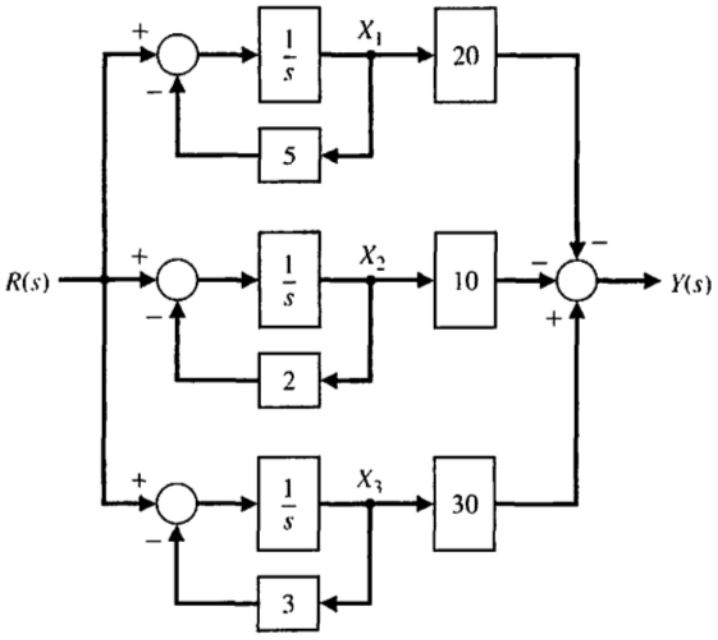
Figura 5 – Diagrama de blocos representando o controle de um motor CC com velocidade como saída



Fonte: Dorf (2011, p. 182)

A partir de manipulações matemáticas relacionadas à expansão de frações parciais, constata-se que $k_1 = -20$, $k_2 = -10$ e $k_3 = 30$ (DORF, 2011, p. 182). Com isto, o sistema exibido na Figura 5 pode ser representado como mostrado na Figura 6 que, como se pode ver, trata X_1, X_2 e X_3 de forma completamente independente e somando suas respectivas contribuições em $Y(s)$. Desta forma, pode-se ver claramente como cada variável de estado contribui para a saída ($Y(s)$) a partir de uma dada entrada ($X(s)$).

Figura 6 – Diagrama de blocos representando o controle de um motor CC com velocidade como saída e implementando desacoplamento das variáveis de estado



Fonte: Dorf (2011, p. 182)

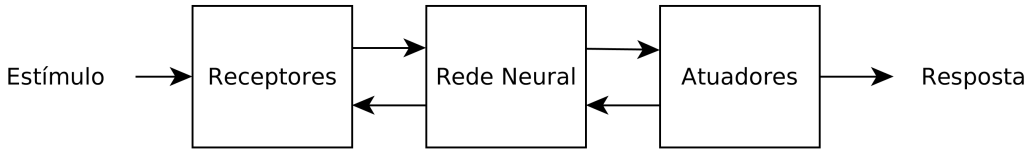
2.2 Redes Neurais Artificiais

Redes Neurais Artificiais (RNAs) são modelos computacionais bioinspirados no sistema neurológico humano. A motivação para o desenvolvimento e uso destes

modelos é a grande complexidade do cérebro humano, definido por Haykin (1998) como um computador altamente complexo, não linear e paralelo. O autor ainda define uma RNA como"*a machine that is designed to* model *the way in which the brain performs a particular task or function of interest*"[1]. Seguindo o modelo biológico do cérebro humano, uma RNA é composta por neurônios artificiais e pelas interações existentes entre estes neurônios: as sinapses.

A Figura 7 ilustra, em um diagrama de blocos, o sistema nervoso como um sistema de três estágios. A Rede Neural representa o cérebro em si, que recebe informações continuamente, as percebe e toma as decisões apropriadas para cada uma delas (HAYKIN, 1998, p. 24).

Figura 7 – Representação em digrama de blocos do sistema nervoso



Fonte: Adaptado de Haykin (1998, p. 28)

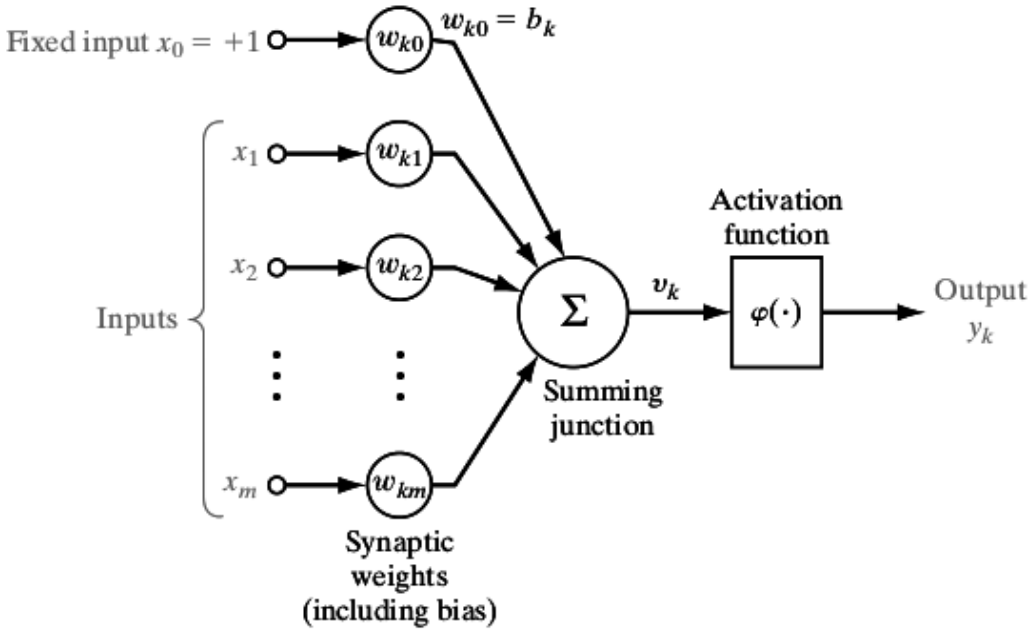
Toda esta comunicação se dá a partir de sinapses, que são unidades estruturais e funcionais que mediam a interação entre neurônios. Seu funcionamento simplificado é o seguinte: o processo pré-sináptico libera uma substância transmissora que é difundida através da junção sináptica entre neurônios e então age sobre o processo pós-sináptico. Então, a sinapse converte o sinal elétrico pré-sinápitico em um sinal químico e, por fim, de volta a um sinal elétrico pós-sináptico (HAYKIN, 1998, p. 28). É através deste processo de comunicação entre neurônios que adquirimos novos conhecimentos e relacionamos estímulos a respostas. É também devido a ele, que podemos fazer associações diversas com acontecimentos no passado, evento que chamamos de *memória*. O imenso poder que os neurônios possuem inspirou modelagens computacionais capazes de atuar em situações em que se deseja obter respostas adequadas a diferentes estímulos, e em outras relacionadas à memória e aprendizado. O modelo de RNA mais aplicado na literatura é o perceptron.

Um perceptron é um modelo computacional de um neurônio não linear e é ilustrado na Figura 8, em que y_k é a saída do sistema obtido após o processamento neural relacionado às entradas x_i, cada qual contribuindo com um peso w_{ki} para a junção de soma representado pelo bloco $\sum$. O processamento neural envolve ainda a função de ativação ϕ que, de acordo com o valor obtido na junção de soma, define o valor da saída y_k. Se o valor v_k for maior que um limiar pré-determinado, a saída do

[1] "uma máquina que é desenvolvida para *modelar* a forma como o cérebro desempenha uma tarefa específica ou função de interesse", tradução nossa.

sistema é ativada e terá o valor 1, caso contrário assumirá o valor 0, simulando assim o processo de transmissão ou não de impulsos elétricos que ocorrem nos neurônios biológicos.

Figura 8 – Modelo não linear de um neurônio



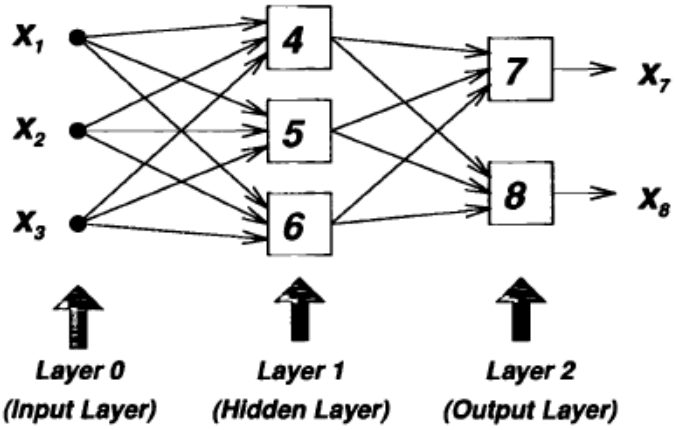
Fonte: Adaptado de Haykin (1998, p. 33)

O modelo simples de um neurônio representado por perceptrons permite a simulação das já definidas sinapses, possibilitando a criação de RNAs complexas formadas por múltiplos neurônios, organizados em cadeias, como é mostrado na Figura 9. Neste exemplo, x_1, x_2 e x_3 são as entradas do sistema; os blocos 4, 5 e 6 representam, cada qual, um neurônio numa camada escondida; os blocos 7 e 8 representam os dois neurônios que compõem a camada de saída; e, por fim, x_7 e x_8 são as saídas de RNA. Redes como essa, que possuem mais de uma camada de neurônios, são denominadas *Multilayer Perceptron* (MLP[2]). As diferentes combinações que se podem obter distribuindo os neurônios de uma RNA de diferentes maneiras fazem com que estas redes possuam aplicações diversas relacionadas à IA e IC.

A principal característica de uma rede neural artificial que a torna interessante para aplicações computacionais é a sua capacidade de aprendizado. O procedimento usado para efetuar o processo de aprendizado é chamado *algoritmo de aprendizado*, cuja função é modificar os pesos sinápticos da rede de forma ordenada para atingir um objetivo desejado (HAYKIN, 1998, p. 24). É a partir deste aprendizado que as RNAs alcançam uma ótima taxa de generalização, fazendo delas fortes aliadas, por exemplo, para reconhecimento de padrões. Controladores que utilizam técnicas relacionadas

[2] Perceptron Multicamadas, tradução nossa

Figura 9 – Modelo de um *MLP*



Fonte: Jang et al. (1997, p. 205)

a RNAs se beneficiam justamente desta capacidade de aprendizado e generalização conferidas por elas.

O controlador desenvolvido neste trabalho utiliza RNAs com treinamento supervisionado, que são treinadas a partir de um conjunto de entradas mapeadas no valor de suas respectivas saídas, resultando na obtenção de um conjunto de retas com parâmetros ajustados de acordo com os dados do treinamento. São as retas obtidas após este treinamento que conferem à rede o poder de generalização, permitindo que novas entradas sejam mapeadas para saídas que condizem com o cenário em questão.

Além disto, como o controlador projetado é do tipo Neuro-Fuzzy, as retas ajustadas após o treinamento se enquadram num grupo especial correspondendo cada qual a uma função de transferência de conjuntos Fuzzy, assunto este que é abordado na seção seguinte.

2.3 Lógica Fuzzy

A lógica fuzzy é uma alternativa à lógica convencional[3], que permite uma abordagem diferente relacionada à pertinência de elementos a conjuntos implementando a possibilidade de se obter uma pertinência parcial para a definição desses conjuntos. A principal aplicação da lógica fuzzy são os sistemas de inferência fuzzy, tema que será abordado na seção 2.3.4. As seções 2.3.1, 2.3.2 e 2.3.3 referentes a conjuntos, regras e raciocínio fuzzy respectivamente tratam dos diferentes componentes utilizados nestes sistemas.

[3] lógica binária

2.3.1 Conjuntos Fuzzy

Os conjunto fuzzy são os componentes elementares dos sistema de inferência fuzzy e se contrapõem àqueles definidos pela lógica tradicional, que restringe, aos valores 0 e 1, o grau de pertinência μ de um elemento u a um conjunto A. Na lógica tradicional, este grau é definido da seguinte forma:

$$\mu A(u) = 1, \text{ se } u \text{ é um elemento do conjunto } A, \text{ e}$$
$$\mu A(u) = 0, \text{ se } u \text{ não é um elemento do conjunto } A$$

Com isto, ou um elemento pertence a um conjunto ou não. Já os conjuntos fuzzy permitem um grau de flexibilidade acerca do grau pertinência de cada elemento ao conjunto, sendo este grau definido por uma *função de pertinência*. A definição formal de conjuntos fuzzy e funções de pertinência é dada por Jang et al. (1997, p. 14) como:

$$A = \{(x,\mu_A(x))|x \in X\}$$

Nesta definição, um conjunto fuzzy A é composto pelos pares $(x,\mu_A(x))$ de cada elemento x pertencente a um conjunto X, em que x é o elemento em si e $\mu_A(x)$ é o grau de pertinência de x ao conjunto A, que pode assumir qualquer valor entre 0 e 1, em que 0 representa a não-pertinência total do elemento ao conjunto e 1 representa a pertinência total a ele.

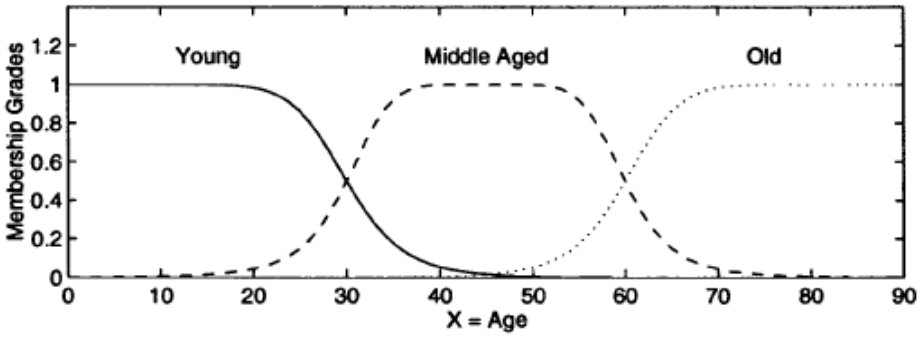
Como se pode perceber, a definição de um conjunto fuzzy é uma simples extensão da referente a um conjunto clássico, no qual a função de pertinência (ou função característica) apenas pode assumir os valores 0 e 1. Se uma função de pertinência $\mu_A(x)$ de um conjunto fuzzy A é restrita a assumir os valores 0 ou 1, então ele é reduzido a um conjunto clássico[4] (JANG et al., 1997, p. 14).

Para melhor exemplificar a definição de conjuntos fuzzy, tome como exemplo a Figura 10. Neste caso, a variável idade (*age*) é dividida em três subconjuntos fuzzy, *young*, *middle aged* e *old*[5], representados pelas funções de pertinência que os descrevem. Cada idade não precisa possuir necessariamente grau de pertinência 1 para algum conjunto e pode ainda pertencer simultaneamente a mais de um. A idade 30, por exemplo pertence ao conjunto jovem com grau de pertinência de 0,5 e também pertence ao conjunto meia idade com o mesmo grau.

Além dos conjuntos fuzzy, há outro aspecto fundamental para o funcionamento desta lógica alternativa e poderosa: as regras fuzzy.

[4] conjunto *crisp*
[5] jovem, meia idade e velho, respectivamente, tradução nossa

Figura 10 – Funções de pertinência representando três conjuntos fuzzy para a variável idade



Fonte: Jang et al. (1997, p. 17)

2.3.2 Regras Fuzzy

As regras fuzzy são os componentes de um sistema de inferência fuzzy responsáveis por definir as relações entre as entradas do sistema e suas saídas e assumem a forma

$$\text{se } x \text{ é } A \text{ então } y \text{ é } B$$

sendo "x é A" denominada premissa e "y é B", consequente da regra em que x e y são *variáveis linguísticas* de entrada e saída respectivamente e A e B são os valores que elas assumem, representados por *termos linguísticos*.

Uma variável linguística, segundo Jang et al. (1997, p. 54), é caracterizada por uma quíntupla $(x,T(x),X,G,M)$ em que x é o nome da variável; $T(x)$ é o conjunto de termos de x, que é o conjunto de seus valores linguísticos ou termos linguísticos; X é o universo de discurso; G é uma regra sintática que gera os termos em $T(x)$; e M é uma regra semântica que associa a cada termo linguístico A seu respectivo $M(A)$, em que $M(A)$ denota um conjunto fuzzy em X.

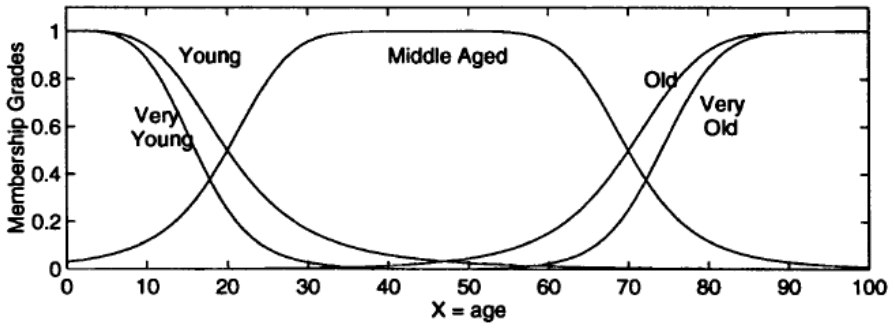
Para facilitar a compreensão das definições relacionadas às variáveis linguísticas, o seguinte exemplo foi dado (JANG et al., 1997, p. 55): se *idade* é interpretado como uma variável linguística, então seu conjunto de termos $T(idade)$ poderia ser dado por:

T(idade) = { *novo, não-novo, muito novo, não muito-novo, . . . ,*
meia-idade, não de meia-idade, . . . ,
velho, não-velho, muito velho, mais ou menos velho, não muito velho, . . . ,
não muito novo e não muito velho, . . . },

Em que cada termo em $T(idade)$ é caracterizado por um conjunto fuzzy de um universo de discurso $X = [0,100]$, como mostrado na Figura 11. Geralmente, se diz

"idade é jovem" para denotar a atribuição do valor linguístico jovem à variável linguística idade. A regra sintática se refere à forma como os valores linguísticos no conjunto de termos $T(idade)$ são gerados. A regra semântica define a função de pertinência de cada valor linguístico do conjunto de termos. A Figura 11 mostra algumas das funções de pertinência típicas para a variável linguística idade.

Figura 11 – Exemplo de função de pertinência do conjunto de termos T(idade)



Fonte: Jang et al. (1997, p. 55)

Diversas regras fuzzy fazem parte do nosso cotidiano. Exemplos possuindo a variável linguística *idade* como variável linguística de entrada e saída respectivamente incluem:

se *idade* é *jovem* então *energia* é *alta*
se *sabedoria* é *grande* então *idade* é *velha*

Um outro componente dos sistemas de inferência fuzzy agrega muito valor ao uso de regras fuzzy, permitindo a aplicação aproximada delas: o raciocínio fuzzy.

