1 Introdução

Quadricópteros 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 quadricóptero De Bothezat conseguiu fazer um voo com duração de dois minutos e quarenta e cinco segundos (ORSAG; BOGDAN, 2012).

Os quadricópteros são divididos em duas categorias principais: os do tipo *Indoor* são aqueles projetados para serem utilizados em ambientes controlados[1], ao passo que os *Outdoor* são aqueles projetados de forma a estarem aptos à utilização mesmo em ambientes externos e, portanto, sujeitos a fatores naturais não controlados. Quadricóptero de ambas as categorias vêm sendo utilizados para diversas finalidades (REZAZADEH et al., 2013).

Na última década, quadricópteros 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 quadricóptero se enquadram na categoria das aeronaves VTOL (*Vertical Take-Off and Landing*[2]), possuindo portanto propulsão vertical e tanto decolagem quanto aterrissagem são 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. Esses quadricópteros não tripulados se enquadram na classe dos *UAVs (Unmanned Aerial Vehicles)*[3]. Apesar da enorme variedade de aplicações, o uso deles também envolve diferentes desafios.

Apesar da grande gama de possibilidades que os quadricópteros oferecem, o controle necessário para mantê-los estáticos ou em movimento no ar não é trivial, principalmente se tratando dos modelos *Outdoor*. A complexidade do controle a ser

[1] e.g. sem a presença de vento.
[2] Decolagem e Aterrissagem Verticais (tradução nossa).
[3] Veículos Aéreos não Tripulados (tradução nossa).

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 quadricópteros podem realizar, que são em três dimensões. Com isto, têm-se seis variáveis de configuração do sistema: três delas são referentes à posição do quadricóptero em cada dimensão: x, y e z e as outras três representam o ângulo do quadricóptero em relação a cada um dos eixos: ao eixo x é chamado de ângulo de *roll* (ou de rolamento); ao eixo y, de ângulo de *pitch* (ou de arfagem); e ao eixo z, de ângulo de *yaw* (ou de guinada).

A proposta deste trabalho é implementar diferentes controladores baseados em duas técnicas da Inteligência Computacional (IC), fuzzy e neuro-fuzzy, para permitir a estabilidade em altitude e atitude de um quadricóptero. Um controlador neuro-fuzzy vai além de um sistema baseado na lógica fuzzy puramente, tendo em vista que o primeiro alia a capacidade de aprendizado de uma RNA à teoria de conjuntos fuzzy.

1.1 Relevância

Muitos dos quadricópteros 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 quadricópteros 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 quadricópteros pode ser aplicado para reconhecimento aéreo de áreas de difícil ou perigoso acesso.

As aplicações dos quadricópteros vão além das já alcançadas hoje pelos de pequeno porte. Segundo Orsag e Bogdan (2012), alguns modelos, 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, investigar diferentes abordagens de Inteligência Computacional para controlar de forma eficiente a estabilidade de atitude e altitude de um quadricóptero.

1.2.1 Objetivos Específicos

Os objetivos específicos deste trabalho são:

- Modelar controladores fuzzy e neuro-fuzzy para a estabilização em altitude de um quadricóptero;
- Modelar controladores fuzzy e neuro-fuzzy para a estabilização em atitude de um quadricóptero;
- Comparar os resultados obtidos pelos diferentes controladores com basem em diferentes métricas quando o sistema é submetido a distúrbios:
  - Variação apresentada;
  - Tempo necessário para a estabilização;
  - Oscilação;
  - Sobrelevação apresentada;
  - Gasto enérgico apresentado pelos controladores.

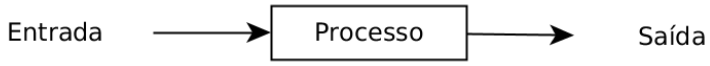
2 Fundamentação Teórica

Neste capítulo, são abordados os temas necessários para a compreensão do trabalho desenvolvido. Na Seção 2.1 é apresentada 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.

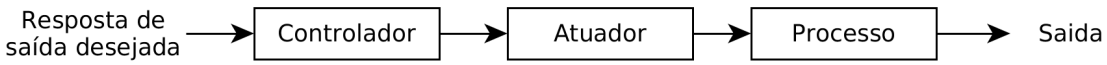
Figura 1 – Diagrama representando um processo



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

O controle de 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

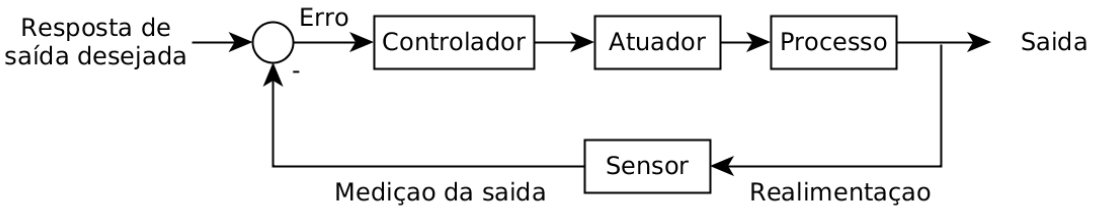


Fonte: Adaptado de Dorf (2011, p. 2)

Em oposição a um sistema de controle em malha aberta, um em malha fechada incorpora, além dos componentes que aquele inclui, uma medição dos estados atuais para serem comparados com os valores desejados para o processo. Um exemplo de um sistema de controle em malha fechada simples é mostrado na Figura 3. Os sistemas em malha fechada apresentam várias vantagens sobre os em malha aberta como, por exemplo, a capacidade de rejeitar distúrbios externos e melhorar a atenuação de ruídos

nas medições, elementos 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. Uma forma de representar sistemas contínuos e é utilizando o espaço de estados.

2.1.1 Espaço de Estados

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

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

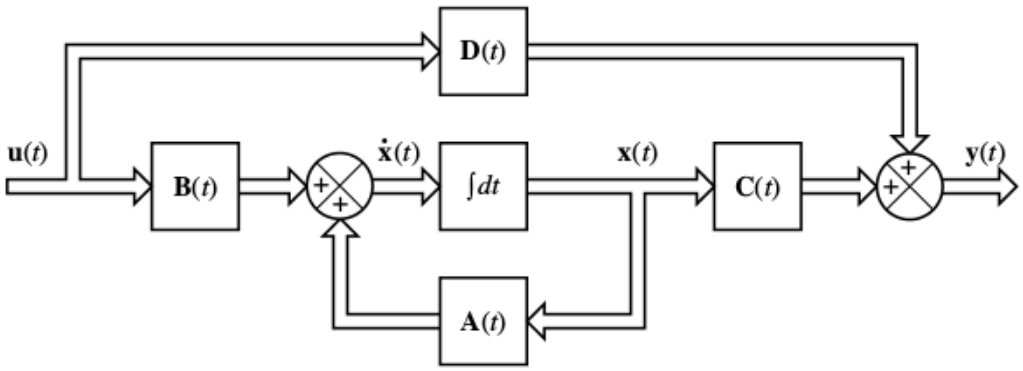
em que A é a matriz de estados, B a matriz de entrada, C a matriz de saída, D a matriz de transmissão direta, x é o vetor de variáveis de estado, u o vetor de entradas e y o vetor de saídas. A Figura 4 exibe um diagrama de blocos para a representação em espaço de estados definida.