2.3.3 Raciocínio Fuzzy

O processo de raciocínio fuzzy, também conhecido como raciocínio aproximado, é um procedimento de inferência que deriva conclusões de um conjunto de regras fuzzy *se-então* como fatos conhecidos (JANG et al., 1997, p. 62) e é a partir dele que se pode fazer uma generalização a partir da regra básica na lógica tradicional com duas variáveis, denominada *modus ponens*.

De acordo com a regra *modus ponens*, podemos inferir a verdade da proposição B a partir da verdade de A e a implicação $A \rightarrow B$ (se A então B). Por exemplo, se A é identifica por "o tomate é vermelho" e B por "o tomate está maduro", então se é verdade que "o tomate é vermelho", é também verdade que "o tomate está maduro".

Contudo, o raciocínio humano emprega constantemente o modus ponens em uma maneira aproximada. Por exemplo, usando a mesma regra de implicação "se

o tomate é vermelho, então ele está maduro", e sabemos que o "o tomate está mais ou menos vermelho" (A'), então podemos inferir que "o tomate está mais ou menos maduro" (B'), em que A' é próximo de A e B' é próximo de B. Quando A, B, A' e B' são conjuntos fuzzy do universo adequado, o procedimento de inferência descrito é chamado raciocínio aproximado ou raciocínio fuzzy, podendo ser também chamado *modus ponens* generalizado (GMP[6]) (JANG et al., 1997, p. 65).

A definição formal do raciocínio aproximado (raciocínio fuzzy) é dada por Jang et al. (1997, p. 65) como: sejam A, A', e B conjuntos fuzzy de X, X e Y respectivamente; assuma que a implicação fuzzy $A \rightarrow B$ é expressa como uma relação R em $X \times Y$. Então, o conjunto fuzzy B induzido por "x é A" e a regra fuzzy "se x é A então y é B" é definida por:

$$\begin{aligned} \mu_{B'}(y) &= \max_x \min[\mu_{A'}, \mu_R(x,y)] \\ &= \vee_x[\mu_{A'}(x) \wedge \mu_R(x,y)], \end{aligned}$$

ou, equivalentemente,

$$B' = A' \circ R = A' \circ (A \rightarrow B).$$

Assim, podemos usar o procedimento de inferência do raciocínio fuzzy para derivar conclusões, dado que a implicação fuzzy $A \rightarrow B$ é definida como uma relação fuzzy binária apropriada.

Uma vez definidos os conjuntos, regras e raciocínio fuzzy, pode-se mostrar como eles são utilizados em conjunto pelos sistemas de inferência fuzzy.

2.3.4 Sistema de Inferência Fuzzy

Um sistema de inferência fuzzy (FIS, do nome em inglês *Fuzzy Inference System*) é uma ferramenta computacional popular e poderosa baseada nos conceitos de teoria de conjuntos fuzzy, regras fuzzy e raciocínio fuzzy. Há uma grande variedade de aplicações para sistemas de inferências fuzzy tais como controle automatizado, classificação de dados, análise de decisão, sistemas especialistas, predição de séries temporais, robótica e reconhecimento de padrões (JANG et al., 1997, p. 73).

A estrutura básica de um FIS consiste de três componentes conceituais: uma base de regras, que contém uma seleção de regras fuzzy; um dicionário (ou base de dados), que define as funções de pertinência usadas nas regras fuzzy; e o mecanismo de raciocínio, que realiza o procedimento de inferência (geralmente o raciocínio fuzzy) sobre as regras e fatos dados para obter uma saída ou conclusão razoável (JANG et al., 1997, p. 73). Sistemas de inferência fuzzy diferentes implementam estruturas

[6] Do inglês, *Generalized Modus Ponens*

ligeiramente diferentes entre si, havendo entretanto, três tipos principais de sistemas que podem ser aplicados aos mais diversos casos.

Os três principais tipos de FIS são os Mamdani, Tsukamoto e Sugeno e a diferença entre eles reside basicamente no consequente de suas regras fuzzy (JANG et al., 1997, p. 74).

No sistema de inferência fuzzy Mamdani, o consequente das regras *se-então* são conjuntos fuzzy. Desta forma, um sistema de inferências fuzzy Mamdani possui regras do seguinte tipo para um sistema com duas entradas e uma saída:

$$\text{se } x \text{ é } A \text{ e } y \text{ é } B, \text{ então } z \text{ é } C$$

Em que x, y e z são variáveis linguísticas nos universos de discurso X, Y e Z, respectivamente e A, B e C são valores (termos) linguísticos. Com isto, um exemplo de sistema de inferências fuzzy Mamdani é formado pelas seguintes regras fuzzy:

$$\begin{cases} \text{Se } X \text{é pequeno e } Y \text{é pequeno então } Z \text{ é muito negativo} \\ \text{Se } X \text{é pequeno e } Y \text{é grande então } Z \text{ é pouco negativo} \\ \text{Se } X \text{é grande e } Y \text{é pequeno então } Z \text{ é pouco positivo} \\ \text{Se } X \text{é grande e } Y \text{é grande então } Z \text{ é muito positivo} \end{cases}$$

No caso do sistema de inferência fuzzy Tsukamoto, o consequente de cada regra fuzzy *se-então* é representada por um conjunto fuzzy com uma função de pertinência monotônica[7]. Um exemplo de modelo fuzzy Tsukamoto de entrada única pode ser expressa como:

$$\begin{cases} \text{Se } X \text{é pequeno então } Y \text{ é } C_1 \\ \text{Se } X \text{é médio então } Y \text{ é } C_2 \\ \text{Se } X \text{é grande então } Y \text{ é } C_3 \end{cases}$$

No sistema de inferência fuzzy Sugeno, também conhecido como TSK (Takagi-Sugeno-Kang), o consequente é um função envolvendo as variáveis de entrada. Com isto, uma regra fuzzy típica de um modelo fuzzy Sugeno possui a seguinte forma:

$$\text{se } x \text{ é } A \text{ e } y \text{ é } B, \text{ então } z = f(x,y),$$

Em que A e B são conjuntos fuzzy do antecedente, enquanto $z = f(x,y)$ é uma função no consequente. Geralmente, $f(x,y)$ é uma função polinomial sobre as variáveis x e y, mas pode ser qualquer função, contanto que seja capaz de descrever apropriadamente a saída do modelo dentro da região fuzzy especificada pelo antecedente da

[7] Uma função sempre crescente ou sempre decresente; uma função cuja derivada nunca muda de sinal

regra. Um exemplo de sistema de inferências fuzzy TSK com duas entradas e uma saída é formado pelas seguintes regras fuzzy:

$$\begin{cases} \text{Se } X \text{é pequeno e } Y \text{é pequeno então } z = -x + y + 1 \\ \text{Se } X \text{é pequeno e } Y \text{é grande então } z = -y + 3 \\ \text{Se } X \text{é grande e } Y \text{é pequeno então } z = -x + 3 \\ \text{Se } X \text{é grande e } Y \text{é grande então } z = x + y + 2 \end{cases}$$

Esta propriedade dos sistemas de inferência fuzzy Sugeno de possuírem, como consequente, funções paramétricas que relacionam as premissas abre várias possibilidades para sua aplicação. Uma delas é implementada por uma técnica que alia o poder das RNAs ao do FIS: o sistema de inferência neuro-fuzzy.

2.4 Redes Neuro-Fuzzy

Aliando o poder das RNAs, que reconhecem padrões e se adaptam para lidar com mudanças no ambiente, aos sistemas de inferência fuzzy, que incorporam o conhecimento humano e executam inferências e tomadas de decisão, surgiram os sistemas de inferência neuro-fuzzy (ANFIS, acrônimo para *Adaptive Neuro-Fuzzy Inference System*), uma nova ferramenta computacional poderosa amplamente utilizada em diversos contextos (JANG et al., 1997, p. 1).

2.4.1 ANFIS: *Adaptive Neuro-Fuzzy Inference Systems*

Segundo Jang et al. (1997, p. 335), do ponto de vista funcional, praticamente não há limitações para a gama de funções de uma rede neuro-fuzzy, exceto pela exigência de ela ser diferenciável por partes. Devido às mínimas restrições, redes neuro-fuzzy podem ser empregadas diretamente em uma grande variedade de aplicações de modelagem, tomadas de decisão, processamento de sinal e controle.

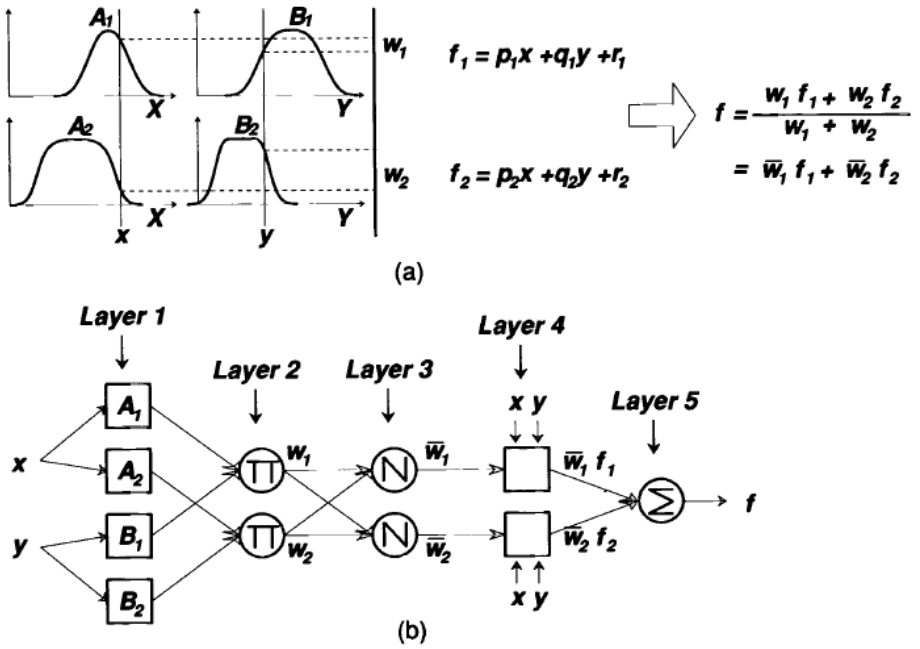
Para simplificar, a arquitetura ANFIS[8] será descrita considerando-a dotada de duas entradas x e y e uma saída z. Para um modelo fuzzy Sugeno de primeira ordem, um conjunto comum de regras do tipo *se-então* é o seguinte:

$$\begin{aligned} &\text{Regra 1: Se } x \text{ é } A_1 \text{ e } y \text{ é } B_1, \text{ então } f_1 = p_1 x + q_1 y + r_1, \\ &\text{Regra 2: Se } x \text{ é } A_2 \text{ e } y \text{ é } B_2, \text{ então } f_2 = p_2 x + q_2 y + r_2, \end{aligned}$$

A Figura 12 ilustra um mecanismo de raciocínio para o modelo Sugeno e a arquitetura ANFIS equivalente, em que nós de uma mesma camada têm funções similares, como descrito a seguir.

[8] Sistema de Inferência Neuro-Fuzzy Adaptativo

Figura 12 – Equivalência entre modelo fuzzy Sugeno e ANFIS; (a) Um modelo fuzzy Sugeno com duas entradas de primeira ordem com duas regras; (b) arquitetura ANIFS equivalente



Fonte: (JANG et al., 1997, p. 336)

Na camada 1 (*Layer 1*), todo nó i é um nó adaptativo com uma função de nó do tipo:

$$O_{1,i} = \mu_{A_i}(x), \text{ para } i = 1, 2, ou$$
$$O_{1,i} = \mu_{B_{i-2}}(y), \text{ para } i = 3, 4$$

em que x (ou y) é a entrada para o nó i e A_i (ou B_{i-2}) é um termo linguístico (como "pequeno" ou "grande") associado a este nó. Em outras palavras, $O_{1,i}$ é o grau de pertinência de um conjunto fuzzy A (A_1, A_2, B_1 ou B_2) e especifica o grau em que a entrada x (ou y) satisfaz o quantificador A. Aqui, a função de pertinência A pode ser qualquer função de pertinência parametrizada apropriada. Parâmetros nesta camada são chamados parâmetros de premissa (JANG et al., 1997, p. 336).

Na camada 2 (*Layer 2*), cada nó é um nó fixo rotulado $\prod$, cuja saída é o produto de todos sinais de entrada:

$$O_{2,i} = w_i = \mu_{A_i}(x)\mu_{B_{i-2}}(y), i = 1, 2.$$

Cada nó de saída representa a força de ativação de uma regra. Em geral, qualquer outro operador norma-T que desempenhe a operação *AND* fuzzy pode ser usado como função de nó nesta camada (JANG et al., 1997, p. 336).

Na camada 3 (*Layer 3*), cada nó é um nó fixo rotulado N. O i-ésimo nó calcula a taxa da força de ativação da i-ésima regra pela soma da força de ativação de todas as regras:

$$O_{3,i} = \bar{w_i} = \frac{w_i}{w_1 + w_2}, i = 1, 2.$$

Por conveniência, saídas desta camada são chamadas forças de ativação normalizadas (JANG et al., 1997, p. 336).

Na camada 4 (*Layer 4*), todo nó i é um nó adaptativo com uma função nodal

$$O_{4,i} = \bar{w_i} f_i = \bar{w_i}(p_i x + q_i y + r_i),$$

em que $\bar{w_i}$ é um taxa da força de ativação normalizada da camada 3 e $\{p_i, q_i, r_i\}$ é o conjunto de parâmetros deste nó. Parâmetros nesta camada são denominados parâmetros consequentes (JANG et al., 1997, p. 336).

O único nó na camada 5 (*Layer 5*) é um nó fixo denominado $\sum$, e é responsável por computar a saída global como a sumarização de todos os sinais de entrada:

$$\text{saída global} = O_{5,i} = \sum_i \bar{w_i} f_i = \frac{\sum_i w_i f_i}{\sum_i w_i}$$

Com esta arquitetura, constrói-se uma rede neuro-fuzzy (ANFIS) que é funcionalmente equivalente a um modelo fuzzy Sugeno (JANG et al., 1997). Com isto, sistemas de inferência neuro-fuzzy podem operar, por exemplo, utilizando o treinamento supervisionado de uma RNA para ajustar o valor dos parâmetros das funções de resposta dos sistemas de inferência fuzzy Sugeno para uma dada operação. Um exemplo de aplicação para este tipo de rede seria o controle de sistemas não lineares.

2.4.2 Controladores Fuzzy e Neuro-Fuzzy

Um controlador lógico fuzzy (FLC)[9] é um sistema de inferência fuzzy projetado especificamente para atuar como controlador de um sistemas que se deseja controlar. Este FIS portanto possui, como entradas, sinais provenientes do sistema e, como saídas, sinais de atuação sobre ele.

Com o avanço dos microprocessadores e hardware, tem se criado uma diversidade ainda maior de domínios para aplicação de controladores lógicos fuzzy, que vão de eletroeletrônicos à indústria automobilística. De fato, para sistemas complexos ou mal definidos que não são facilmente sujeitados a métodos de controle convencionais, os FLCs provêm uma alternativa viável, uma vez que eles podem capturar os aspectos

[9] Do inglês, *Fuzzy Logic Controller*

qualitativos aproximados do raciocínio de um humano e o processo de tomada de decisão. De qualquer forma, sem a capacidade adaptativa, a performance de um FLC se baseia exclusivamente em dois fatores: a disponibilidade de humanos com expertise e as técnicas de aquisição de conhecimento para converter esta expertise humana nas apropriadas regras fuzzy *se-então* e funções de pertinência. Estes dois fatores restringem substancialmente o domínio de aplicações dos FLCs (JANG et al., 1997, p. 451).

Nesse contexto, o uso de controladores neuro-fuzzy passou a se mostrar muito interessante, tendo em vista que eles eliminam os dois fatores citados que restringem bastante o domínio de aplicações dos FLCs, uma vez que os sistemas ANFIS preenchem as lacunas deixadas pelos FIS, eliminando a necessidade de humanos com expertise sobre o sistema e encorporando técnicas de aquisição de conhecimento.

Com isto, podem-se identificar algumas propriedades particulares aos controladores ANFIS que os tornam ótimas opções para diversos casos (JANG et al., 1997, p. 458):

1. Habilidade de aprendizado
2. Operação paralela
3. Representação do conhecimento estruturado
4. Melhor integração com outros métodos de controle

Como comparativo, é importante salientar que uma RNA possui as propriedades 1 e 2, mas não as 3 e 4, que são devidos à contribuição dos sistemas de inferência fuzzy (JANG et al., 1997, p. 458). Um ANFIS, por aliar as duas técnicas, possui todas essas propriedades.

Ainda segundo o autor, há diversas formas diferentes de se projetar controladores neuro-fuzzy. A maioria deles são não lineares. Assim sendo, uma análise rigorosa para sistemas de controle neuro-fuzzy é difícil e continua sendo uma área desafiadora para outras investigações. Por outro lado, um controlador neuro-fuzzy geralmente contém um grande número de parâmetros, o que o torna mais versátil do que um controlador linear ao lidar com características não lineares de plantas. Desta forma, controladores neuro-fuzzy quase sempre superam controladores puramente lineares se devidamente projetados.

3 Sistemas Não Lineares

Este trabalho consiste no projeto de um controlador para um quadrotor que, por se tratar de um sistema não linear intrinsecamente instável é comumente relacionado ao sistema de um pêndulo invertido, outro sistema do mesmo tipo mas com menor grau de complexidade. Desta forma, este capítulo trata, primeiramente e de forma mais superficial, do sistema do pêndulo invertido apenas para fazer uma introdução aos sistemas instáveis e, depois, faz uma introdução ao sistema de um quadrotor, que é o foco principal deste trabalho.

3.1 Analogia ao Pêndulo Invertido

Um sistema de pêndulo invertido consiste em uma haste suspensa verticalmente sobre um carro motorizado, como mostrado na Figura 13. O objetivo do controle de atitude, que é referente ao controle de orientação angular, é manter a haste na posição vertical mesmo após o sistema sofrer alguma perturbação. Como o pêndulo invertido é um sistema instável, após sofrer esta perturbação a haste deverá cair a menos que uma ação de controle adequada seja exercida.