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,

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

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

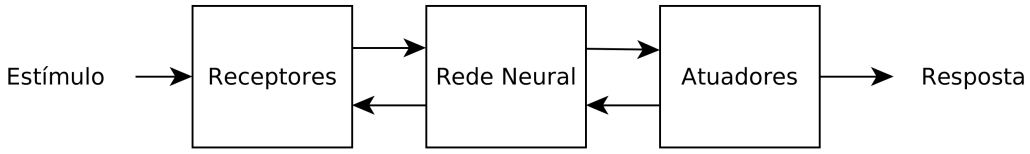


Fonte: Ogata (2010, p. 32)

uma RNA é composta por neurônios artificiais e pelas interações existentes entre estes neurônios: as sinapses.

A Figura 5 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 5 – Representação em digrama de blocos do sistema nervoso



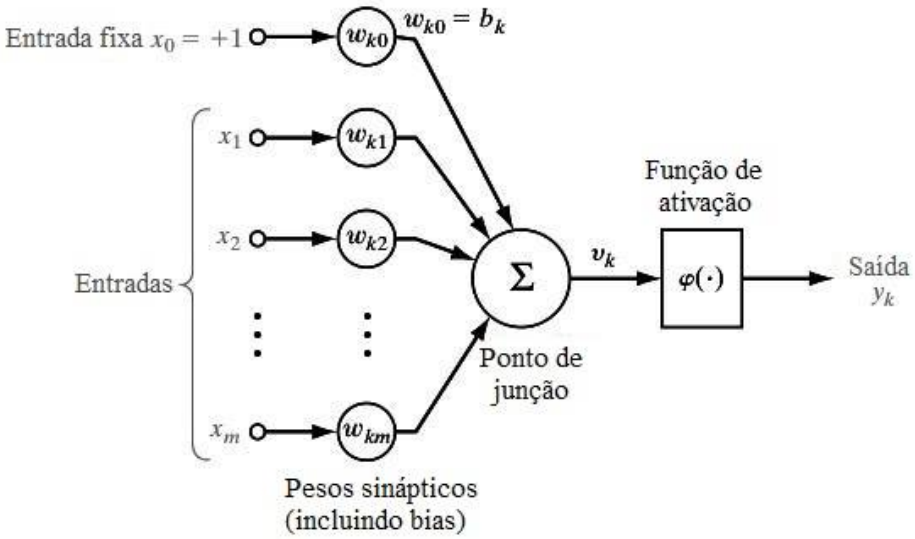
Fonte: Adaptado de Haykin (1998, p. 28)

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 por meio 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. Um modelo neural comumente aplicado na literatura em problemas relacionados às RNAs é o perceptron.

Um perceptron é um modelo computacional de um neurônio não linear e é ilustrado na Figura 6, 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 representada 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 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 6 – Modelo não linear de um neurônio

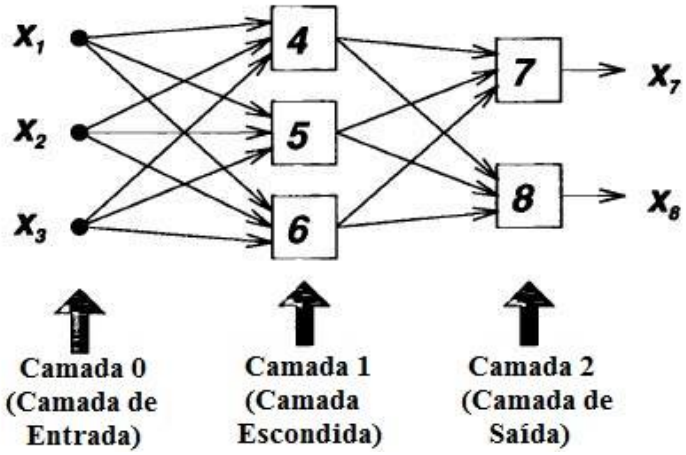


Fonte: Adaptado de Haykin (1998, p. 33)

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 7. 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 possam ser utilizadas em aplicações diversas relacionadas à IA e IC.

[2] Perceptron Multicamada (tradução nossa).

Figura 7 – Modelo de um *MLP*



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

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 a RNAs se beneficiam justamente destas capacidades 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 pertinência 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

[3] lógica aristotélica.

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.

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:

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

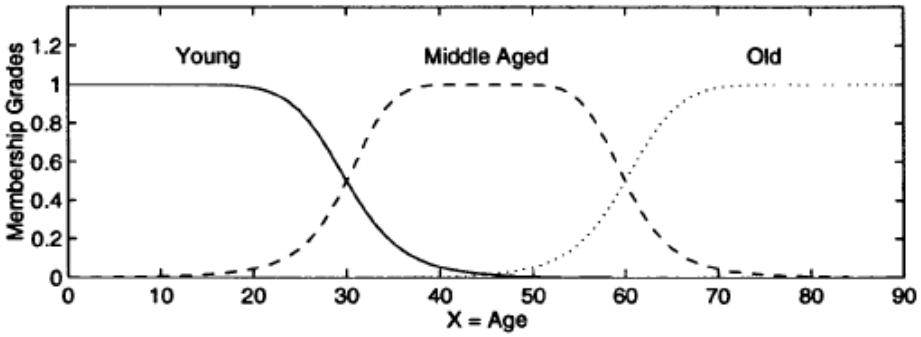
Nesta definição, um conjunto *fuzzy* A é composto pelos pares $(x,\mu_A(x))$ de cada elemento x pertencente a um conjunto X, em que x é o elemento em si e $\mu_A(x)$ é o grau de pertinência de x ao conjunto A, que pode assumir qualquer valor entre 0 e 1, em que 0 representa a não-pertinência total do 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 (JANG et al., 1997, p. 14).

Para melhor exemplificar a definição de conjuntos *fuzzy*, tome como exemplo a Figura 8. Neste caso, a variável idade (*age*) é dividida em três subconjuntos *fuzzy*, *young*, *middle aged* e *old*[4], 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.

[4] jovem, meia idade e velho, respectivamente (tradução nossa).

Figura 8 – Funções de pertinência representando três conjuntos *fuzzy* para a variável idade



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

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

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.