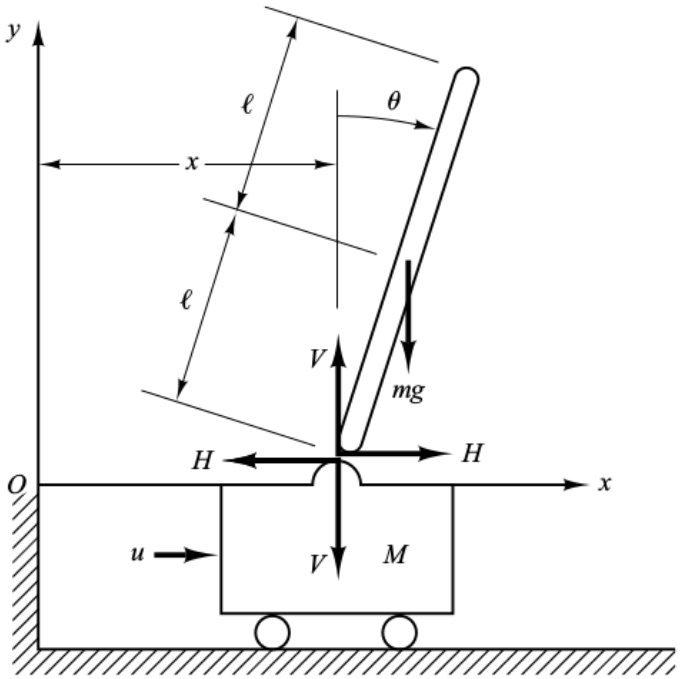
Neste exemplo, é considerado um problema bidimensional, em que o pêndulo se move apenas para a direita e para esquerda (i.e. no plano da página), mas não para frente e para trás (i.e no plano ortogonal a ela). Outra simplificação feita é a consideração de que o centro de gravidade da haste do pêndulo seja seu centro geométrico.

Além disso, o sistema mostrado na Figura 13, assume que uma força u seja aplicada ao carro, considerando M como a massa do carro, m a massa da haste, g a força da gravidade, x o deslocamento no eixo horizontal, l a metade do comprimento da haste, θ o ângulo da haste com relação ao eixo vertical, P o ponto de rotação do pêndulo sobre o carro, H o movimento horizontal do centro de gravidade da haste do pêndulo e V o movimento vertical do centro de gravidade da haste do pêndulo.

Com estes parâmetros, Ogata (2010, p. 69) mostra que o sistema descrito pode ser representado da seguinte maneira no espaço de estados:

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$

Figura 13 – Diagrama do sistema de pêndulo invertido



Fonte: Adaptado de Ogata (2010, p. 69)

em que:

$$x = \begin{bmatrix} \theta \\ \dot{\theta} \\ x \\ \dot{x} \end{bmatrix} \quad y = \begin{bmatrix} \theta \\ x \end{bmatrix}$$

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{M+m}{Ml}g & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -\frac{m}{M}g & 0 & 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ -\frac{1}{Ml} \\ 0 \\ \frac{1}{Ml} \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad D = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

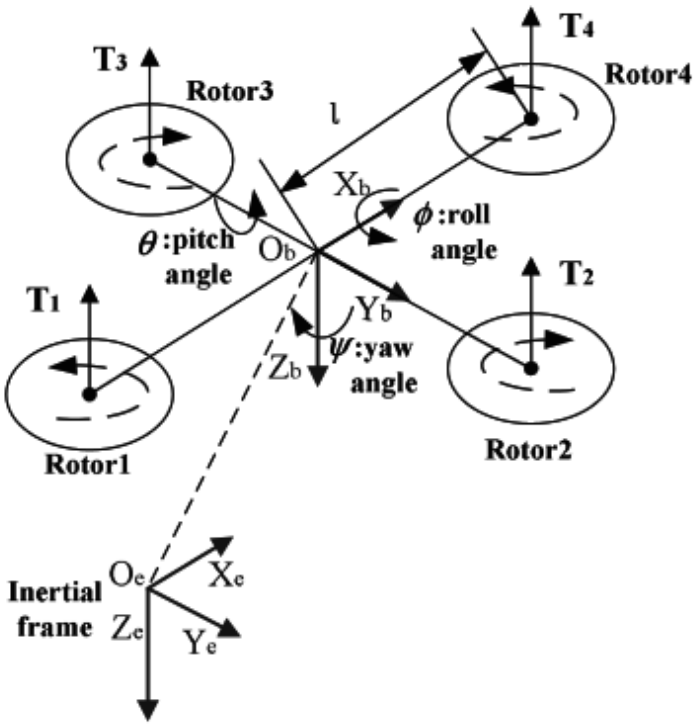
Como o sistema de pêndulo invertido não é o foco deste trabalho, sendo referenciado apenas como uma analogia para sistemas não-lineares instáveis, o desenvolvimento da modelagem matemática mostrado por Ogata (2010, p. 70) não foi explicitado.

Uma vez exibido um modelo de um sistema instável mais simples, pode-se melhor compreender a modelagem de um sistema mais complexo como é o de um quadrotor, assunto este que é abordado na próxima seção.

3.2 Quadrotor

Como definido no Capítulo 1, um quadrotor é uma aeronave cuja propulsão é obtida a partir do uso de quatro rotores e um diagrama o representando é mostrado na Figura 14.

Figura 14 – Diagrama de um quadrotor



Fonte: Adaptado de Gao et al. (2014b)

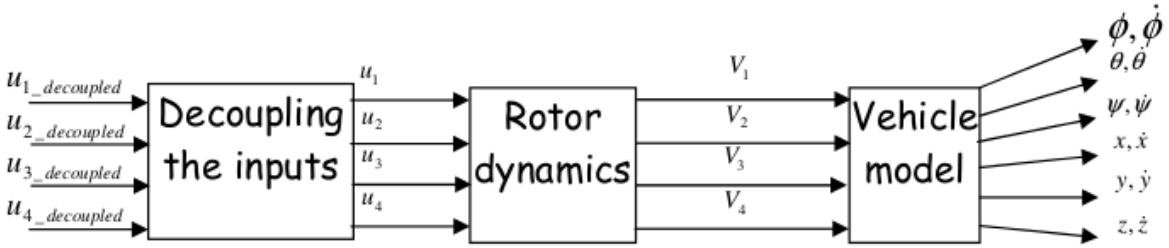
Neste diagrama, x, y e z indicam a orientação dos eixos, F_1, F_2, F_3 e F_4 são as forças geradas pela propulsão de cada um dos motores. As variáveis ϕ, θ, e ψ são os ângulos de rotação do quadricóptero em relação aos eixos x, y e z respectivamente. Estes ângulos são também chamados ângulos de *roll* (arfagem), *pitch* (rolamento) e *yaw* (guinada), respectivamente. Além disto, o diagrama mostra ainda o sentido de rotação de cada um dos quatro rotores. Como se pode ver, os rotores dispostos sobre um mesmo eixo possuem um mesmo sentido de rotação. Desta forma, os rotores 1 e 3 (dispostos sobre o eixo x), giram no sentido horário. Já os rotores 2 e 4 (dispostos sobre o eixo y) giram no sentido anti-horário. Esta disposição dos rotores permite que os empuxos horizontais se anulem, possibilitando a estabilidade do quadricóptero quando submetido a uma ação de controle devida (BASRI et al., 2014, p. 1).

A estabilidade de um quadrotor, entretanto, assume mais de uma forma, sendo dividida em dois aspectos principais: altitude e atitude. O primeiro diz respeito ao posicionamento do quadrotor sobre o eixo z e seu controle é necessário tanto para possi-

bilitar que ele se mantenha num estado estacionário quanto para fazer movimentações no plano XY numa altitude fixa, o que pode ser crucial se ele estiver sendo operado num ambiente limitado inferior e/ou superiormente por alguma espécie de barreira. Já o segundo aspecto, a atitude, se refere à estabilidade angular do quadrotor, permitindo que sua orientação, representada pelos ângulos ϕ, θ, e ψ seja controlada.

Uma vez descrito o comportamento geral do sistema, pode-se partir para a sua modelagem matemática. A modelagem retratada é a que foi desenvolvida por Balas (2007) e é representada pela Figura 15

Figura 15 – Diagrama do sistema representando um quadrotor modelado por Balas (2007)



Fonte: Balas (2007, p. 49)

3.2.1 Modelagem Matemática

O quadrotor é controlado a partir da variação da velocidade dos seus quatro rotores, que são completamente independentes entre si. Desta forma, sendo l o comprimento de cada haste a partir do centro geométrico do quadrotor e considerando v_i e τ_i respectivamente como o torque e o impulso do *i*-ésimo rotor, podem-se considerar as entradas do sistema como sendo (BALAS, 2007, p. 4):

$$u_1 = \tau_1 + \tau_2 + \tau_3 + \tau_4 \tag{4}$$

$$u_2 = l(\tau_3 - \tau_4) \tag{5}$$

$$u_3 = l(\tau_1 - \tau_2) \tag{6}$$

$$u_4 = v_1 + v_2 + v_3 + v_4 \tag{7}$$

em que u_1 é o impulso total, u_2 é o momento de *roll*, u_3 é o momento de *pitch* e u_4 é o momento de *yaw*.

Além disso, a aceleração em cada deixo pode ser representada por (BALAS, 2007,

p. 5):

$$\ddot{x} = -\frac{sen(\theta)cos(\phi)}{m}u_1 \tag{8}$$

$$\ddot{y} = \frac{sen(\phi)}{m}u_1 \tag{9}$$

$$\ddot{z} = -\frac{cos(\theta)cos(\phi)}{m}u_1 + g \tag{10}$$

em que ϕ e θ são os ângulos em torno dos eixos x e y respectivamente, m é a massa do quadrotor e g é a gravidade à qual ele é submetido.

Além das acelerações em cada eixo, é também necessário se definir a aceleração *em torno* de cada eixo, ou seja, as acelerações angulares em torno de x, y e z. Para tanto, assumindo que a estrutura do quadrotor seja rígida, sendo I_{xx}, I_{yy} e I_{zz} o momento de inércia do quadrotor ao longo dos eixos x, y e z respectivamente e considerando que os momentos de inércia ao longo de x e y se equivalem, as acelerações angulares podem ser representadas a partir das seguintes equações (BALAS, 2007, p. 6):

$$\ddot{\phi} = -\dot{\psi}\dot{\theta}cos(\phi) + \frac{cos(\psi)}{I_{xx}}u_2 - \frac{sen(\psi)}{I_{yy}}u_3 + \frac{I_{yy} - I_{zz}}{I_{xx}}(\dot{\psi} - \dot{\theta}sen(\phi))\dot{\theta}cos(\phi) \tag{11}$$

$$\begin{aligned}\ddot{\theta} = \frac{\dot{\psi}\dot{\phi}}{cos(\phi)} + \dot{\phi}\dot{\theta}tan(\phi) + \frac{sen(\psi)}{cos(\phi)I_{xx}}u_2 + \frac{cos(\psi)}{cos(\phi)I_{yy}}u_3 \\ - \frac{I_{yy} - I_{zz}}{I_{xx}}(\psi - \dot{\theta}sen(\phi))\frac{\dot{\phi}}{cos(\phi)}\end{aligned} \tag{12}$$

$$\begin{aligned}\ddot{\psi} = \dot{\phi}\dot{\psi}tan(\phi) + \frac{\dot{\phi}\dot{\theta}}{cos(\phi)} + \frac{sen(\psi)tan(\phi)}{I_{xx}}u_2 + \frac{cos(\phi)tan(\psi)}{I_{yy}}u_3 + \frac{1}{I_{zz}}u_4 \\ - \frac{I_{yy}I_{zz}}{I_{xx}}(\dot{\psi} - \dot{\theta}sen(\phi))\dot{\phi}tan(\phi)\end{aligned} \tag{13}$$

Por fim, deve-se relacionar a velocidade de cada rotor i ao impulso e torque sobre ele. Esta relação é dada por (BALAS, 2007, p. 7):

$$v_i = \tau_i(V_c + v_i) + 0.125\rho bcR_p^4\omega_{m\_i}^2 C_d \tag{14}$$

sendo v_i o torque sobre o rotor, τ_i o impulso atuando sobre ele, V_c a velocidade vertical, v_i a velocidade induzida no motor, ρ a densidade do ar, b o número de pás, c o comprimento delas, R_p o raio da hélice, C_d o coeficiente de arrasto e ω_{m_i} a velocidade angular do rotor.

Após a modelagem de um sistema complexo como este, é de se desejar que se possa isolar a resposta de determinadas variáveis às entradas. Para tanto, pode-se utilizar o processo de desacoplamento de entradas.

3.2.2 Desacoplamento das Entradas

Para obter uma resposta isolada das variáveis do sistema a partir dos sinais de entrada dele, um bloco de desacoplamento foi modelado em (BALAS, 2007, p. 62). Neste caso, foram considerados os acoplamentos entre ϕ, θ, ψ e u_2, u_3, u_4, relacionados da seguinte maneira:

$$u_{2\_decoupled} = cos(\psi_0)u_2 - \frac{I_{xx}sen(\psi_0)}{I_{yy}}u_3 = I_{xx}\ddot{\phi} \tag{15}$$

$$u_{3\_decoupled} = \frac{sen(\psi_0)I_{yy}}{cos(\phi_0)I_{xx}}u_2 + \frac{cos(\psi_0)}{cos(\phi_0)}u_3 = I_{yy}\ddot{\theta} \tag{16}$$

$$u_{4\_decoupled} = \frac{I_{zz}sen(\psi_0)tan(\phi_0)}{I_{xx}}u_2 + \frac{I_{zz}cos(\psi_0)tan(\phi_0)}{I_{zz}}u_3 + u_4 = I_{zz}\ddot{\psi} \tag{17}$$

Como se pode ver pelas equações, aplicando esse desacoplamento obtêm-se as variáveis $u_{2\_decoupled}$, $u_{3\_decoupled}$ e $u_{4\_decoupled}$ relacionadas aos ângulos ϕ, θ e ψ respectivamente e considerando o momento de inércia sobre os respectivos eixos, isolando assim a resposta das diferentes variáveis do sistema.

Além disso, a partir destas equações, podem-se representar essas transformações utilizando o formato matricial da seguinte maneira (BALAS, 2007, p. 62):

$$\begin{bmatrix} u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} cos(\psi_0) & sen(\psi_0)cos_(\phi_0)\frac{I_{xx}}{I_{yy}} & 0 \\ -sen(\psi_0)\frac{I_{yy}}{I_{xx}} & cos(\psi_0)cos(\phi_0) & 0 \\ 0 & -sen(\phi_0)\frac{I_{zz}}{I_{yy}} & 1 \end{bmatrix} \begin{bmatrix} u_{2\_decoupled} \\ u_{3\_decoupled} \\ u_{4\_decoupled} \end{bmatrix}$$

Após os processos de modelagem matemática do sistema e o devido desacoplamento de suas entradas, pode-se representar seu comportamento geral a partir dos espaços de estado, prática comum para descrever sistemas dinâmicos.

3.2.3 Representação no Espaço de Estados

Como mostrado em (BALAS, 2007, p. 63), o sistema modelado, já incluindo o desacoplamento de entradas, pode ser representado da seguinte forma no espaço de estados.

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned} \tag{18}$$

Com o vetor de estados X sendo dado por:

$$X = \begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} & \phi & \theta & \psi & \dot{\phi} & \dot{\theta} & \dot{\psi} \end{bmatrix}^T$$

O vetor de entrada U sendo:

$$U = \begin{bmatrix} u_1 & u_{2\_decoupled} & u_{3\_decoupled} & u_{4\_decoupled} \end{bmatrix}^T$$

E as matriz A, B, C e D sendo definida como:

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -g & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & g & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -1/m & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & cos(\psi)/I_{xx} & -sen(\psi)/I_{yy} & 0 \\ 0 & sen(\psi)/(cos(\phi)I_{xx}) & cos(\psi)/(cos(\phi)I_{yy}) & 0 \\ 0 & sin(\psi)tan(\phi)/I_{xx} & cos(\psi) * tan(\phi)/I_{yy} & 1/I_{zz} \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$D = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

em que g é a gravidade, m, a massa do quadrotor e os ângulos ϕ, θ e ψ representam os ângulos em torno dos eixos x, y e z respectivamente. Além disto, I_{xx}, I_{yy} e I_{zz} representam o momento de inércia do quadricóptero ao longo destes mesmos eixos.

Ao longo deste trabalho, essa foi a modelagem adotada para simular o sistema de um quadrotor.

4 Trabalhos Relacionados

Na literatura, diversos são os exemplos de sistemas não lineares para os quais se desenvolvem diferentes tipos de controladores. Dentre estes tipos, encontramos os PD (Proporcional-Derivativo), PID (Proporcional-Integral-Derivativo), LQ (Linear-Quadrático), Fuzzy e ANFIS dentre outros sendo que este último tem se mostrado bastante presente para tentar atuar sobre diferentes tipos de sistemas.

A grande quantidade de materiais propondo a implementação de controladores neuro-fuzzy se deve, em parte, ao fato de geralmente, para efetuar tal controle, não ser necessário se fazer uma modelagem matemática aprofundada do sistema que, em diversos casos, é de grande complexidade e, por isso, demandaria um grande esforço sendo que em muitos casos os resultados não seriam tão bons quanto os desejados além de oferecer ainda uma boa robustez ao controle.

Além do próprio controle de estabilidade de helicópteros quadrotores, podem-se encontrar exemplos de diversos sistemas que implementam controladores neuro-fuzzy. É o caso da implementação de um controlador para *see and avoid*[1] (referente ao desvio de ocasionais obstáculos) de um *drone*, como se pode encontrar em Olivares-Mendez et al. (2012).

Já Guo et al. (2003), propõem a implementação de um controlador neuro-fuzzy para controlar a estabilidade sobre o balanço de um barco. Os autores mostraram, ainda, que o controlador neuro-fuzzy obteve melhor resultado do que um controlador PID, chegando à conclusão de que controladores neuro-fuzzy são promissores para efetuar tal controle.

A grande gama de opções de controle sobre quadrotores pode ser dividida em dois grupos principais: o controle *tradicional* e o controle *inteligente*. O primeiro diz respeito ao uso de técnicas consolidadas para controle baseado na atuação sobre eles a partir do conhecimento de sua modelagem, ao passo que o segundo se refere às técnicas de controle que envolvem componentes de Inteligência Computacional, agregando, por exemplo, representação de estados através de variáveis linguísticas e capacidade de aprendizagem.

4.1 Controle Tradicional de Quadrotores

Esta seção aborda diferentes propostas de controladores tradicionais para quadrotores visando a contextualização do estado da arte para depois se poder comparar

[1] Ver e evitar, tradução nossa

alguns resultados das técnicas tradicionais e inteligentes de controle para esse sistema.

Em (RAZINKOVA et al., 2014), foi proposto um controlador PD para a descrição adequada das trajetórias definidas de um helicóptero quadrotor *Indoor*. Para permitir um funcionamento adequado *Outdoor*, foi integrado ao controlador PD um controle adaptativo, adicionando termos ao controlador convencional, de forma a permitir o ajuste correto da trajetória do quadrotor quando submetido a distúrbios nos eixos X e Y (plano horizontal). Os resultados mostraram que o controlador adaptativo melhorou consideravelmente o controle de trajetória do drone, reduzindo em 64% o erro de posicionamento para os casos de trajetória em linha reta ao longo do plano XY, e em 72% para os casos de trajetória circular sobre o plano XY. Segundo os autores, este resultado representa uma melhora significativa, uma vez que o controlador resultante possui uma arquitetura simples e não requer computação extensiva, o que é indesejado para qualquer controle de sistema, especialmente para UAVs, devido à sua limitação de bateria.