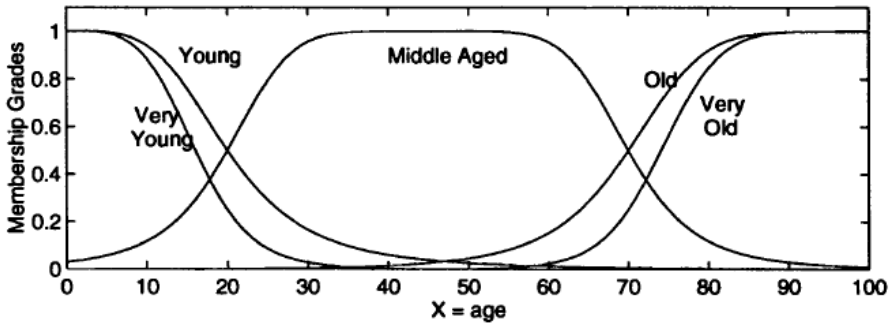
Para facilitar a compreensão das definições relacionadas às variáveis linguísticas, o seguinte exemplo foi dado (JANG et al., 1997, p. 55): se *idade* é interpretado como uma

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 9. 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 9 mostra algumas das funções de pertinência típicas para a variável linguística idade.

Figura 9 – 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[5]) (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

[5] Do inglês, *Generalized Modus Ponens.*

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

Os dois principais tipos de FIS são os Mamdani 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*:

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

Já no sistema de inferência *fuzzy* Sugeno, também conhecido como TSK (Takagi-Sugeno-Kang), o consequente é uma função envolvendo as variáveis de entrada. Com isto, uma regra *fuzzy* típica de um modelo *fuzzy* Sugeno apresenta 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 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 resulta em 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, no sentido matemático, 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.

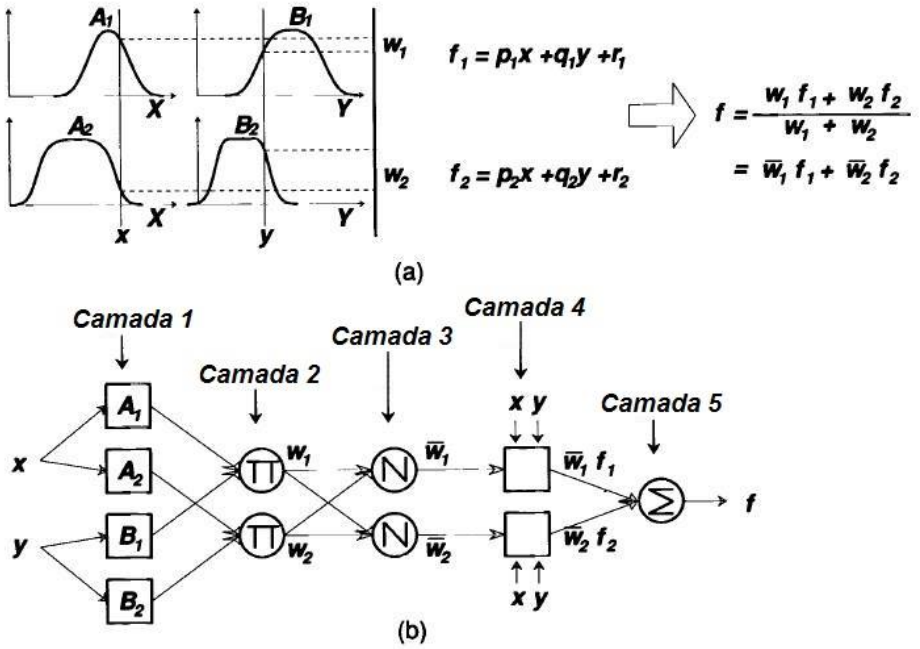
Para simplificar, a arquitetura ANFIS[6] será descrita considerando-a dotada de duas entradas x e y e uma saída z. Para um modelo *fuzzy* Sugeno de primeira ordem, um conjunto comum de regras do tipo *se-então* é o seguinte:

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

A Figura 10 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.

[6] Sistema de Inferência Neuro-*Fuzzy* Adaptativo (tradução nossa).

Figura 10 – 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, todo nó i é um nó adaptativo com uma função de nó do tipo:

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

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, 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[7] que desempenhe a operação *AND fuzzy* pode ser usado como função de nó nesta camada (JANG et al., 1997, p. 337).

[7] Operador de interseção (JANG et al., 1997, p. 337)

Na Camada 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, 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 é um nó fixo denominado $\sum$, e é responsável por computar a saída global como a soma ponderada 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)[8] é um sistema de inferência fuzzy projetado especificamente para atuar como controlador de um sistema 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 proveem uma alternativa viável, uma vez que eles podem capturar os aspectos 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

[8] Do inglês, *Fuzzy Logic Controller*.

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.

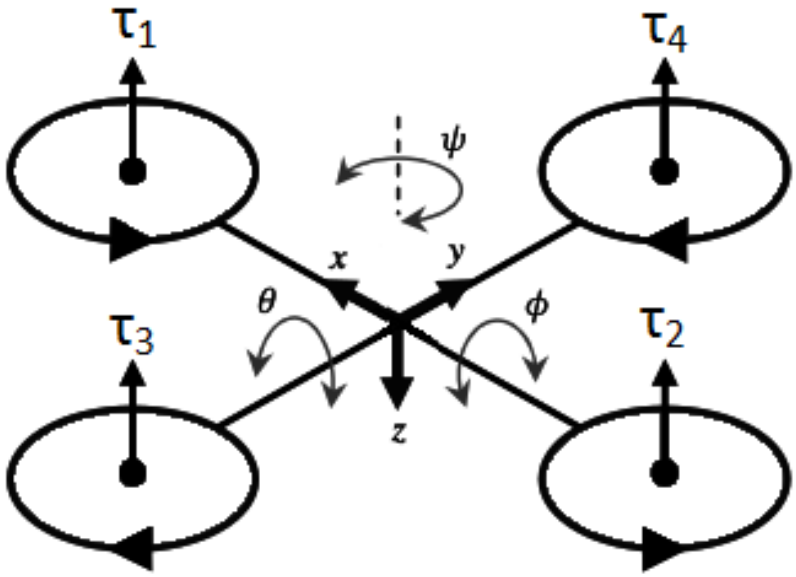
Como comparativo, é importante salientar que uma RNA possui as propriedades 1 e 2, mas não as 3 e 4, que são devidas à contribuição dos sistemas de inferência fuzzy (JANG et al., 1997, p. 458). Um ANFIS, por aliar as duas técnicas, possui todas essas propriedades.

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 Modelagem de um Quadricóptero

Como definido no Capítulo 1, um quadricóptero é uma aeronave cuja propulsão é obtida a partir do uso de quatro rotores e sua representação é mostrada na Figura 11.

Figura 11 – Representação de um quadricóptero



Fonte: Adaptado de Basri et al. (2014)

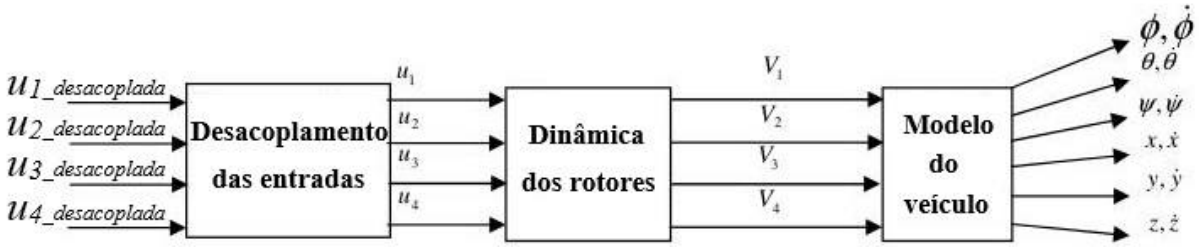
Neste diagrama, x, y e z indicam a orientação dos eixos, τ_1, τ_2, τ_3 e τ_4 são as forças geradas pela propulsão de cada um dos motores. As variáveis ϕ, θ, e ψ são os ângulos de rotação do quadricóptero em relação aos eixos x, y e z respectivamente. Estes ângulos são também chamados ângulos de *roll* (rolamento), *pitch* (arfagem) 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 2 (dispostos sobre o eixo x), giram no sentido anti-horário. Já os rotores 3 e 4 (dispostos sobre o eixo y) giram no sentido 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 quadricóptero, entretanto, assume mais de uma forma, sendo dividida em dois aspectos principais: altitude e atitude. O primeiro diz respeito ao posicionamento do quadricóptero sobre o eixo z e seu controle é necessário tanto para possibilitar 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 quadricóptero, 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 na Figura 12

Figura 12 – Diagrama do sistema representando um quadricóptero modelado por Balas (2007)



Fonte: Adaptado de Balas (2007, p. 49)

3.1 Modelagem Matemática

O quadricóptero 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 quadricóptero 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):

$$u_1 = \tau_1 + \tau_2 + \tau_3 + \tau_4 \tag{1}$$

$$u_2 = l(\tau_3 - \tau_4) \tag{2}$$

$$u_3 = l(\tau_1 - \tau_2) \tag{3}$$

$$u_4 = v_1 + v_2 + v_3 + v_4 \tag{4}$$

em que u_1 é o impulso total, u_2 é o momento de rolamento, u_3 é o momento de arfagem e u_4 é o momento de guinada.

Além disso, aplicando as leis de Newton, a aceleração em cada eixo pode ser representada por (BALAS, 2007, p. 5):

$$\ddot{x} = -\frac{sen(\theta)cos(\phi)}{m}u_1 \tag{5}$$

$$\ddot{y} = \frac{sen(\phi)}{m}u_1 \tag{6}$$

$$\ddot{z} = -\frac{cos(\theta)cos(\phi)}{m}u_1 + g \tag{7}$$

em que ϕ e θ são os ângulos em torno dos eixos x e y respectivamente, m é a massa do quadricóptero 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 quadricóptero seja rígida, sendo I_{xx}, I_{yy} e I_{zz} o momento de inércia do quadricóptero 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):

$$\ddot{\phi} = -\dot{\psi}\dot{\theta}cos(\phi) + \frac{cos(\psi)}{I_{xx}}u_2 - \frac{sen(\psi)}{I_{yy}}u_3 + \frac{I_{yy} - I_{zz}}{I_{xx}}(\dot{\psi} - \dot{\theta}sen(\phi))\dot{\theta}cos(\phi) \tag{8}$$

$$\begin{aligned}\ddot{\theta} = \frac{\dot{\psi}\dot{\phi}}{cos(\phi)} + \dot{\phi}\dot{\theta}tan(\phi) + \frac{sen(\psi)}{cos(\phi)I_{xx}}u_2 + \frac{cos(\psi)}{cos(\phi)I_{yy}}u_3 \\ - \frac{I_{yy} - I_{zz}}{I_{xx}}(\psi - \dot{\theta}sen(\phi))\frac{\dot{\phi}}{cos(\phi)}\end{aligned} \tag{9}$$

$$\begin{aligned}\ddot{\psi} = \dot{\phi}\dot{\psi}tan(\phi) + \frac{\dot{\phi}\dot{\theta}}{cos(\phi)} + \frac{sen(\psi)tan(\phi)}{I_{xx}}u_2 + \frac{cos(\phi)tan(\psi)}{I_{yy}}u_3 + \frac{1}{I_{zz}}u_4 \\ - \frac{I_{yy}I_{zz}}{I_{xx}}(\dot{\psi} - \dot{\theta}sen(\phi))\dot{\phi}tan(\phi)\end{aligned} \tag{10}$$

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):

$$v_i = \tau_i(V_c + v_i) + 0.125\rho bcR_p^4\omega_{m\_i}^2C_d \tag{11}$$

sendo v_i o torque sobre o rotor, τ_i o impulso atuando sobre ele, V_c a velocidade vertical[1], 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 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).

[1] A velocidade vertical do quadricóptero também é indicada por $\dot{z}$

Neste caso, foram considerados os acoplamentos entre ϕ, θ, ψ e u_2, u_3, u_4, relacionados da seguinte maneira:

$$u_{2_{\_decoupled}} = cos(\psi_0)u_2 - \frac{I_{xx}sen(\psi_0)}{I_{yy}}u_3 = I_{xx}\ddot{\phi} \tag{12}$$

$$u_{3_{\_decoupled}} = \frac{sen(\psi_0)I_{yy}}{cos(\phi_0)I_{xx}}u_2 + \frac{cos(\psi_0)}{cos(\phi_0)}u_3 = I_{yy}\ddot{\theta} \tag{13}$$

$$u_{4_{\_decoupled}} = \frac{I_{zz}sen(\psi_0)tan(\phi_0)}{I_{xx}}u_2 + \frac{I_{zz}cos(\psi_0)tan(\phi_0)}{I_{zz}}u_3 + u_4 = I_{zz}\ddot{\psi} \tag{14}$$

em que $\dot{\phi}_0$, $\dot{\theta}_0$ e $\dot{\psi}_0$ são os ângulos no iniciais de rolamento, arfagem e guinada respectivamente; $\dot{\phi}$, $\dot{\theta}$ e $\dot{\psi}$ são os ângulos atuais de rolamento, arfagem e guinada; e I_{xx}, I_{yy} e I_{zz} são o momento de inércia em torno dos eixos x, y e z respectivamente.

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 estados, prática comum para descrever sistemas dinâmicos.

3.2.1 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.

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned} \tag{15}$$

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 matrizes A, B, C e D sendo definidas como:

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

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -1/m & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & cos(\psi_0)/I_{xx} & -sen(\psi_0)/I_{yy} & 0 \\ 0 & sen(\psi_0)/(cos(\phi_0)I_{xx}) & cos(\psi_0)/(cos(\phi_0)I_{yy}) & 0 \\ 0 & sin(\psi_0)tan(\phi_0)/I_{xx} & cos(\psi_0)tan(\phi_0)/I_{yy} & 1/I_{zz} \end{bmatrix}$$

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

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

em que g é a gravidade, m, a massa do quadricóptero e os ângulos ϕ_0, θ_0 e ψ_0 representam os ângulos iniciais 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.