Em (MUSTAPA et al., 2014) é proposto um controlador PID para efetuar o controle sobre a estabilidade da altitude de um helicóptero quadrotor. Neste trabalho, foi utilizado um modelo matemático desenvolvido previamente para descrever o comportamento do sistema. Para ajustar os parâmetros do controlador PID, foi necessário realizar um experimento para determinar o momento de inércia do drone. Uma vez levantados os valores necessários, um controlador PID foi ajustado e se mostrou eficiente. A simulação do sistema se deu no Mat-lab Simulink.

Khatoon et al. (2014) também propuseram um controlador PID para controlar a estabilidade em altitude de um drone, mantendo a posição no eixo XY constante, mesmo esta altitude sendo uma variável muito sensível a mudanças em outros parâmetros. No trabalho, é mostrado que um controlador PID sozinho é capaz de exercer tal controle com robustez. A escolha deste controlador se deu graças à sua robustez e facilidade de modelagem. Entretanto, é também apontado pelos autores que, apesar da simplicidade para se modelar um controlador PID, isto requer uma modelagem do sistema como um todo, o que não é fácil, devido à sua estrutura complexa, suas dinâmicas não lineares e sua natureza subatuada. O sistema modelado para o drone mostrou ser altamente instável, justificando a necessidade de um controlador. A partir de extensivas simulações no MATLAB/Simulink, o sistema desenvolvido se mostrou bem sucedido, implementando de fato um controle robusto de altitude para um helicóptero quadrotor.

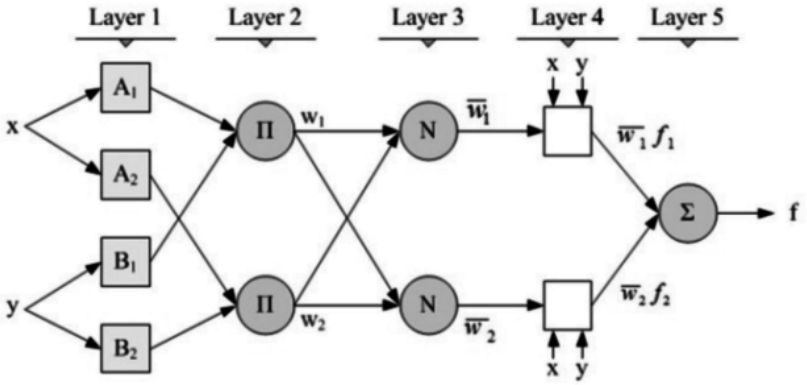
Além desses controladores que aplicam técnicas tradicionais, são também vários os exemplos de trabalhos que retratam o controle inteligente de quadrotores.

4.2 Controle Inteligente de Quadrotores

Além das técnicas de controle tradicionais, encontram-se também na literatura diversas abordagens para a construção de controladores de estabilidade para *drones* utilizando técnicas da Inteligência Computacional. As propostas vão do uso de algoritmos de otimização (e.g. algoritmos genéticos, PSO) ao uso de neuro-fuzzy expandido para o espaço dos números complexos.

Em (REZAZADEH et al., 2013), é proposto um controlador neuro-fuzzy (ANFIS) para controlar a estabilidade em altitude de um quadrotor. O módulo fuzzy acoplado a um controlador PID é usado para controlar cada atitude (i.e. cada um dos ângulos). Os controladores fuzzy atualizam os ganhos do PID *online* para produzir uma resposta apropriada. A Figura 16 ilustra o controlador neuro-fuzzy usado para garantir a estabilidade do quadrotor sendo esta a estrutura típica de uma arquitetura ANFIS para um sistema com duas entradas e uma saída.

Figura 16 – Estrutura do ANFIS multicamadas de duas entradas e uma saída



Fonte: Adaptado de Rezazadeh et al. (2013)

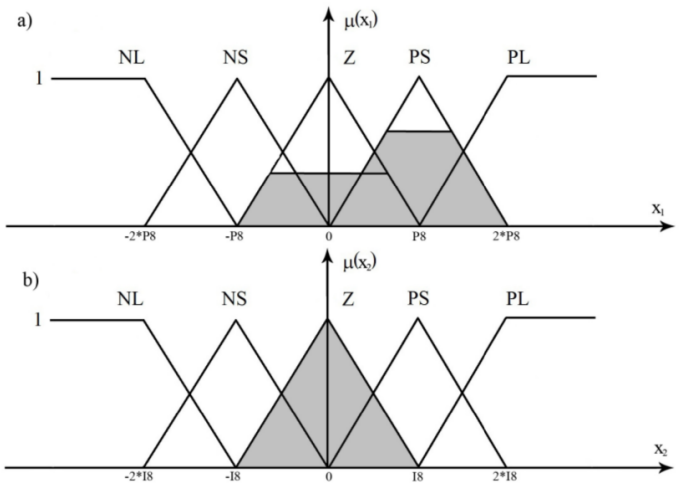
Como se pode ver pela Figura 16, a saída é calculada diretamente pelo ajuste dos pesos das entradas de acordo com as regras de inferência fuzzy. Estas regras, que são a base de conhecimento, são determinadas por um algoritmo computacional baseado em redes neurais (REZAZADEH et al., 2013). O controlador proposto foi comparado a um simples controlador PID e, a partir de simulações, foi mostrado que obteve melhor resposta, por exemplo, sobre o tempo de assentamento e a sobrelevação. Outro resultado obtido foi de que o controlador ANFIS é capaz de estabilizar o sistema quando exposto a perturbações externas.

Em (MAHFOUZ et al., 2013), foi proposto um controlador parecido. Neste trabalho, também foi sugerido um controlador ANFIS que, juntamente a um algoritmo genético, ajusta os pesos de um controlador PID. Mais uma vez, o controlador desenvolvido obteve sucesso, sendo apto a garantir a estabilidade de um drone mesmo

quando sujeito a condições externas. No trabalho, foi mostrado que a sobrelevação do controlador ANFIS é menor do que a do PID. Além disto, o erro do estado estável no caso do ANFIS é zero, enquanto o erro do estado estável no caso do controaldor PID não o é. Uma conclusão muito importante a que se chega é que, do ponto de vista de robustez, os dois controladores são efetivos; do ponto de vista de performance, o ANFIS é melhor que o PID.

Em (MAJ; BUTKIEWICZ, 2013) é proposto um controle de estabilidade para um quadricóptero usando lógica fuzzy. Neste trabalho, três controladores fuzzy são usados: um para cada um dos dois eixos horizontais e mais um controlador geral para todo o sistema. Com isto, o erro angular e a aceleração são as variáveis linguísticas. Em ambos os casos, são usados cinco termos linguísticos: muito negativo (NL), pouco negativo (NS), zero (Z), pouco positivo (PS) e muito positivo (PL). Os conjuntos NS, Z e PS são triangulares enquanto os conjuntos NL e PL são trapezoidais, como mostra a Figura 17.

Figura 17 – Conjuntos fuzzy usados em (MAJ; BUTKIEWICZ, 2013); a) erro de posição; b) erro de giro



Fonte: (MAJ; BUTKIEWICZ, 2013)

O bloco de fuzzificação calcula o grau de confiança dos conjuntos de entrada e um segundo bloco executa a inferência, que calcula o grau de confiança dos conjuntos de saída. As operações MIN e MAX são usadas como operadores AND e OR respectivamente para os conjuntos fuzzy. O conjunto de regras utiizadas são mostradas no Quadro 1 e os conjuntos fuzzy antes da defuzzificação são mostrados na Figura 18.

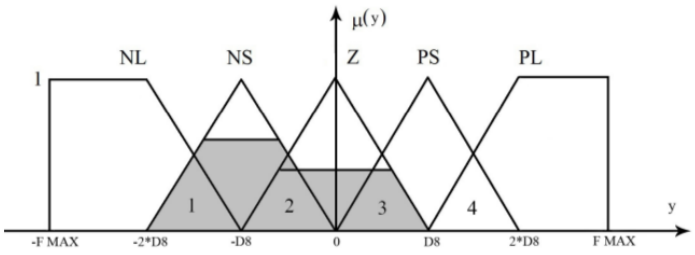
Os autores obtiveram sucesso no controle de estabilidade do *drone*, mas somente utilizando um bloco integrador. Apesar de os reguladores e motores serem do mesmo tipo, algumas características deles são levemente diferentes. Como consequência, os motores não giram todos na mesma velocidade causando uma inclinação do quadrotor. Esta inclinação pode ser corrigida com o uso de um bloco integrador. Desta forma, foi

Quadro 1 – Regras de inferência fuzzy usadas em (MAJ; BUTKIEWICZ, 2013)

	Erro de Giro				
Erro Angular	**NL**	**NS**	**Z**	**PS**	**PL**
NL	PL	PL	PL	PS	Z
NS	PL	PL	PS	Z	NS
Z	PL	PS	Z	NS	NL
PS	PS	Z	NS	NL	NL
PL	Z	NS	NL	NL	NL

Fonte: Adaptado de Maj e Butkiewicz (2013)

Figura 18 – Conjuntos fuzzy antes da defuzzificação usados em (MAJ; BUTKIEWICZ, 2013)



Fonte: (MAJ; BUTKIEWICZ, 2013)

observado que um controlador puramente fuzzy pode não ser suficiente para lidar com esta situação mas que, acoplando um bloco integrador, o sistema certamente pode ser estabilizado.

Coza e Macnab (2006) propuseram um controlador fuzzy adaptativo para controlar a estabilidade de um quadricóptero *Outdoor*. A escolha deste tipo de controlador se deu graças a inconvenientes apresentados por outros métodos. Controle adaptativo pode ser apropriado mas requer conhecimento aprofundado do tipo de não linearidades nas dinâmicas. Um SMC (*Sliding Mode Control*[2]) pode ser apropriado mas pode causar vibração indesejada no sinal de controle. Controles por RNAs podem ser apropriados, mas requerem intenso esforço computacional para operar.

A abordagem utilizada em (COZA; MACNAB, 2006) se baseia na ideia de que diferentes conjuntos de centros de decodificação são capazes de aproximar uniformemente a mesma função não linear. Desta forma, é usado treino supervisionado sobre os centros alternados: o erro do estado treina o centro de controle e a diferença na saída treina os centros alternados. A partir de simulações, mostrou-se que o resultado obtido foi um controle estável, computacionalmente eficiente e teoricamente robusto a perturbações. Entretanto, em uma das situações de simulações de perturbações por

[2] Controle de Deslizamento de Modo, tradução nossa

vento, o controlador teve de sacrificar a boa performance computacional para prevenir o desvio de centro. Ainda assim, foi mostrado que um modelo fuzzy adaptativo usando atualização de controle e centro pode ser usado para cada eixo de rotação: X, Y e Z. Para tanto, apenas quatro funções Gaussianas de pertinência fuzzy foram usadas para cada entrada de controle.

Em (RABHI et al., 2011) uma outra abordagem é utilizada. Neste caso, os autores propuseram um controlador Fuzzy TSK (Takagi-Sugeno-Kang) para controlar a estabilidade de um quadrotor. O controlador foi modelado usando ferramentas matemáticas, mais especificamente a LMI (*Linear Matrix Inequalities*[3]) e modelado empregando PDC (*Parallel Distributed Compensation*[4]). O objetivo foi projetar um controlador fuzzy realimentado para garantir a estabilidade do sistema com robustez. Simulações mostram que o controlador projetado garante, de fato, a estabilidade global do sistema em malha fechada.

Sheikhpour e Shouraki (2013) seguiram a mesma linha de Rabhi et al. (2011) e propuseram um controlador fuzzy TSK para estabilização de altura de um quadrotor. Mais uma vez, a técnica PDC foi utilizada para projetar o controlador de realimentação. Neste trabalho, entretanto, o controle de estabilidade levou em conta especificações de performance como taxa de decaimento e restrições na entrada. Como ambas condições podem ser representadas por LMIs, elas também foram usadas neste caso, sendo mais complexas por lidar com as restrições dadas. Foi mostrado que simultaneamente à resolução das LMIs, foi projetado não apenas um controlador fuzzy estável mas também com inclusão de resposta de velocidade desejada e restrições na altitude de entrada no controlador. Os resultados obtidos em simulações mostram a viabilidade desta abordagem.

Niroumand et al. (2013) propuseram uma forma alternativa de controlador também usando a lógica fuzzy. Na abordagem utilizada, primeiramente foi derivado um modelo dinâmico não linear de um quadrotor e então foi desenvolvido o controlador híbrido usando métodos de controle tradicionais e inteligentes para estabilização de dinâmicas rotacionais. A técnica IBS (*Integral Backstepping*) é um poderoso método de controle tradicional amplamente utilizado para tal tipo de sistema mas encontrar o coeficiente apropriado do algoritmo é um trabalho crítico. Neste caso, esse problema foi resolvido usando um método de controle fuzzy. Foi mostrado nos resultados que o método IBS possui domínio de atração mais amplo em comparação a métodos de controle lineares como o PID e uma melhor convergência devido a condições iniciais rígidas. Foi mostrado ainda que ambos controladores IBS e Fuzzy IBS foram capazes de controlar o sistema de forma apropriada. Entretanto, o controlador FIBS (*Fuzzy Integral*

[3] Desigualdades Matriciais Lineares
[4] Compensação Paralela Distribuída

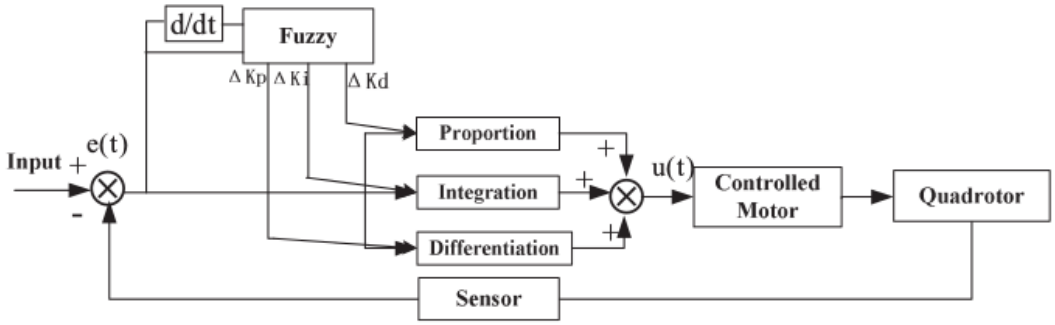
Backstepping) obteve resultados levemente superiores ao IBS além de ter apresentado benefícios como uma melhor rejeição a perturbações e melhor robustez.

Já em (BASRI et al., 2014), foi proposto um controlador FSBC (*Fuzzy Supervisory Backstepping Controller*[5]). O controlador projetado consiste em controlador de *backstepping* que pode selecionar automaticamente seus parâmetros *on line* por um mecanismo de supervisão fuzzy. O critério de estabilidade para a estabilização do quadrotor é provada pelo teorema de Lyapunov. Extensivas simulações mostram a eficácia desta abordagem, apontando que uma alta precisão no transiente e no tempo de controle foram alcançados. Além disto, os resultados indicam que a técnica proposta pode estabilizar quadrotores com melhor performance se comparado a técnicas de estabilização lineares.

Em (GAO et al., 2014b), é proposto um outro tipo de controlador híbrido. Neste caso, o controle é feito a partir de um sistema composto por dois controladores independentes: um controlador *Backstepping* e um segundo controlador Fuzzy PID, que alia a lógica fuzzy a um controlador PID. Quando o quadrotor voa em boas condições, o controle de estabilidade é assumido pelo controlador *Backstepping*. Quando o quadrotor encontra vento ou outro fator de perturbação durante o voo, o controlador Fuzzy PID é usado para estabilizá-lo.

O controlador Fuzzy PID é um controlador PID que emprega o Sistema de Inteferência Fuzzy (FIS) para ajustar os parâmetros K_p, K_i, K_d, ganhos proporcional, integral e diferencial respectivamente. A Figura 19 mostra o diagrama de blocos representando o controlador Fuzzy PID.

Figura 19 – Diagrama de blocos do controlador híbrido usando em (GAO et al., 2014b)



Fonte: Gao et al. (2014b)

Em relação à estrutura fuzzy, há duas entradas para a inferência fuzzy: o erro e a derivada do erro. São três saídas, cada uma sendo um parâmetro do controlador PID Δkp, Δki e Δkd. O universo de discurso foi normalizado e dividido em sete subconjuntos fuzzy. Os termos linguísticos foram definidos da seguinte forma: NB, negativo grande; NM, negativo médio; NS, negativo pequeno; ZO, aproximadamente zero; PS, positivo

[5] Controlador por Backstepping Fuzzy Supervisionado, tradução nossa

pequeno; PM, positivo médio; e PB, positivo grande. Os conjuntos fuzzy para cada variável de entrada consistem de sete variáveis linguísticas: e = {NB, NM, NS, ZO, PS, PM, PB}, e_c = {NB, NM, NS, ZO, PS, PM, PB}. As variáveis linguísticas das saídas são atribuídas da seguinte forma: Δkp = {ZO, PS, PM, PB}, Δki = {ZO, PS, PM, PB}, Δkp = {ZO, PS, PM, PB}. As regras de inferência Fuzzy para as variáveis de saída Δkp, Δki e Δkd são mostradas nos Quadros 2, 3 e 4 respectivamente.

Quadro 2 – Regras de inferência fuzzy usadas em (GAO et al., 2014b) para a variável ΔK_p

	ΔE_c						
E	**NB**	**NN**	**NS**	**ZO**	**PS**	**PM**	**PB**
NB	PB	PM	ZO	ZO	ZO	PM	PB
NM	PB	PM	ZO	ZO	ZO	PM	PB
NS	PB	PB	PS	ZO	PS	PB	PB
ZO	PB	PB	PS	ZO	PS	PB	PB
PS	PB	PB	PS	ZO	PS	PB	PB
PM	PB	PM	ZO	ZO	ZO	PM	PB
PB	PB	PM	ZO	ZO	ZO	PM	PB

Fonte: Adaptado de Gao et al. (2014b)

Quadro 3 – Regras de inferência fuzzy usadas em (GAO et al., 2014b) para a variável ΔK_i

	ΔE_c						
E	**NB**	**NN**	**NS**	**ZO**	**PS**	**PM**	**PB**
NB	ZO	ZO	ZO	PS	ZO	ZO	ZO
NM	ZO	ZO	ZO	PS	ZO	ZO	ZO
NS	ZO	ZO	PS	PS	PS	ZO	ZO
ZO	ZO	ZO	PS	PS	PS	ZO	ZO
PS	ZO	ZO	PS	PS	PS	ZO	ZO
PM	ZO	ZO	ZO	PS	ZO	ZO	ZO
PB	ZO	ZO	ZO	PS	ZO	ZO	ZO

Fonte: Adaptado de Gao et al. (2014b)

O controlador proposto alcançou sucesso nos testes realizados, mostrando permitir completar a trajetória sem que perturbações afetem a precisão de controle. O Sistema de Inferência Fuzzy se mostrou eficiente para ajustar os ganhos do controlador PID de forma a possibilitar o controle eficiente do quadrotor.