Como se pode perceber, a matriz C é uma matriz diagonal, o que implica no fato de a saída do sistema ser representada pelo vetor de estados.

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

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 quadricópteros, 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 quadricópteros 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 Quadricópteros

Esta seção aborda diferentes propostas de controladores tradicionais para quadricópteros visando a contextualização do estado da arte para depois se poder comparar alguns resultados das técnicas tradicionais e inteligentes de controle para esse sistema.

[1] Ver e evitar (tradução nossa)

Em (RAZINKOVA et al., 2014), foi proposto um controlador PD para a descrição adequada das trajetórias definidas de um quadricóptero *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 quadricóptero 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 visando a estabilidade da altitude de um quadricóptero. 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 foi realizada no Matlab® 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 quadricóptero.

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

4.2 Controle Inteligente de Quadricópteros

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 sistemas neuro-*fuzzy* expandido para o espaço dos números complexos.

Em (REZAZADEH et al., 2013), é proposto um controlador neuro-*fuzzy* (ANFIS) para o controle estabilizante de altitude de um quadricóptero. O módulo *fuzzy* acoplado a um controlador PID é usado para controlar a atitude. Os controladores *fuzzy* atualizam os ganhos do PID *online* para produzir uma resposta apropriada. 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. O controlador proposto foi comparado a um simples controlador PID e, a partir de simulações, foi mostrado que se obteve melhor resposta, por exemplo, tendo em vista 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.

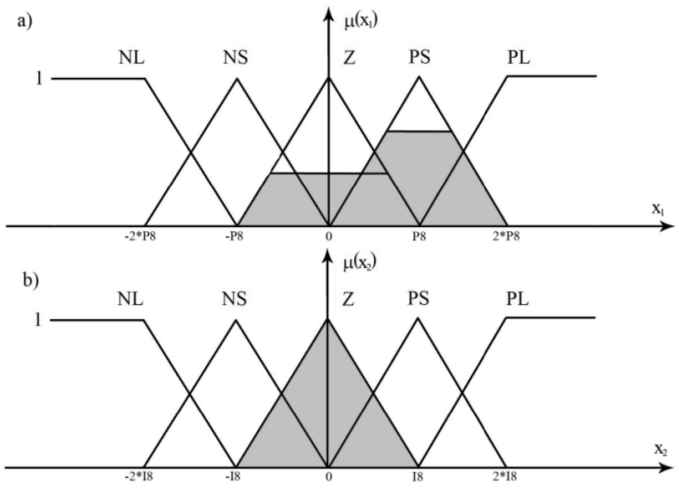
Em (MAHFOUZ et al., 2013), foi proposto um controlador parecido. Neste trabalho, também foi sugerido um controlador ANFIS que, juntamente com um algoritmo genético, ajusta os pesos de um controlador PID. Mais uma vez, foi obtido sucesso com o controlador desenvolvido, sendo possível garantir a estabilidade de um *drone* mesmo quando sujeito a perturbações. No trabalho, foi mostrado que a sobrelevação do controlador ANFIS é menor do que a do PID. Além disto, o erro de regime permanente no caso do ANFIS é zero, enquanto, 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 (x e y) 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 13.

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 utilizadas é mostrado no Quadro 1.

Os autores obtiveram sucesso no controle estabilizante do *drone*, mas somente utilizando um bloco integrador. Apesar de os reguladores e motores serem do mesmo

Figura 13 – Conjuntos *fuzzy* usados em (MAJ; BUTKIEWICZ, 2013); a) erro de posição; b) erro de giro



Fonte: (MAJ; BUTKIEWICZ, 2013)

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)

tipo, algumas características deles são levemente diferentes. Como consequência, os motores não giram todos na mesma velocidade causando uma inclinação do quadricóptero. Esta inclinação pode ser corrigida com o uso de um bloco integrador. Desta forma, foi 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 do sistema. Um SMC (*Sliding Mode Control*[2]) pode ser apropriado mas pode causar chaveamento indesejado no sinal de controle. Sistemas de controle com RNAs podem ser apropriados, mas requerem intenso esforço computacional para

[2] Controle por Modos Deslizantes (tradução nossa).

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 vento, o controlador teve de sacrificar a bom desempenho 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 de pertinência Gaussianas 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 quadricóptero. O controlador foi modelado usando ferramentas matemáticas, mais especificamente LMIs (*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 quadricóptero. 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 desempenho 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 velocidade de resposta desejada e restrições na entrada de altitude 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 quadricóptero e então foi desenvolvido um controlador híbrido usando métodos de controle tradicionais e inteligentes para estabilização

[3] Desigualdades Matriciais Lineares (tradução nossa).
[4] Compensação Paralela Distribuída (tradução nossa).

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. 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 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 quadricóptero 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 quadricópteros com melhor desempenho 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 Adaptativo, que alia a lógica *fuzzy* a um controlador PID. Quando o quadricóptero voa em boas condições, o controle de estabilidade é assumido pelo controlador *Backstepping*. Quando o quadricóptero encontra vento ou outro fator de perturbação durante o voo, o controlador *Fuzzy* PID Adaptativo[6] é usado para estabilizá-lo. . O controlador proposto alcançou sucesso nos testes realizados, permitindo que se complete a trajetória sem que perturbações afetem a precisão do controle. Com isso, o Sistema de Inferência *Fuzzy* se mostrou eficiente para ajustar os ganhos do controlador PID de forma a possibilitar o controle eficiente do quadricóptero.

Em (GAO et al., 2014a), um outro trabalho dos mesmos autores, são comparados os desempenhos de três controladores diferentes: PID, *Fuzzy* PD Adaptativo e *Backstepping*. Mais uma vez, aqui o Sistema de Inferência *Fuzzy* tem como função ajustar os parâmetros do controlador PD: ΔK_p e ΔK_d. Nas simulações feitas foi verificado que, embora o controlador *Backstepping* seja pior que os outros dois, ele pode rapidamente

[5] Controlador por *Backstepping Fuzzy* Supervisionado (tradução nossa).

[6] O controlador *Fuzzy* PID Adaptativo é um controlador PID que emprega o Sistema de Inferência *Fuzzy* (FIS) para ajustar os parâmetros K_p, K_i, K_d, ganhos proporcional, integral e derivativo respectivamente.

suprimir o impacto de perturbações. Foi mostrado ainda que o controlador *Fuzzy* PD Adaptativo obteve o melhor resultado relacionado a rejeição de perturbações. O tempo de subida do controlador *Fuzzy* PD Adaptativo é ligeiramente menor nos eixos x e z mas um pouco maior no eixo y.