Em outro trabalho, os mesmos autores implementaram um outro controlador híbrido. Em (GAO et al., 2014a), foi proposto um controlador Fuzzy PD. Além disto, neste trabalho, são comparadas as performances de três controladores diferentes: PID, Fuzzy PD e *Backstepping*. Mais uma vez, aqui o controlador fuzzy tem como função ajustar os parâmetros do controlador PD: ΔK_p e ΔK_d.

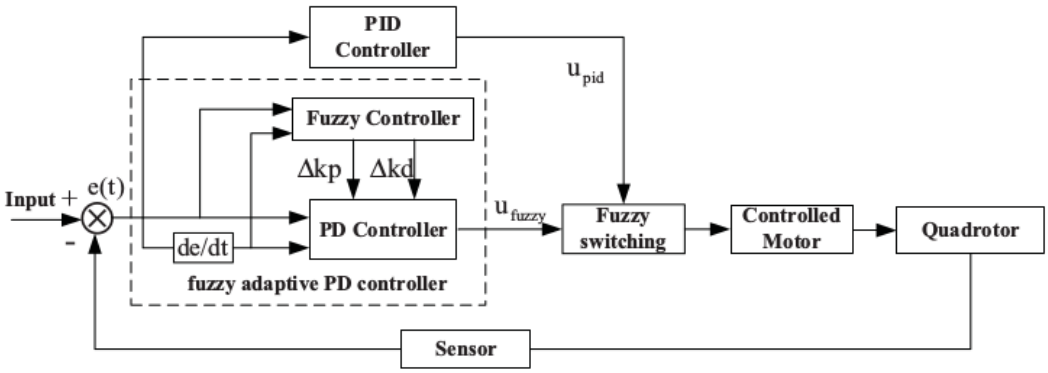
Quadro 4 – Regras de inferência fuzzy usadas em (GAO et al., 2014b) para a variável ΔK_d

	ΔE_c						
E	**NB**	**NN**	**NS**	**ZO**	**PS**	**PM**	**PB**
NB	PB	PB	PM	PS	PM	PB	PB
NM	PB	PB	PM	PS	PM	PB	PB
NS	PB	PM	PS	ZO	PS	PM	PB
ZO	PB	PM	PS	ZO	PS	PM	PB
PS	PB	PM	PS	ZO	PS	PM	PB
PM	PB	PB	PM	ZO	PM	PB	PB
PB	PB	PB	PM	PS	PM	PB	PB

Fonte: Adaptado de Gao et al. (2014b)

A Figura 20 mostra o diagrama de blocos do controlador neste caso. Como se pode ver, o diagrama mostra o controlador Fuzzy PD e um controlador PID.

Figura 20 – Diagrama de blocos do controlador híbrido usando em (GAO et al., 2014a)



Fonte: Gao et al. (2014a)

Em relação à estrutura fuzzy, mais uma vez aqui há duas entradas para a inferência: o erro, sua derivada e os conjuntos fuzzy para cada uma delas consistem de sete variáveis linguísticas: e = {NB, NM, NS, ZO, PS, PM, PB}, e_c = {NB, NM, NS, ZO, PS, PM, PB}. A variável de saída do sistema Fuzzy PD é apenas uma: U_{fuzzy} = {ZO, PS, PM, PB}. O Quadro 5 mostra as regras definidas para o sistema de inferência fuzzy.

O controlador Fuzzy PD é adotado para garantir uma supressão rápida e com sobrelevação quando o desvio é grande. Já o controlador PID é adotado para eliminar o estado estacionário quando o desvio é pequeno.

A definição de qual dos controles irá atuar, depende do bloco *Fuzzy switching*, que avalia os sinais u_{pid} e u_{fuzzy}, e utiliza o mais apropriado entre eles. A regra utilizada é a seguinte: se $E(k)$ é SE e $\Delta EC(k)$ é $S\Delta EC$ então U é U_{pid}, senão U é U_{fuzzy}, em que

Quadro 5 – Regras de inferência fuzzy usadas em (GAO et al., 2014a)

	ΔE_c						
E	**NB**	**NN**	**NS**	**ZO**	**PS**	**PM**	**PB**
NB	PB	PM	PM	PS	PM	PM	PB
NM	PB	ZO	ZO	PS	ZO	ZO	ZO
NS	ZO	ZO	PS	ZO	PM	PB	ZO
ZO	PM	PB	PS	ZO	ZO	PM	PB
PS	PS	PB	PS	PM	PS	ZO	PB
PM	PB	ZO	ZO	PM	ZO	PM	ZO
PB	ZO	PM	PM	ZO	PM	PM	PM

Fonte: Adaptado de Gao et al. (2014a)

U_{fuzzy} e U_{pid} são as saídas dos controladores Fuzzy PD e PID respectivamente. SE e $S\Delta EC$ são, respectivamente, funções de pertinência fuzzy das variáveis E e ΔE_c.

Nas simulações feitas, foram comparados três controladores diferentes: *Backstepping*, PID e Fuzzy PD. Foi verificado que, embora o controlador *Backstepping* seja pior que os outros dois, ele pode rapidamente suprimir o impacto de perturbações. Foi mostrado ainda que o controlador Fuzzy PD obteve o melhor resultado relacionado a rejeição de perturbações. O tempo de subida do controlador Fuzzy PD é ligeiramente menor nos eixos x e z mas um pouco maior no eixo y. Após numerosas simulações, verificou-se que o controlador proposto é, de fato, eficaz para manter a estabilidade de um helicóptero quadrotor.

Outras abordagens híbridas ainda foram adotadas para o projeto de controladores de estabilidade para *drones*. Em (YACEF et al., 2013), foi proposto o uso do PSO para ajuste de um IBC. O método é usado para minimizar o erro quadrático (SE) de uma função de custo que quantifica a performance de todo o sistema. Resultados de numerosas simulações mostram a validação e as boas performances alcançadas pelo método proposto.

Em (BOUBERTAKH et al., 2013) é proposta uma ideia parecida: utilizar o algoritmo PSO para ajustar os pesos de quatro controladores PD, cada qual responsável por controlar um rotor. A ideia é usar o PSO para minimizar o SE da função de custo que quantifica o sistema, assim como visto em (YACEF et al., 2013). As simulações feitas mostraram a eficácia do controlador proposto.

Como se pode ver, as técnicas inteligentes têm sido largamente utilizadas para propor diferentes tipos de controladores para quadrotores. Como já dito anteriormente, isso deve às inúmeras vantagens citadas que elas trazem. Algumas dessas técnicas foram utilizadas neste trabalho, como é mostrado a seguir.

5 Metodologia

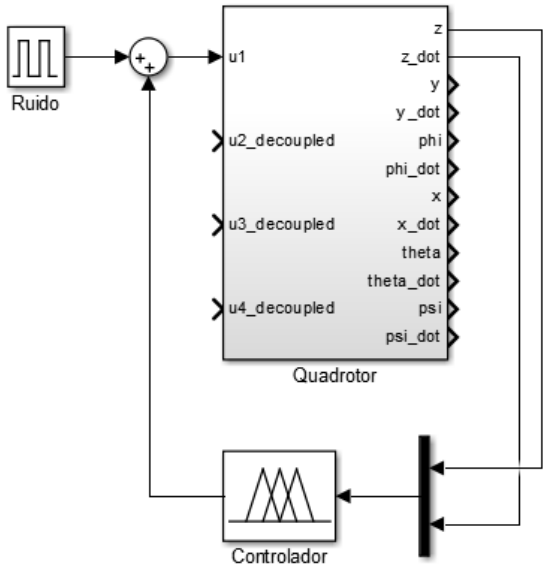
Em ambiente simulado, utilizando a modelagem computacional de um quadrotor com desacoplamento das entradas proposto por Balas (2007) e mostrado na seção 3.2, foram propostos, para estabilizar o sistema, controladores aplicando duas técnicas diferentes de IC: fuzzy e neuro-fuzzy. Todas as simulações e execuções se deram utilizando a versão R2015a do MATLAB.

5.1 Controladores Fuzzy

O projeto dos controlador fuzzy foi focado na estabilização de atitude e altitude de um drone sujeito a parâmetros específicos[1], sendo eles: a gravidade $g = 9{,}81\ m/s^2$, a massa do quadrotor $m = 2{,}3\ kg$ e o comprimento de cada haste do drone $l = 0{,}5\ m$. Um dos controladores projetado tem, como objetivo, controlar a altitude do drone, ao passo que um segundo visa a estabilização de sua atitude.

O controlador de altitude possui duas entradas e uma saída. As entradas são referentes à posição vertical do quadrotor (z) e sua respectiva velocidade ($\dot{z}$), ao passo que a saída diz respeito ao sinal de controle a ser aplicado sobre o sistema para estabilizar sua altitude (u_1). Sua aplicação no sistema é mostrada na Figura 21.

Figura 21 – Diagrama do sistema de controle de altitude utilizando controlador fuzzy



Utilizando o *Fuzzy Logic Toolbox* do MATLAB, cada variável linguística do controlador fuzzy foi divida em três conjuntos: N (negativo), Z (zero) e P (positivo), baseado

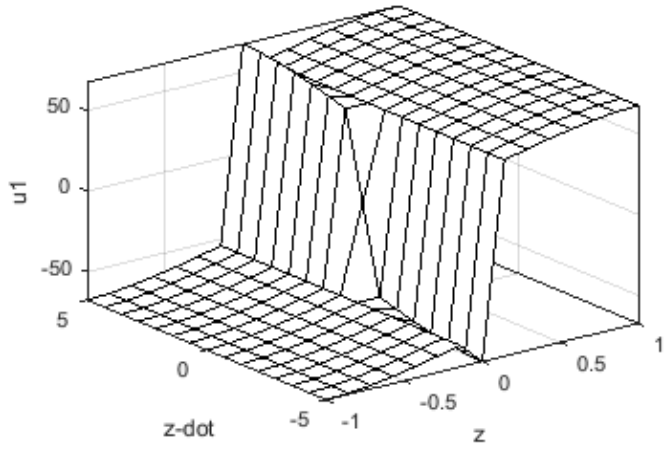
[1] Estes valores foram também utilizados por Balas (2007) para definir uma especificação do sistema

nos trabalhos de Maj e Butkiewicz (2013) e Gao et al. (2014b). As regras fuzzy definidas para este controlador são mostradas no Quadro 6, e a Figura 22 exibe seu equivalente em superfície.

Quadro 6 – Regras fuzzy para modelagem do controle de altitude

z	$\dot{z}$	u_1
N	-	N
P	-	P
Z	N	N
Z	Z	Z
Z	P	P

Figura 22 – Superfície das regras do sistema de controle fuzzy para a altitude do quadrotor



O controlador de atitude projetado também possui duas entradas e uma saída. Desta vez, entretanto, as entradas são referentes ao ângulo em relação ao eixo horizontal (ϕ ou θ) e sua respectiva variação ($\dot{\phi}$ ou $\dot{\theta}$). Mais uma vez, cada variável linguística foi dividida em três conjuntos: N, Z e P.

As regras que regem o controlador de atitude são sintetizadas no Quadro 7 e podem ser vistas na superfície de regras mostradas na Figura 23.

Já as Figuras 24 e 25 exibem os diagramas do sistema controlado, com atuação sobre os ângulos ϕ e θ, respectivamente.

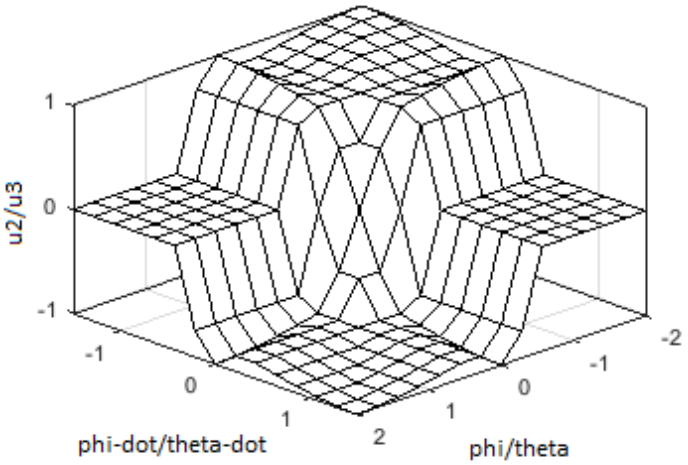
5.2 Controladores Neuro-Fuzzy

A partir dos controladores de atitude e altitude fuzzy projetados, foram propostos dois controladores do tipo neuro-fuzzy: um para cada dos casos.

Quadro 7 – Regras fuzzy para modelagem do controle de atitude

ϕ/θ	$\dot{\phi}/\dot{\theta}$	u_2/u_3
P	P	N
P	Z	N
P	N	Z
N	N	P
N	Z	P
N	P	Z
Z	Z	Z
Z	N	P
Z	P	N

Figura 23 – Superfície das regras do sistema de controle fuzzy para a atitude do quadrotor



Para tanto, foram utilizados os códigos mostrados nos Apêndices A e B. No processo de criação do controlador de altitude neuro-fuzzy, foram gerados trezentos[2] pares de entradas e cada um deles foi submetido ao processo de inferência fuzzy utilizando o controlador previamente modelado. Dois terços desses dados foram utilizados para gerar o conjunto de treinamento, representado pela variável `train` e o um terço restante foi armazenado na variável `test` e utilizado para validação do treinamento. Então, utilizando o comando `mam2sug` do MATLAB, foi gerado um modelo fuzzy Sugeno a partir do Mamdani que havia sido modelado e este novo arquivo foi salvo sob o nome `fis_altitude_neuro.fis`.

Feito isto, utilizou-se o comando comando `anfisedit` para abrir o *Neuro-Fuzzy Designer* do MATLAB, cuja interface é mostrada na Figura 26. No campo marcado pelo número 2 na imagem (*Generate FIS*), clicou-se no botão *Load* e se selecionou o arquivo `fis_altitude_neuro.fis` que fora gerado pelo código executado. Após isto, no

[2] Este valor foi arbitrado por corresponder a uma quantidade razoável para treinar a RNA sem que se alcance o sobreajuste, conhecido como *overfitting*

Figura 24 – Diagrama do sistema de controle de atitude utilizando controlador fuzzy sobre o ângulo ϕ

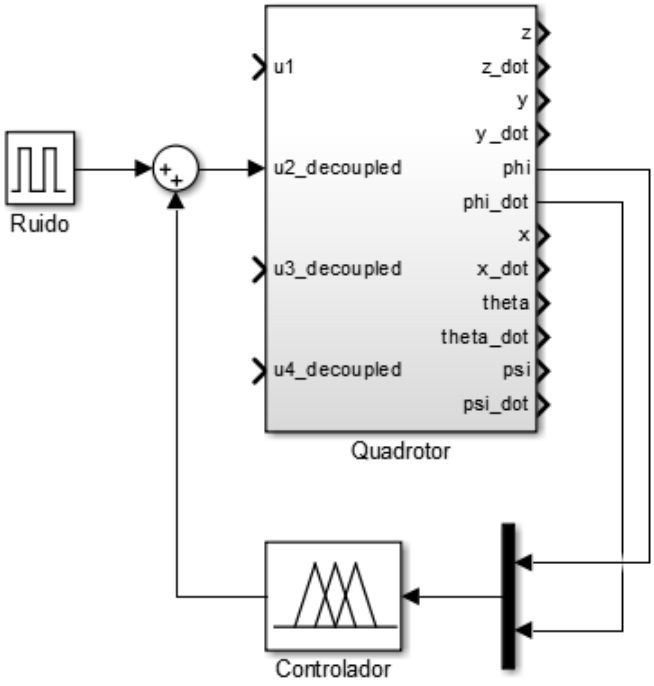
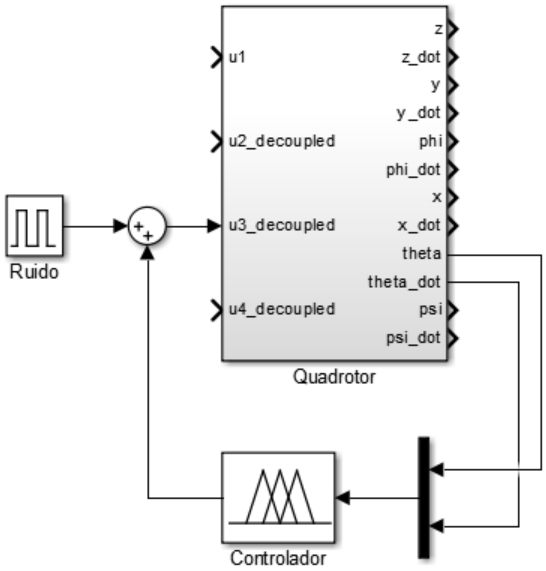


Figura 25 – Diagrama do sistema de controle de atitude utilizando controlador fuzzy sobre o ângulo θ



campo marcado pelo número 1 (*Load Data*), marcou-se *Training* e *worksp* para utilizar uma variável da área de trabalho do MATLAB para treinar a rede. Após clicar em *Load Data*, digitou-se `train`, nome da variável definida no código. Então, no campo marcado pelo número 3, marcou-se *Training Data* e se clicou no botão *Test Now* para executar o treinamento da rede. Após estes passos, a rede neuro-fuzzy foi devidamente treinada e

sua estrutura, mostrada na Figura 27, pode ser obtida clicando no botão *Structure* logo acima do campo 3. Esta estrutura relaciona as variáveis de entrada e suas funções de pertinência, através das regras fuzzy, à saída do sistema e às suas funções de pertinência, em que cada componente representa um neurônio da RNA obtida.

Figura 26 – Interface gráfica da ferramenta *Neuro-Fuzzy Designer* com destaque aos três campos necessários para treinamento e teste da rede neuro-fuzzy

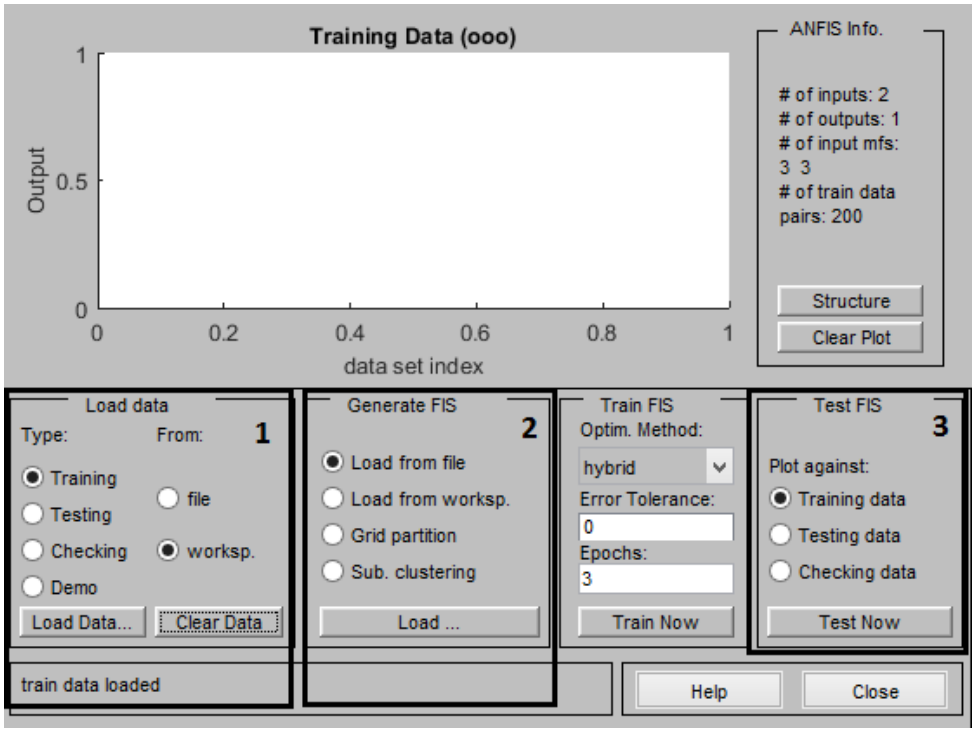
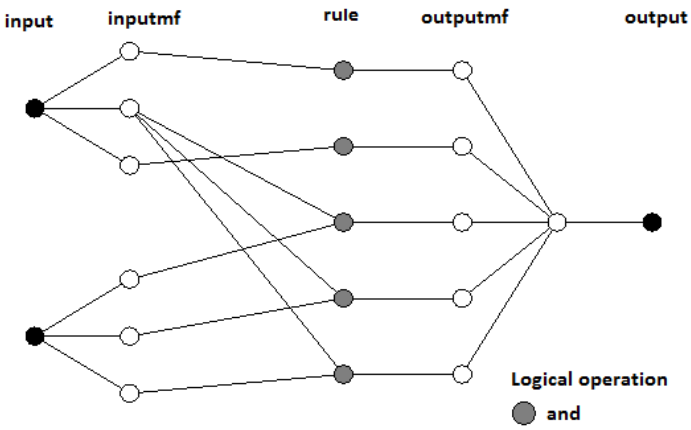


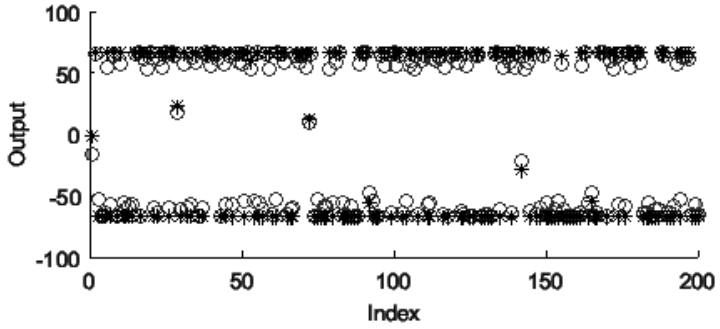
Figura 27 – Diagrama da RNA referente ao controlador neuro-fuzzy para altitude



Após o término do treinamento, deve-se submeter a rede ao processo de treinamento. Para tanto, basta selecionar *Testing* no campo marcado pelo número 1, deixar marcada a opção *worksp*, clicar no botão *Load Data* e escolher a variável `test`, que também foi definida no código executado.

A Figura 28 mostra o gráfico obtido na ferramenta após o processo de treinamento, em que os círculos brancos mostram os dados utilizados no treinamento e os asteriscos pretos indicam o valor referentes a eles obtidos pela rede treinada.

Figura 28 – Resultado obtido pelo treinamento da RNA para controle de altitude



Um processo similar foi aplicado para modelar o controlador de atitude neuro-fuzzy, como mostra o Apêndice B. As Figuras 29 e 30 mostram o diagrama da RNA referente ao controlador neuro-fuzzy para atitude e o resultado obtido pelo seu treinamento respectivamente.

Figura 29 – Diagrama da RNA referente ao controlador neuro-fuzzy para atitude

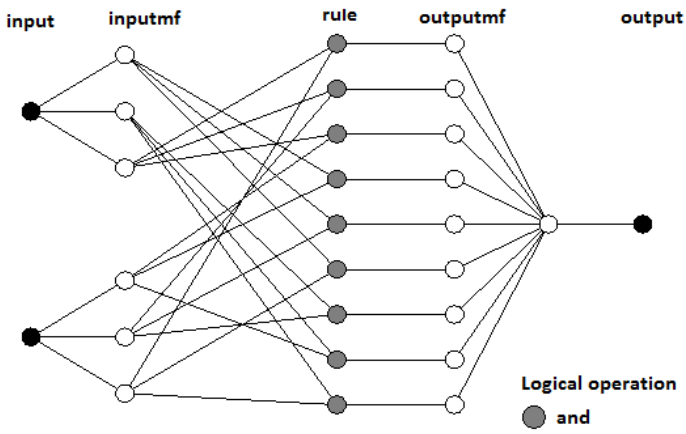
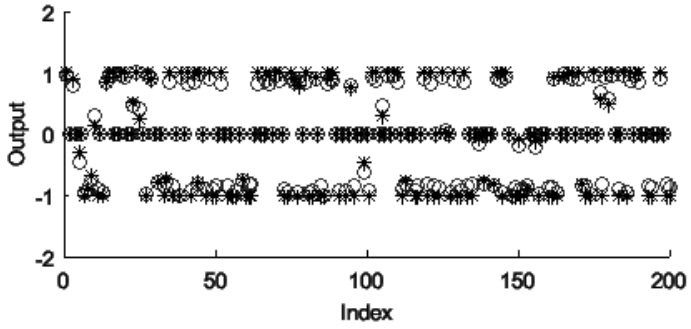


Figura 30 – Resultado obtido pelo treinamento da RNA para controle de atitude



O processo de treinamento determina o comportamento dos controladores neuro-fuzzy projetados, cujas superfícies de regras são exibidas nas Figuras 31 e 32.

Figura 31 – Superfície das regras do sistema de controle neuro-fuzzy para a altitude do quadrotor

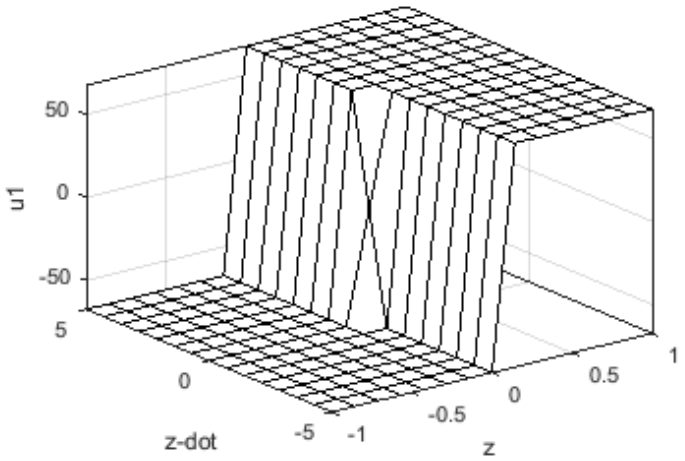
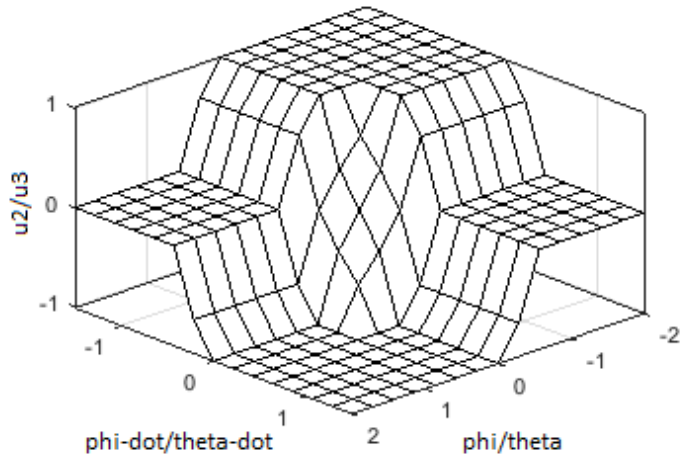


Figura 32 – Superfície das regras do sistema de controle fuzzy para a atitude do quadrotor



5.3 Experimentos Realizados

O primeiro experimento feito foi para mostrar a instabilidade do sistema, mostrando a resposta das variáveis relacionadas à atitude (ângulos ϕ e θ) e altitude (z) quando o sistema é submetido a um breve impulso em suas entradas, simulando qualquer força que possa atuar brevemente sobre o quadrotor, como um golpe sofrido por qualquer objeto que colida com ele. Após contextualizada a necessidade de controladores, passou-se à sua implementação e uso.

Uma vez projetados os controladores fuzzy e neuro-fuzzy, o sistema foi sujeitado a distúrbios em atitude e altitude para verificar o funcionamento deles sob condições

similares às mostradas quando nenhum controle agia sobre ele fazendo com que o sistema divergisse.. Primeiramente, o comportamento de ambos os controladores foi verificado quando atuando sobre o sistema para os quais eles foram projetados, com $g = 9{,}81\ m/s^2$, $m = 2{,}3\ kg$ e $l = 0{,}5\ m$.

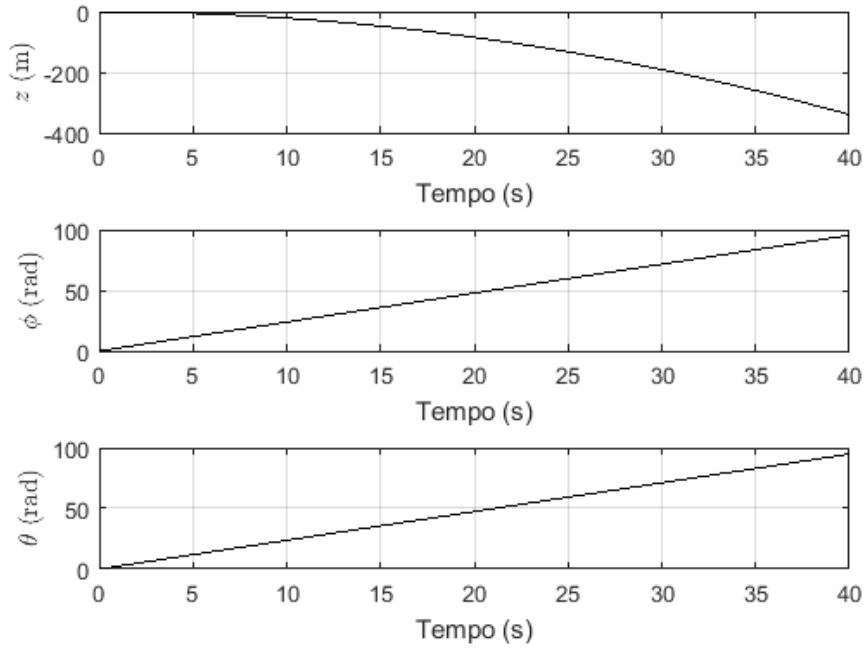
Em seguida, para testar a robustez de cada controlador, foi feita uma simulação em que eles atuam sobre um sistema cuja massa do quadrotor é $m = 5\ kg$, valor este que foi escolhido por variar o parâmetro massa em mais de 100 %.

Os resultados obtidos são mostrados no capítulo seguinte.

6 Resultados

Ao aplicar um impulso nas entradas do sistema modelado para representar o quadrotor, os estados relacionados à altitude e atitude divergem, como é mostrado na Figura 33. Nota-se que a divergência dos ângulos (ϕ e θ) é linear, mostrando que o impulso aplicada na entrada é, de fato, a aceleração do sistema sobre essas saídas. Já a variável relacionada à atitude (z) decai em parábola, devido à ação da gravidade, o que indica que o impulso é também, de fato, uma aceleração extra do sistema no eixo z, como a modelagem adotada propõe.

Figura 33 – Divergência das variáveis relacionadas a altitude e atitude quando o sistema é submetido a um impulso nas entradas



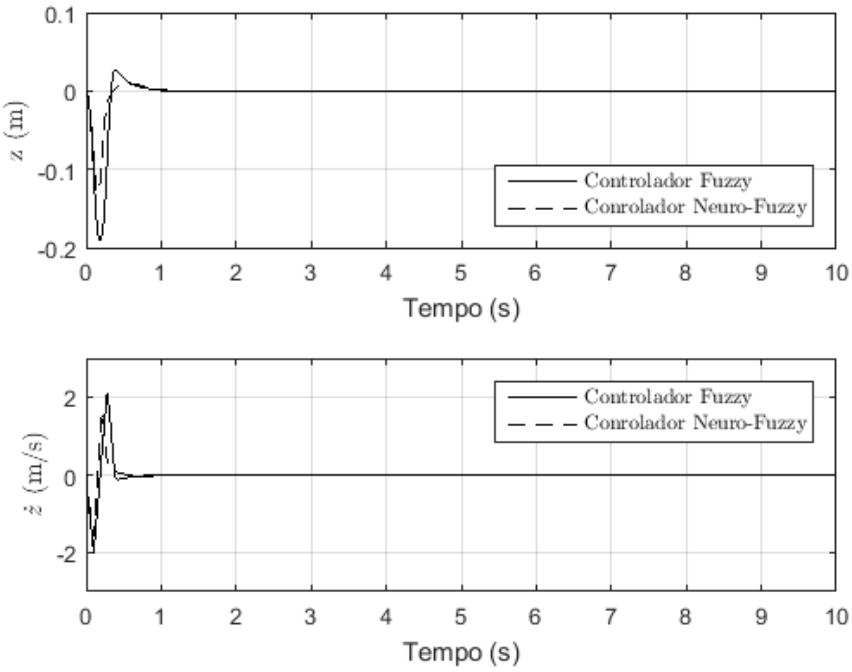
Com estes resultados, fica clara a necessidade de controladores para estabilizar o sistema. Os próximos resultados mostram a resposta do sistema estabilizado pelos controladores fuzzy e neuro-fuzzy projetados.

A Figura 34 mostra a posição no eixo vertical (z) do drone, bem como sua variação no sistema em que atua o controlador fuzzy projetado. Como se pode ver, o distúrbio foi devidamente controlado, fazendo com que o quadrotor retornasse à posição inicial $z = 0$ e também ao repouso[1] representado por $\dot{z} = 0$. Nesta figura, entretanto, não fica tão clara a diferença de desempenho dos controladores fuzzy e neuro-fuzzy, aspecto que pode ser claramente verificado na Figura 35. Como se pode ver, tanto para z quanto

[1] i.e. velocidade nula

para $\dot{z}$, o neuro-fuzzy apresenta desempenho melhor. No controle sobre a posição z, o controlador neuro-fuzzy apresentou redução do tempo de convergência em 29%, e da variação do sistema em 31%, além de eliminar a sobrelevação apresentada pelo fuzzy. Já sobre a velocidade $\dot{z}$, apresentou uma redução no tempo de convergência de 29% além melhorar a variação do sistema em 23%. A partir destes resultados, verifica-se que o controlador neuro-fuzzy fez com que o distúrbio fosse melhor absorvido e que sua correção ocorresse mais rapidamente.

Figura 34 – Comparação da resposta das saídas z e $\dot{z}$ no controle de altitude fuzzy e neuro-fuzzy para o sistema com massa $m = 2kg$



Já as Figuras 36 e 37 mostram a estabilidade de atitude em torno dos eixos x e y (i.e em relação ao plano horizontal XY), representados por θ e ϕ respectivamente. Como se pode ver, ambos os estados são devidamente controlados e, com isto, o quadrotor volta à estabilidade horizontal, com ângulos e velocidades angulares nulas no estado permanente.

A partir das Figuras 38 e 39, que mostram as respostas obtidas em mais detalhes, pode-se ver que, mais uma vez o controlador neuro-fuzzy mais uma vez teve desempenho superior ao fuzzy, fazendo com que os ângulos ϕ e θ convergissem 2% mais rápido, além de reduzir suas variações em 13%. Com relação às velocidades angulares $\dot{\phi}$ e $\dot{\theta}$, foi capaz de reduzir o tempo de convergência em 3%, não afetando a variação nem a sobrelevação apresentada pelo sistema quando estabilizado pelo controlador fuzzy. Desta forma, verifica-se que o controle neuro-fuzzy levou o sistema a uma menor variação, representando que o ângulo máximo de inclinação alcançado pelo drone é

Figura 35 – Comparação em mais detalhes da resposta das saídas z e $\dot{z}$ no controle de altitude fuzzy e neuro-fuzzy para o sistema com massa $m = 2kg$

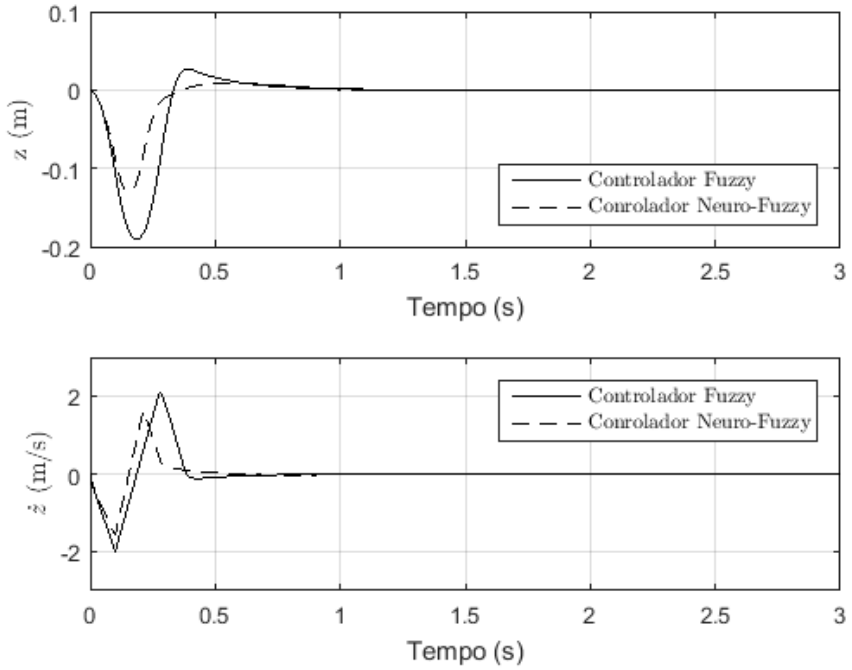
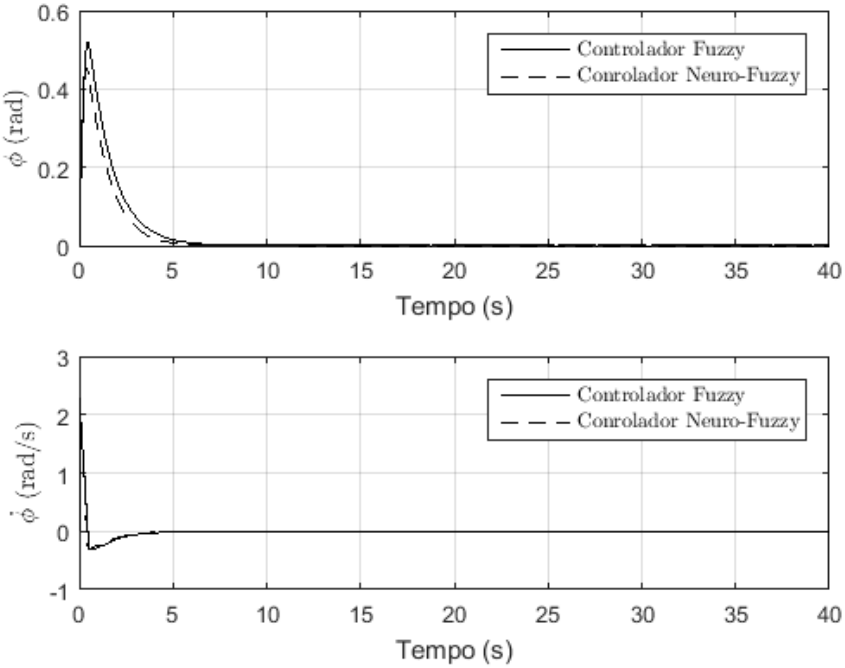


Figura 36 – Comparação da resposta das saídas ϕ e $\dot{\phi}$ no controle de atitude fuzzy e neuro-fuzzy para o sistema com massa $m = 2kg$



menor e corrigido mais rapidamente.