Outras abordagens híbridas ainda foram adotadas para o projeto de controladores de estabilidade para *drones*. Em (YACEF et al., 2013), foi proposto o uso da técnica *Particle Swarm Optimization*[7] (PSO) para ajuste de um IBC. O método é usado para minimizar o erro quadrático (SE) de uma função de custo que quantifica o desempenho de todo o sistema. Resultados de numerosas simulações mostram a validação e as bons desempenhos 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 quadricópteros. Como já dito anteriormente, isso se deve às inúmeras vantagens citadas que elas trazem. Algumas dessas técnicas foram utilizadas neste trabalho, como é mostrado a seguir.

[7] Otimização por Enxame de Partículas (tradução nossa).

5 Metodologia

Todo o trabalho foi desenvolvido em ambiente simulado, utilizando o *software* Matlab®. Primeiramente, foram representados no Simulink® dois sistemas de quadricóptero seguindo a modelagem proposta por Balas (2007): ambos submetidos a uma gravidade $g = 9{,}8$ m/s^2 e com comprimento de cada haste $l = 0{,}5$ m. Os dois sistemas diferem entretanto na massa do quadricóptero modelado em cada caso: $m = 2{,}3$ kg e $m = 5$ kg.

O sistema com massa $m = 2{,}3$ kg foi então utilizado para mostrar o desacoplamento das entradas e a instabilidade do sistema. Para tanto, o modelo foi submetido a sinais em pulso em cada uma de suas entradas. Então, a partir da resposta do sistema a essas entradas, foram modelados dois controladores *fuzzy* para estabilizar a atitude e altitude do quadricóptero. Para tanto, foi utilizada a ferramenta *Fuzzy Logic Toolbox* do Matlab®.

A partir dos controladores *fuzzy* desenvolvidos e utilizando a ferramenta *Neuro-Fuzzy Designer* também do Matlab® foram modelados dois controladores neuro-*fuzzy* para controlar a atitude e altitude do *drone*.

Os controladores *fuzzy* e neuro-*fuzzy* foram então comparados tanto para o sistema com massa de $m = 2{,}3$ kg quanto para o de $m = 5$ kg. Os aspectos levados em conta para a comparação dos controladores foram:

- Variação apresentada;
- Tempo necessário para a estabilização;
- Oscilação;
- Sobrelevação apresentada;
- Gasto enérgico apresentado pelos controladores.

Por fim, o sistema com massa m = 2,3 kg, para o qual os controladores foram desenvolvidos, foi submetido a um cenário que envolve ruídos de medição para verificar se o controle implementado se mostra robusto.

6 Desenvolvimento do Trabalho

Este capítulo trata da forma como o trabalho foi desenvolvido de forma a aplicar a metodologia proposta, descrevendo os processos utilizados para a verificação do desacoplamento e instabilidade do sistema (Seção 6.1); a modelagem dos controladores *fuzzy* (Seção 6.2) e neuro-*fuzzy* (Seção 6.3); e a descrição detalhada dos experimentos realizados (Seção 6.4).

6.1 Verificação do Desacoplamento das Entradas

6.2 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 quadricóptero $m = 2{,}3$ kg e o comprimento de cada haste do *drone* $l = 0{,}5$ m. Um dos controladores projetado tem, como objetivo, controlar a altitude do quadricóptero, ao passo que um segundo visa a sua estabilização de sua atitude.

O controlador de altitude possui duas entradas e uma saída. As entradas são referentes à posição vertical do quadricóptero (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 14.

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), tomando como base os trabalhos de Maj e Butkiewicz (2013) e Gao et al. (2014b). As regras fuzzy definidas para este controlador são mostradas no Quadro 2, e a Figura 15 exibe seu equivalente em superfície.

Quadro 2 – 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

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

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

Figura 14 – Diagrama do sistema de controle de altitude utilizando controlador fuzzy

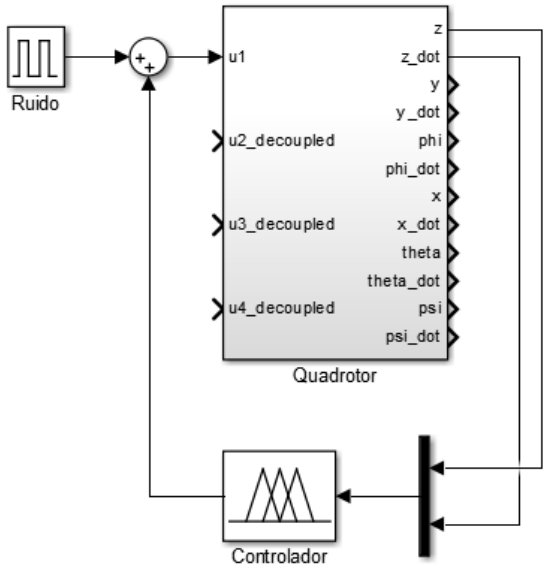
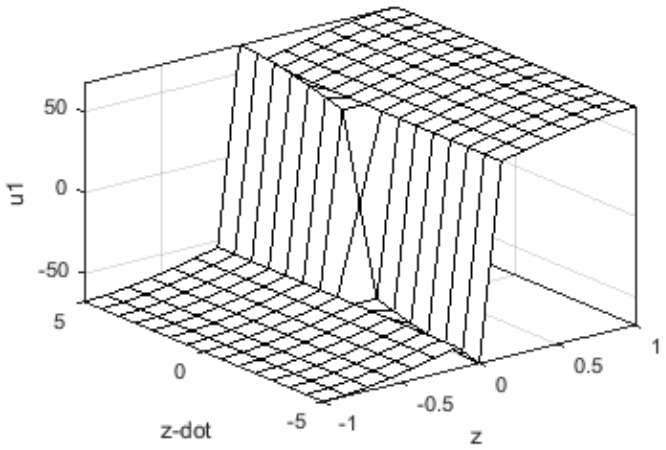


Figura 15 – Superfície das regras do sistema de controle fuzzy para a altitude do quadrotor



(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.

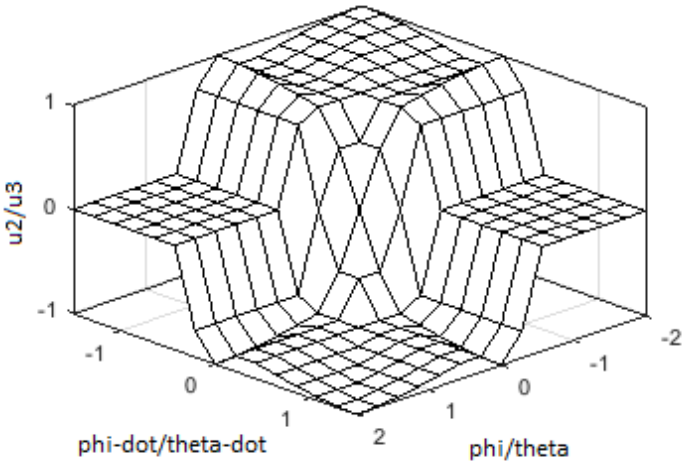
As regras que regem o controlador de atitude são sintetizadas no Quadro 3 e podem ser vistas na superfície de regras mostradas na Figura 16.