Além da verificação da eficiência dos controladores atuando sobre o sistema para o qual foram projetados, eles foram testados num sistema em que um dos parâmetros

Figura 37 – Comparação da resposta das saídas θ e $\dot{\theta}$ no controle de atitude fuzzy e neuro-fuzzy para o sistema com massa $m = 2kg$

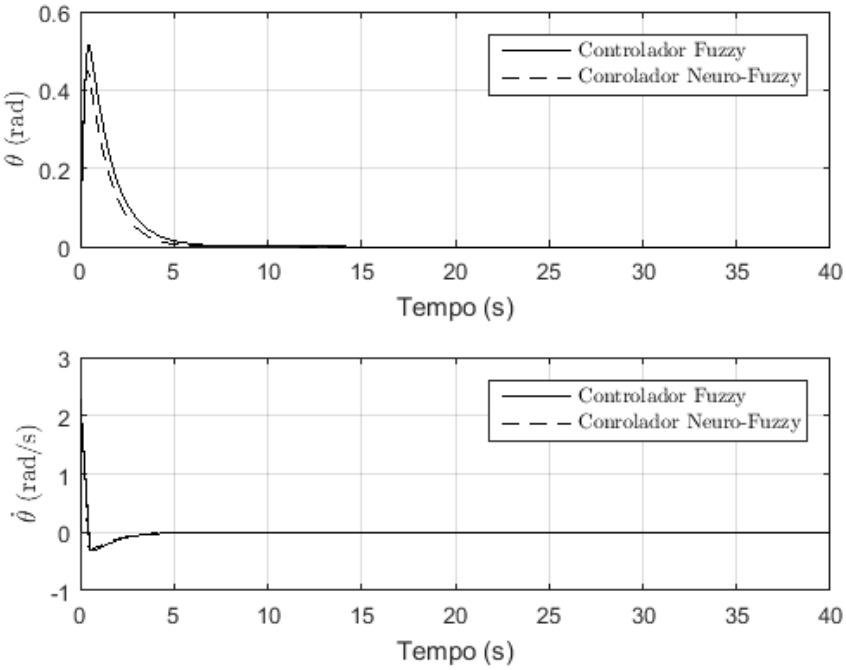
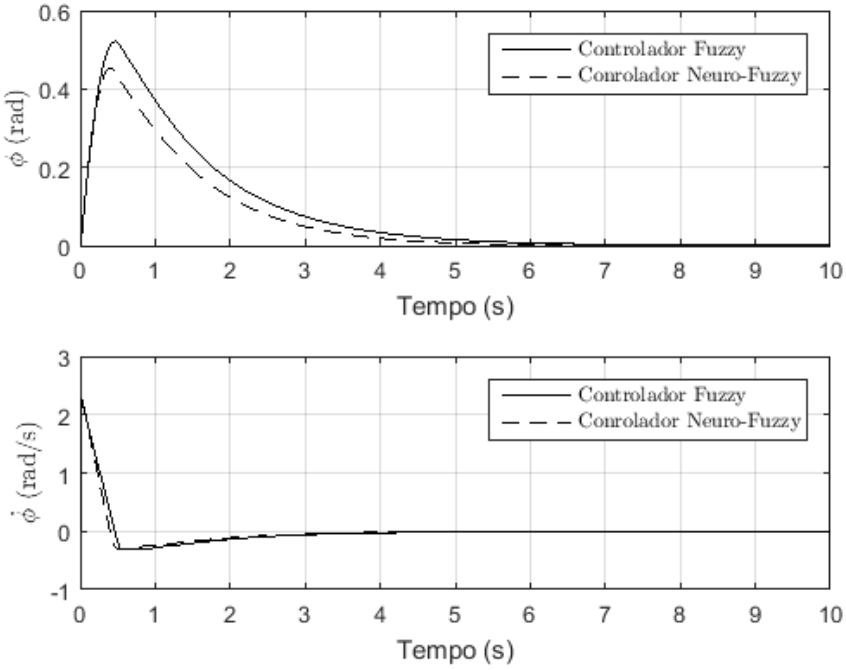


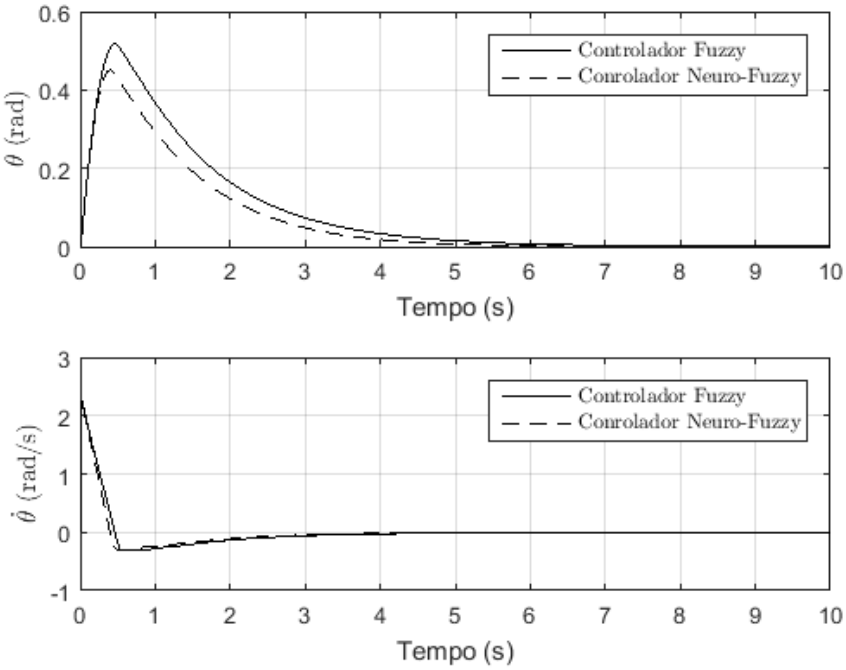
Figura 38 – Comparação em mais detalhes da resposta das saídas ϕ e $\dot{\phi}$ no controle de atitude fuzzy e neuro-fuzzy para o sistema com massa $m = 2kg$



foi acrescido de mais de 100 %, com a massa passando de 2,3 kg para 5 kg.

A resposta dos controladores fuzzy e neuro-fuzzy para altitude do drone nessas circunstâncias são mostradas nas Figuras 40 e 41, sendo que esta segunda é apenas

Figura 39 – Comparação em mais detalhes da resposta das saídas θ e $\dot{\theta}$ no controle de atitude fuzzy e neuro-fuzzy para o sistema com massa $m = 2kg$



uma forma melhor de comparar a ação dos dois controladores. Como se pode perceber, ambos os controladores levaram à estabilização do sistema, sendo que desta vez cada um obteve desempenho melhor sob determinados aspectos. No controle da posição vertical z, o neuro-fuzzy apresentou tempo de convergência 57% maior, em parte causado por uma sobrelevação, que não foi apresentada pelo fuzzy. Em contrapartida, o neuro-fuzzy apresentou uma menor variação, a reduzindo em 20% se comparado ao fuzzy. Com relação à velocidade $\dot{z}$, o controlador neuro-fuzzy apresentou aumento de 23% no tempo de convergência, mas melhorou o sistema nos quesitos variação e sobrelevação, as reduzindo em 16% e 33%, respectivamente. Esses resultados apontam que o quadrotor, quando submetido ao controle neuro-fuzzy, apresentou movimentos mais suaves até ter sua altitude estabilizada, apesar de ter sido necessário mais tempo para que ela ocorresse.

Por fim, as Figuras 42 e 43 mostram os resultados obtidos pelos controladores de atitude no sistema com massa $m = 5\ kg$. Percebe-se que mais uma vez o sistema convergiu ao seu estado de estabilidade com os ângulos nulos e velocidades angulares também nulas, representando que o quadrotor, após a ação de controle, tanto fuzzy quanto neuro-fuzzy, fica estável e com orientação plana[2].

As Figuras 44 e 45 mostram em mais detalhes as repostas obtidas pelos controladores sobre a atitude do sistema com massa $m = 5\ kg$. A partir delas, nota-se que mais

[2] i.e paralela ao plano XY

Figura 40 – Comparação da resposta das saídas z e $\dot{z}$ no controle de altitude fuzzy e neuro-fuzzy para o sistema com massa $m = 5kg$

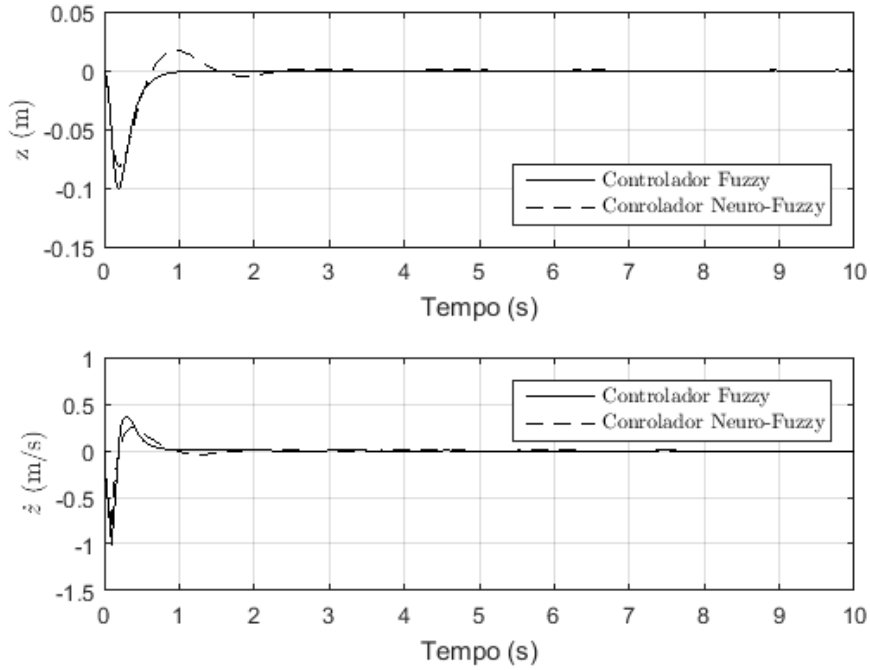
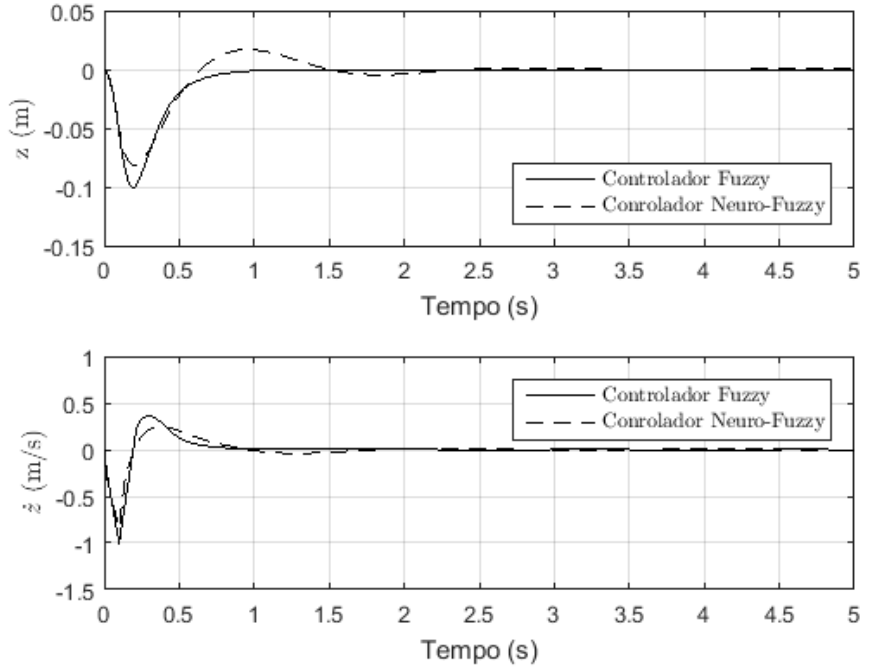


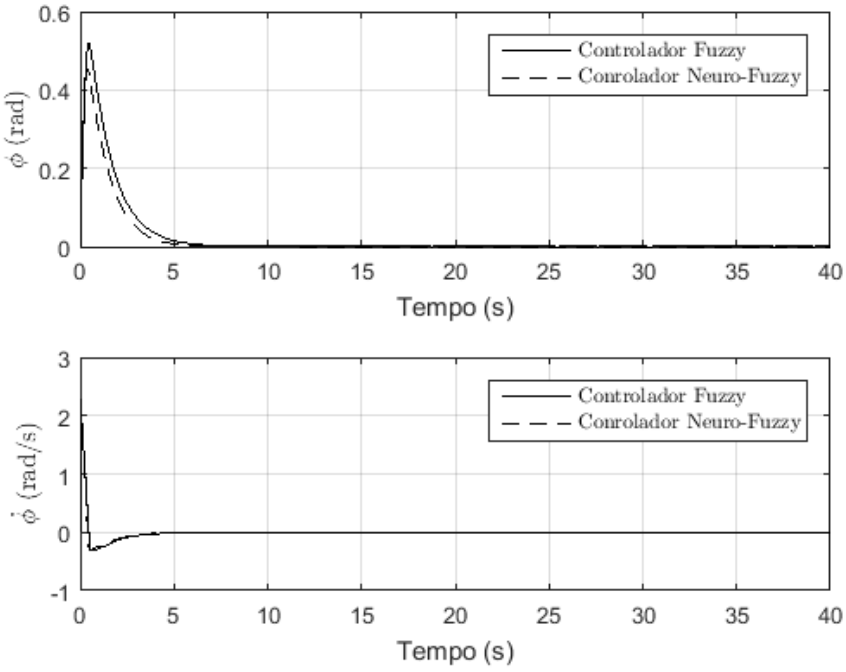
Figura 41 – Comparação em mais detalhes da resposta das saídas z e $\dot{z}$ no controle de altitude fuzzy e neuro-fuzzy para o sistema com massa $m = 5kg$



uma vez o controlador neuro-fuzzy apresentou desempenho levemente superior ao fuzzy. Sobre os ângulos ϕ e θ, a convergência ocorreu 3% mais rapidamente e a variação apresentada reduziu 14%. Já sobre as velocidades angulares $\dot{\phi}$ e $\dot{\theta}$, a redução de tempo de

convergência com o neuro-fuzzy foi de 2%, mantendo a mesma sobrelevação e variação oferecidas pelo fuzzy. Com isto, mais uma vez o controle neuro-fuzzy fez com que o ângulo máximo de inclinação do quadrotor fosse inferior ao alcançado pelo sistema controlado pelo fuzzy,e além de reduzir o tempo necessário para sua estabilização definitiva.

Figura 42 – Comparação da resposta das saídas ϕ e $\dot{\phi}$ no controle de atitude fuzzy e neuro-fuzzy para o sistema com massa $m = 5kg$



6.1 Análise Consolidada dos Resultados

Como se pode ver, os controladores de atitude e altitude tanto fuzzy quanto neuro-fuzzy foram eficientes levando à estabilização do sistema em todos os casos testados, inclusive na situação em que a massa do sistema foi aumentada em mais de 100 %. Isso indica que estes controladores podem ser utilizados, por exemplo, em situações em que o drone precisaria transportar uma carga que tenha sua massa ou até mesmo uma superior.

Em quase todos os casos, nota-se um comportamento do controlador neuro-fuzzy superior ao do fuzzy, o que já era esperado tendo em vista que o primeiro alia o poder do segundo às vantagens das RNAs, fazendo com que, a partir de um treinamento supervisionado utilizando o próprio modelo fuzzy, possa se construir um controle mais abrangente e com resposta melhorada.