Já as Figuras 17 e 18 exibem os diagramas do sistema controlado, com atuação sobre os ângulos ϕ e θ, respectivamente.

Quadro 3 – 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 16 – Superfície das regras do sistema de controle fuzzy para a atitude do quadrotor



6.3 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.

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`.

[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 17 – Diagrama do sistema de controle de atitude utilizando controlador fuzzy sobre o ângulo ϕ

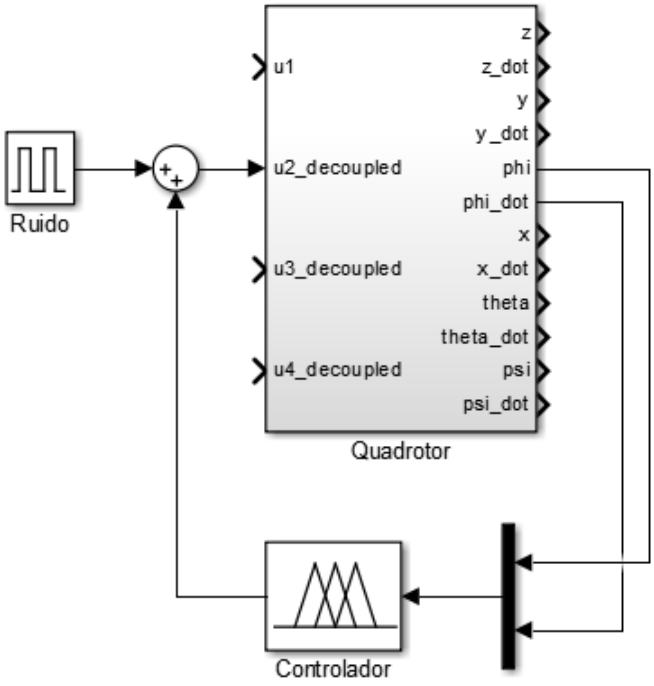
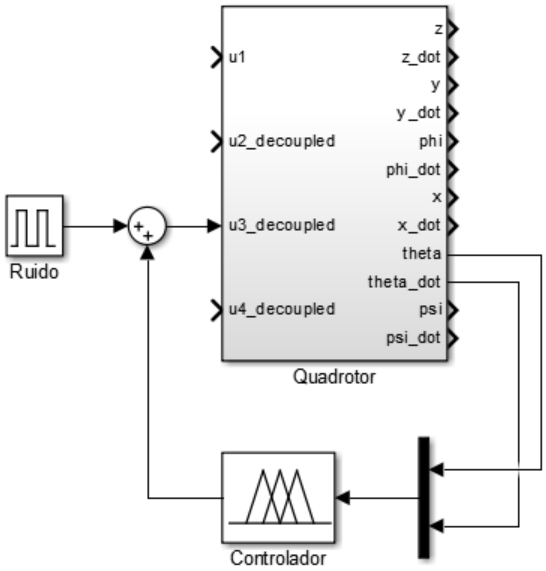


Figura 18 – Diagrama do sistema de controle de atitude utilizando controlador fuzzy sobre o ângulo θ



Feito isto, utilizou-se o comando comando `anfisedit` para abrir o *Neuro-Fuzzy Designer* do MATLAB, cuja interface é mostrada na Figura 19. 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 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 20, 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 19 – Interface gráfica da ferramenta *Neuro-Fuzzy Designer* com destaque aos três campos necessários para treinamento e teste da rede neuro-fuzzy

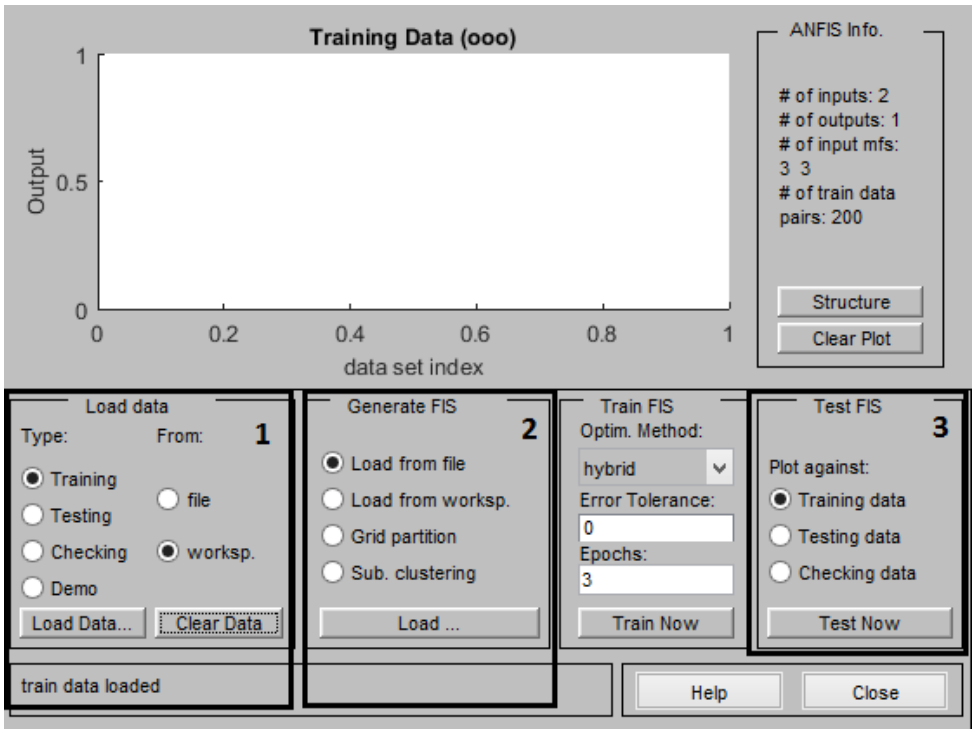
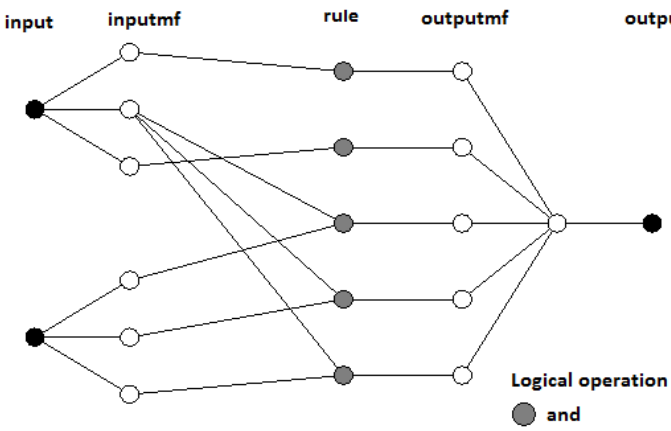


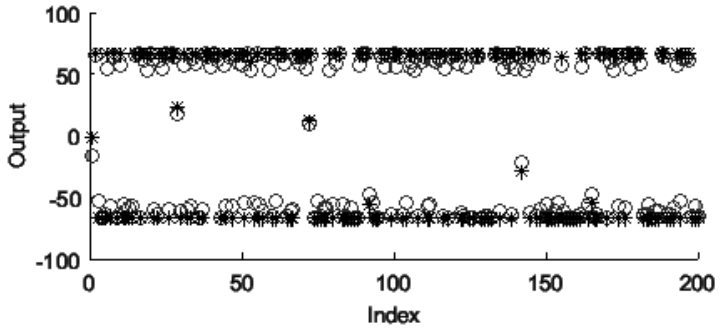
Figura 20 – Diagrama da RNA referente ao controlador neuro-fuzzy para altitude



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.