Figura 43 – Comparação da resposta das saídas θ e $\dot{\theta}$ no controle de atitude fuzzy e neuro-fuzzy para o sistema com massa $m = 5kg$

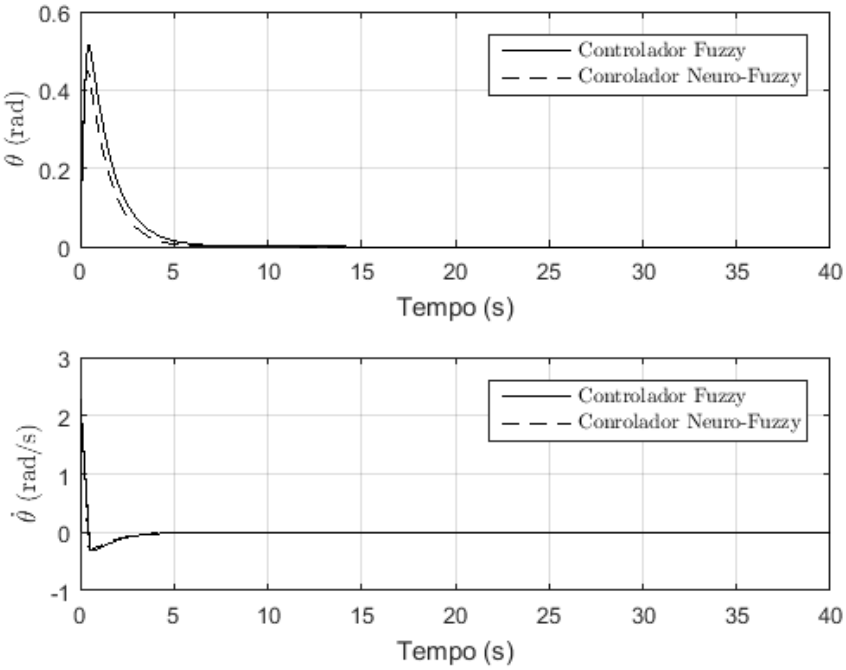
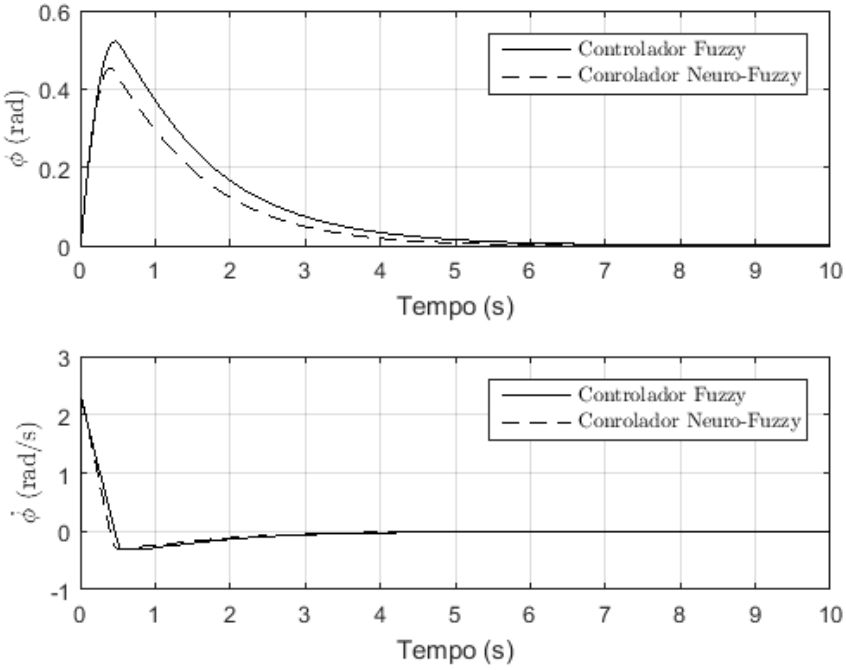
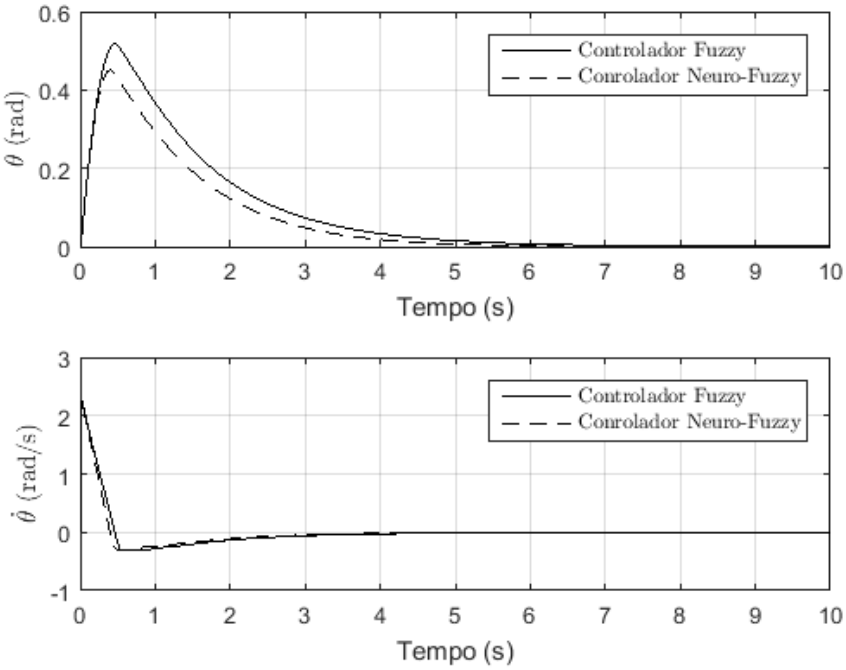


Figura 44 – Comparação da resposta das saídas ϕ e $\dot{\phi}$ no controle de atitude fuzzy e neuro-fuzzy para o sistema com massa $m = 5kg$



Estes resultados mostram que, de fato, as técnicas de Inteligência Computacional podem ser aplicadas para projetar controladores eficientes e robustos para atuar sobre sistemas multivariável e que, além disto, o poder de treinamento dos ANFISs realmente

Figura 45 – Comparação da resposta das saídas θ e $\dot{\theta}$ no controle de atitude fuzzy e neuro-fuzzy para o sistema com massa $m = 5kg$



é capaz de fazer com que o desempenho de controladores neuro-fuzzy seja melhorado se comparado a de controladores puramente fuzzy.

7 Conclusão

Ao longo deste trabalho, discorreu-se sobre o crescente uso de estratégias da Inteligência Computacional para implementar controladores de sistemas não lineares. Além disto, como foi mostrado pelos experimentos realizados, o uso de controladores devidamente projetados é fundamental para fazer com que esses sistemas instáveis atuem da forma planejada e possam ser estabilizados.

No contexto deste trabalho, o sistema controlado é um quadrotor e as variáveis são referentes à sua atitude e altitude, representando portanto um controle multivariável. As alternativas propostas como controladores foram o fuzzy e o neuro-fuzzy. A opção pelo primeiro se deveu ao fato de ele permitir a modelagem a partir de variáveis e termos linguísticos além de acrescer robustez ao sistema. Já a opção pelo segundo, neuro-fuzzy, foi devido ao fato de este agregar as características de RNAs aos sistemas fuzzy, possuindo um poder de generalização e aprendizado capaz de melhorar sua performance.

De fato, os resultados mostram que o controlador neuro-fuzzy realmente obteve melhor desempenho. No controle de altitude, reduziu o tempo de convergência em 29% e a variação do sistema em 31% além de eliminar a sobrelevação apresentada pelo fuzzy. O controle neuro-fuzzy de atitude também apresentou melhoras, apesar de não tão significativas quanto essas: reduziu o tempo de convergência em 2% e a variação do sistema em 13%.

Além disto, num teste para verificar a robustez dos controladores, a massa do sistema foi acrescida em 117%, passando de 2 kg para 5 kg. Neste novo contexto, o controlador de atitude neuro-fuzzy mais uma vez foi superior ao fuzzy, reduzindo o tempo de convergência em 3% e da variação em 14%. Já no controle de altitude, o único fator melhorado pelo neuro-fuzzy foi a variação do sistema, sendo reduzida em 20%, ao passo que seu tempo de convergência cresceu 57%, além de ter sido inserida uma sobrelevação na resposta.

Apesar das diferenças de desempenho, os controladores fuzzy e neuro-fuzzy tanto para atitude quanto para altitude do sistema foram capazes de estabilizá-lo, mesmo quando submetido a uma variação substancial de parâmetros, que foi representada pelo aumento da massa em mais de 100%, mostrando assim a robustez intrínseca a ambos os controladores.

Desta forma, mostra-se que se podem usar técnicas de IC para controlar, de forma robusta e eficiente, sistemas não-lineares complexos e, além disso, que controladores neuro-fuzzy podem ser utilizados para melhorar o desempenho de controladores fuzzy

apesar de, em algumas situações, piorar a resposta se comparado a estes.

7.1 Trabalhos futuros

Os resultados obtidos neste trabalho abrem espaço para se projetarem controladores equivalentes aos construídos ao longo dele para controlar um quadrotor real, extrapolando o cenário de simulações.

Referências

ADIGBLI, P. Nonlinear Attitude and Position Control of a Micro Quadrotor using Sliding Mode and Backstepping Techniques. **3rd US-European Competition and Workshop on Micro Air Vehicle Systems (MAV07) & European Micro Air Vehicle Conference and Flight Competition (EMAV2007), 17-21 September 2007, Toulouse, France**, n. September, p. 17–21, 2007. Citado na página 2.

AL-YOUNES, Y.; JARRAH, M. Attitude stabilization of quadrotor uav using backstepping fuzzy logic & backstepping least-mean-square controllers. In: **Mechatronics and Its Applications, 2008. ISMA 2008. 5th International Symposium on**. [S.l.: s.n.], 2008. p. 1–11. Citado na página 1.

BALAS, C. **Modeling and Linear Control of a Quadrotor**. Dissertação (Mestrado) — Cranfield University, Reino Unido, 2007. Citado 5 vezes nas páginas viii, 23, 24, 25 e 38.

BASRI, M.; HUSAIN, A.; DANAPALASINGAM, K. Fuzzy supervisory backstepping controller for stabilization of quadrotor unmanned aerial vehicle. In: **Intelligent and Advanced Systems (ICIAS), 2014 5th International Conference on**. [S.l.: s.n.], 2014. p. 1–5. Citado 2 vezes nas páginas 22 e 34.

BOUABDALLAH, S.; MURRIERI, P.; SIEGWART, R. Design and control of an indoor micro quadrotor. **IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004**, v. 5, 2004. ISSN 1050-4729. Citado na página 2.

BOUBERTAKH, H.; BENCHAREF, S.; LABIOD, S. Pso-based pid control design for the stabilization of a quadrotor. In: **Systems and Control (ICSC), 2013 3rd International Conference on**. [S.l.: s.n.], 2013. p. 514–517. Citado na página 37.

BOUDJEDIR, H. Adaptive Neural Network for a Quadrotor Unmanned Aerial Vehicle. **International Journal in Foundations of Computer Science & Technology**, v. 2, n. 4, p. 1–13, 2012. ISSN 18397662. Citado na página 2.

COZA, C.; MACNAB, C. A new robust adaptive-fuzzy control method applied to quadrotor helicopter stabilization. In: **Fuzzy Information Processing Society, 2006. NAFIPS 2006. Annual meeting of the North American**. [S.l.: s.n.], 2006. p. 454–458. Citado na página 32.

DORF, R. **Modern control systems**. 12. ed. Upper Saddle River, N.J: Pearson Prentice Hall, 2011. ISBN 978-0136024583. Citado 4 vezes nas páginas 4, 5, 6 e 7.

GAO, Q.; YUE, F.; HU, D. Research of precision flight control for quadrotor uav. In: **Guidance, Navigation and Control Conference (CGNCC), 2014 IEEE Chinese**. [S.l.: s.n.], 2014. p. 2369–2374. Citado 5 vezes nas páginas viii, xi, 35, 36 e 37.

GAO, Q.; YUE, F.; HU, D. Research of stability augmentation hybrid controller for quadrotor uav. In: **Control and Decision Conference (2014 CCDC), The 26th Chinese**. [S.l.: s.n.], 2014. p. 5224–5229. Citado 7 vezes nas páginas viii, xi, 22, 34, 35, 36 e 39.

GUO, C.; SIMAAN, M.; SUN, Z. Neuro-fuzzy intelligent controller for ship roll motion stabilization. In: **Intelligent Control. 2003 IEEE International Symposium on**. [S.l.: s.n.], 2003. p. 182–187. ISSN 2158-9860. Citado na página 28.

HAYKIN, S. **Neural Networks: A Comprehensive Foundation**. NJ, USA: Prentice Hall PTR Upper Saddle River, 1998. ISBN 0132733501. Citado 2 vezes nas páginas 8 e 9.

JANG, J.; SUN, C.; MIZUTANI, E. **Neuro-fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence**. [S.l.]: Prentice Hall, 1997. ISBN 9780132610667. Citado 10 vezes nas páginas 10, 11, 12, 13, 14, 15, 16, 17, 18 e 19.

KHATOON, S.; SHAHID, M.; IBRAHEEM; CHAUDHARY, H. Dynamic modeling and stabilization of quadrotor using pid controller. In: **Advances in Computing, Communications and Informatics (ICACCI, 2014 International Conference on**. [S.l.: s.n.], 2014. p. 746–750. Citado na página 29.

MAHFOUZ, M.; ASHRY, M.; ELNASHAR, G. Design and Control of Quad-Rotor Helicopters Based on Adaptive Neuro-Fuzzy Inference System. v. 2, n. 12, p. 479–485, 2013. ISSN 2278-0181. Citado na página 30.

MAJ, W.; BUTKIEWICZ, B. Flying n-copter with fuzzy logic control. In: **Signal Processing Symposium (SPS), 2013**. [S.l.: s.n.], 2013. p. 1–6. Citado 5 vezes nas páginas viii, xi, 31, 32 e 39.

MUSTAPA, Z.; SAAT, S.; HUSIN, S.; ABAS, N. Altitude controller design for multicopter uav. In: **Computer, Communications, and Control Technology (I4CT), 2014 International Conference on**. [S.l.: s.n.], 2014. p. 382–387. Citado na página 29.

NIROUMAND, F.; FAKHARIAN, A.; SEYEDSAJADI, M. Fuzzy integral backstepping control approach in attitude stabilization of a quadrotor uav. In: **Fuzzy Systems (IFSC), 2013 13th Iranian Conference on**. [S.l.: s.n.], 2013. p. 1–6. Citado na página 33.

OGATA, K. **Modern Control Engineering**. 5. ed. [S.l.]: Prentice Hall, 2010. ISBN 978-0136156734. Citado 3 vezes nas páginas 6, 20 e 21.

OLIVARES-MENDEZ, M.; CAMPOY, P.; MELLADO-BATALLER, I.; MEJIAS, L. See-and-avoid quadcopter using fuzzy control optimized by cross-entropy. In: **Fuzzy Systems (FUZZ-IEEE), 2012 IEEE International Conference on**. [S.l.: s.n.], 2012. p. 1–7. ISSN 1098-7584. Citado na página 28.

ORSAG, M.; BOGDAN, S. Influence of Forward and Descent Flight on Quadrotor Dynamics. **Recent Advances in Aircraft Technology**, p. 141–156, 2012. Citado 2 vezes nas páginas 1 e 3.

RABHI, A.; CHADLI, M.; PEGARD, C. Robust fuzzy control for stabilization of a quadrotor. In: **Advanced Robotics (ICAR), 2011 15th International Conference on**. [S.l.: s.n.], 2011. p. 471–475. Citado na página 33.

RAZINKOVA, A.; GAPONOV, I.; CHO, H.-C. Adaptive control over quadcopter uav under disturbances. In: **Control, Automation and Systems (ICCAS), 2014 14th International Conference on**. [S.l.: s.n.], 2014. p. 386–390. ISSN 2093-7121. Citado na página 29.

REZAZADEH, S.; ARDESTANI, M.; SADEGHI, P. Optimal attitude control of a quadrotor uav using adaptive neuro-fuzzy inference system (anfis). In: **Control, Instrumentation, and Automation (ICCIA), 2013 3rd International Conference on**. [S.l.: s.n.], 2013. p. 219–223. Citado 2 vezes nas páginas 1 e 30.

SENKUL, F.; ALTUG, E. Modeling and control of a novel tilt - roll rotor quadrotor uav. In: **Unmanned Aircraft Systems (ICUAS), 2013 International Conference on**. [S.l.: s.n.], 2013. p. 1071–1076. Citado na página 1.

SHEIKHPOUR, S.; SHOURAKI, S. B. A model-based fuzzy controller using the parallel distributed compensation method for quadrotor attitude stabilization. In: **Electrical Engineering (ICEE), 2013 21st Iranian Conference on**. [S.l.: s.n.], 2013. p. 1–6. Citado na página 33.

YACEF, F.; BOUHALI, O.; HAMERLAIN, M.; REZOUG, A. Pso optimization of integral backstepping controller for quadrotor attitude stabilization. In: **Systems and Control (ICSC), 2013 3rd International Conference on**. [S.l.: s.n.], 2013. p. 462–466. Citado na página 37.

Apêndices

APÊNDICE A – Código para Criação de Modelo Neuro-Fuzzy para Altitude e Definição de Dados para Treinamento

```
% le arquivo fis referente ao controle de altitude
fismat = readfis('fis_altitude.fis');

% define numero de casos a serem avaliados (treinamento + teste)
n = 300;
% define conjunto de n entradas aleatorias para o sistema fuzzy
% respeitando o range de cada entrada
input = zeros(n,2);
for i=1:n
    z_value = rand * 2 - 1;
    z_dot_value = rand * 10 - 5;
    input(i,:) = [ z_value z_dot_value ];
end

% avalia resposta fuzzy para cada entrada
output= evalfis(input,fismat);

% define data como vetor relacionando cada conjunto de entradas ...
   a saida
% - obtida pelo sistema fuzzy
data = [];
for i=1:n
   data(i,:) = [ input(i,:) output(i) ];
end

% define que 2/3 dos dados obtidos serao usasdos para treinamento
% e 1/3 sera usado para teste da rede
train = data(1:2*n/3,:);     % dados para treinamento
test = data(2*n/3+1:n,:);    % dados para validacao do sistema ...
   treinado

% gera modelo fuzzy Sugeno a partir do Mamdani modelado
sugFIS = mam2sug(fismat);
% salva modelo Sugeno em disco com o nome fis_altitude_neuro.fis
writefis(sugFIS, 'fis_altitude_neuro.fis');
```

APÊNDICE B – Código para Criação de Modelo Neuro-Fuzzy para Atitude e Definição de Dados para Treinamento

```
% le arquivo fis referente ao controle de atitude
fismat = readfis('fis_atitude.fis');

% define numero de casos a serem avaliados (treinamento + teste)
n = 300;
% define conjunto de n entradas aleatorias para o sistema fuzzy
% respeitando o range de cada entrada
input = zeros(n,2);
for i=1:n
    phi_value = rand * 4 - 2;
    phi_dot_value = rand * 3 - 1.5;
    input(i,:) = [ phi_value phi_dot_value ];
end

% avalia resposta fuzzy para cada entrada
output= evalfis(input,fismat);

% define data como vetor relacionando cada conjunto de entradas ...
   a saÃda
% obtida pelo sistema fuzzy
data = [];
for i=1:n
   data(i,:) = [ input(i,:) output(i) ];
end

% define que 2/3 dos dados obtidos serao usasdos para treinamento
% e 1/3 sera usado para teste da rede
train = data(1:2*n/3,:);     % dados para treinamento
test = data(2*n/3+1:n,:);    % dados para validaÃ§Ã£o do sistema ...
   treinado

% gera modelo fuzzy Sugeno a partir do Mamdani modelado
sugFIS = mam2sug(fismat);
% salva modelo Sugeno em disco com o nome fis_atitude_neuro.fis
writefis(sugFIS, 'fis_atitude_neuro.fis');
```