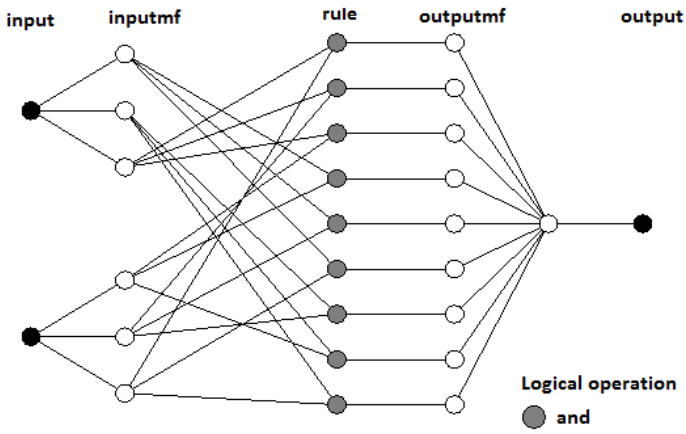
A Figura 21 mostra o gráfico obtido na ferramenta após o processo de treinamento, em que os círculos brancos mostram os dados utilizados no treinamento e os asteriscos pretos indicam o valor referentes a eles obtidos pela rede treinada.

Figura 21 – 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 22 e 23 mostram o diagrama da RNA referente ao controlador neuro-fuzzy para atitude e o resultado obtido pelo seu treinamento respectivamente.

Figura 22 – Diagrama da RNA referente ao controlador neuro-fuzzy para atitude



O processo de treinamento determina o comportamento dos controladores neuro-fuzzy projetados, cujas superfícies de regras são exibidas nas Figuras 24 e 25.

Figura 23 – Resultado obtido pelo treinamento da RNA para controle de atitude

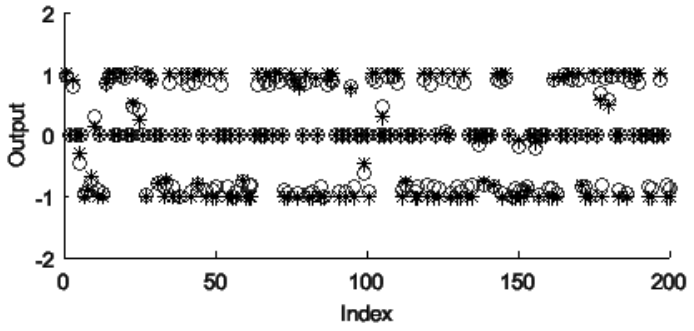


Figura 24 – Superfície das regras do sistema de controle neuro-fuzzy para a altitude do quadrotor

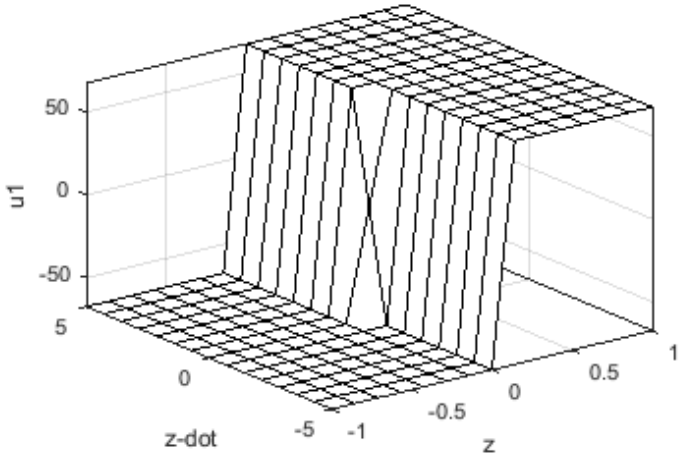
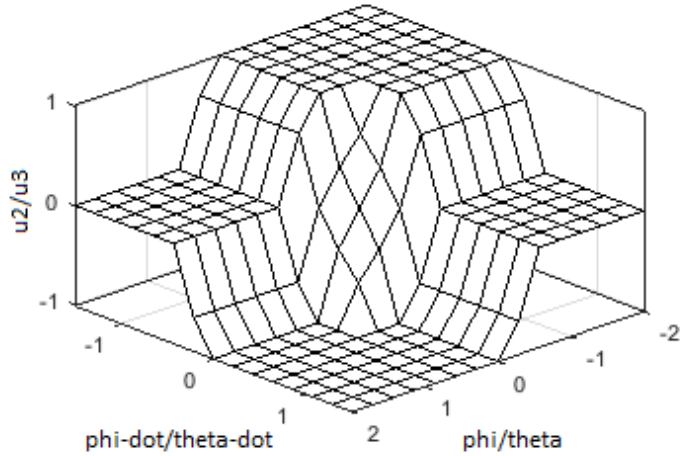


Figura 25 – Superfície das regras do sistema de controle fuzzy para a atitude do quadrotor



6.4 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.

Referências

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 19, 20, 21, 31 e 32.

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 18 e 29.

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 30.

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 2 vezes nas páginas 27 e 28.

DORF, R. **Modern control systems**. 12. ed. Upper Saddle River, N.J: Pearson Prentice Hall, 2011. ISBN 978-0136024583. Citado 2 vezes nas páginas 4 e 5.

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 na página 29.

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 2 vezes nas páginas 29 e 32.

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 24.

HAYKIN, S. **Neural Networks: A Comprehensive Foundation**. NJ, USA: Prentice Hall PTR Upper Saddle River, 1998. ISBN 0132733501. Citado 4 vezes nas páginas 5, 6, 7 e 8.

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 8, 9, 10, 11, 12, 13, 14, 15, 16 e 17.

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 25.

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 26.

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 3 vezes nas páginas 26, 27 e 32.

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 25.

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 28.

OGATA, K. **Modern Control Engineering**. 5. ed. [S.l.]: Prentice Hall, 2010. ISBN 978-0136156734. Citado na página 6.

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 24.

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 2.

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 28.

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 25.

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 26.

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 28.

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 30.

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');
```