

1 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 $g = 9,8 \text{ m/s}^2$ e com comprimento de cada haste $l = 0,5 \text{ m}$. Os dois sistemas diferem entretanto na massa do quadricóptero modelado em cada caso: $m = 2,3 \text{ kg}$ e $m = 5 \text{ kg}$.

O sistema com massa $m = 2,3 \text{ kg}$ 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 $m = 2,3 \text{ kg}$ quanto para o de $m = 5 \text{ kg}$. Os aspectos levados em conta para a comparação dos controladores foram:

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

Por fim, o sistema com massa $m = 2,3 \text{ kg}$, 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.

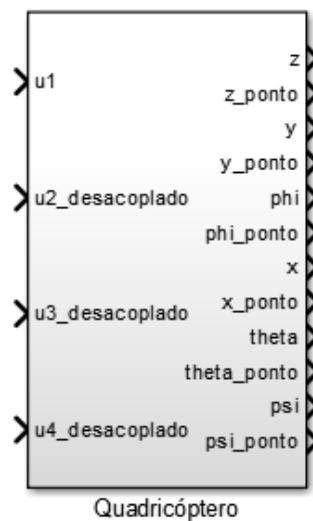
2 Desenvolvimento do Trabalho

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 2.1); a modelagem dos controladores *fuzzy* (Seção 2.2) e *neuro-fuzzy* (Seção 2.3); e a descrição detalhada dos experimentos realizados (Seção 2.4).

2.1 Verificação do Desacoplamento das Entradas

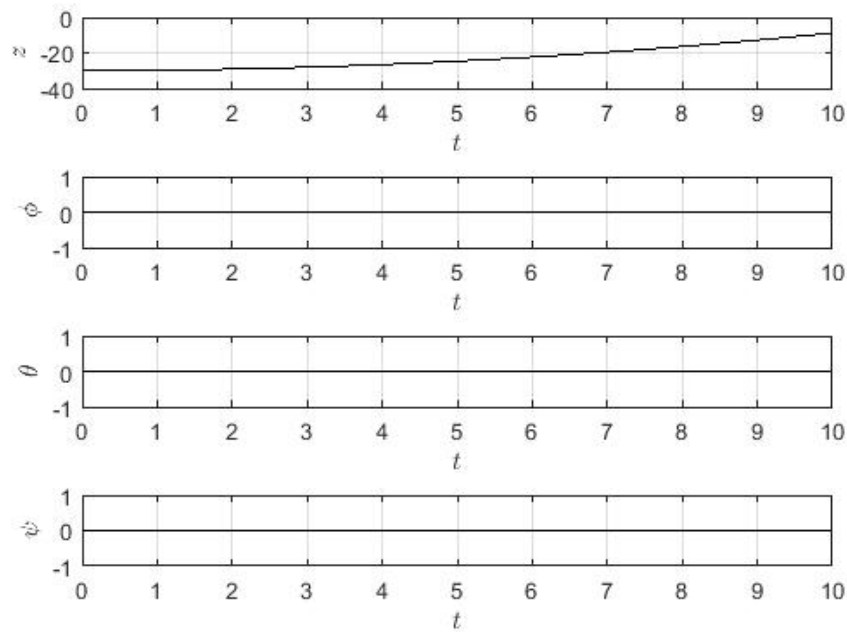
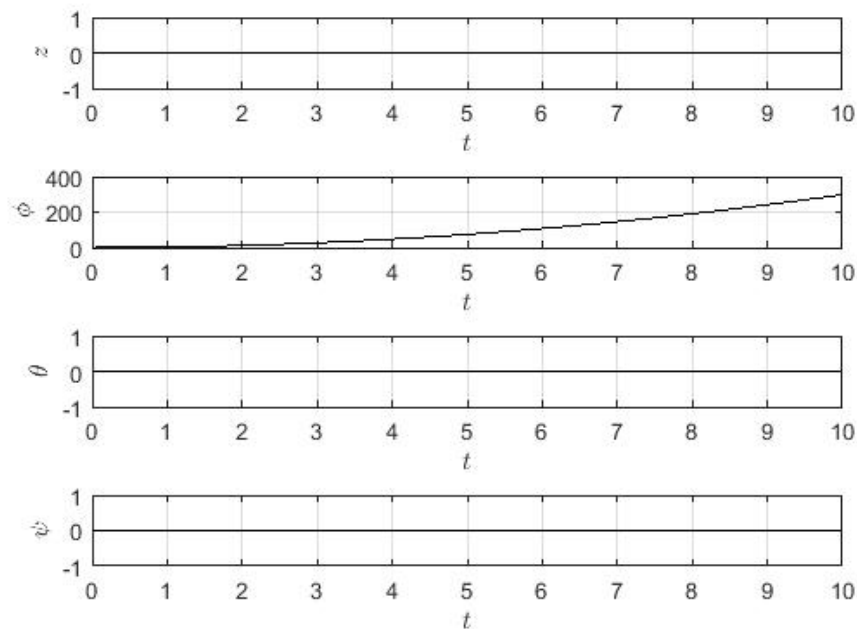
A representação do quadricóptero criada no Simulink® seguindo a modelagem de Balas (2007) é mostrada na Figura 1.

Figura 1 – Representação do quadricóptero no *Simulink*

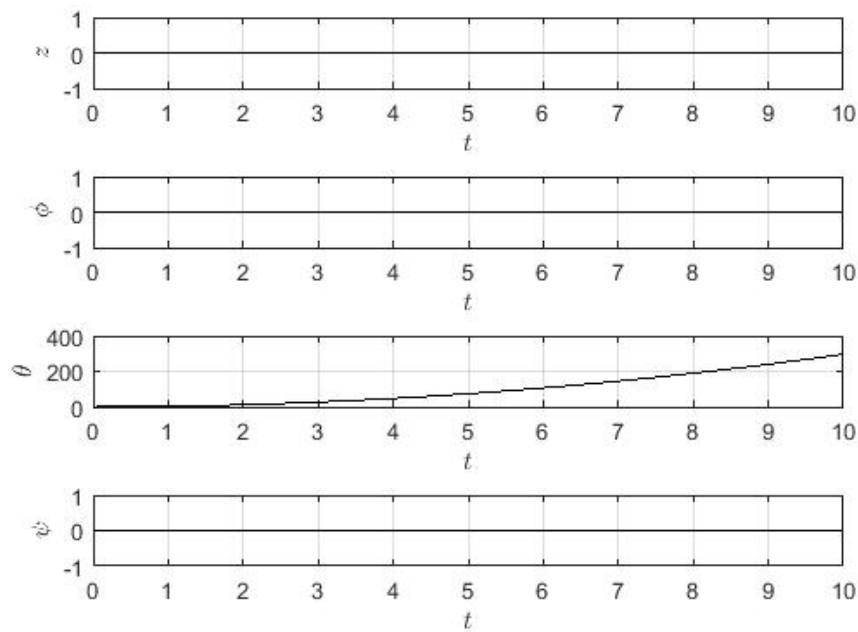
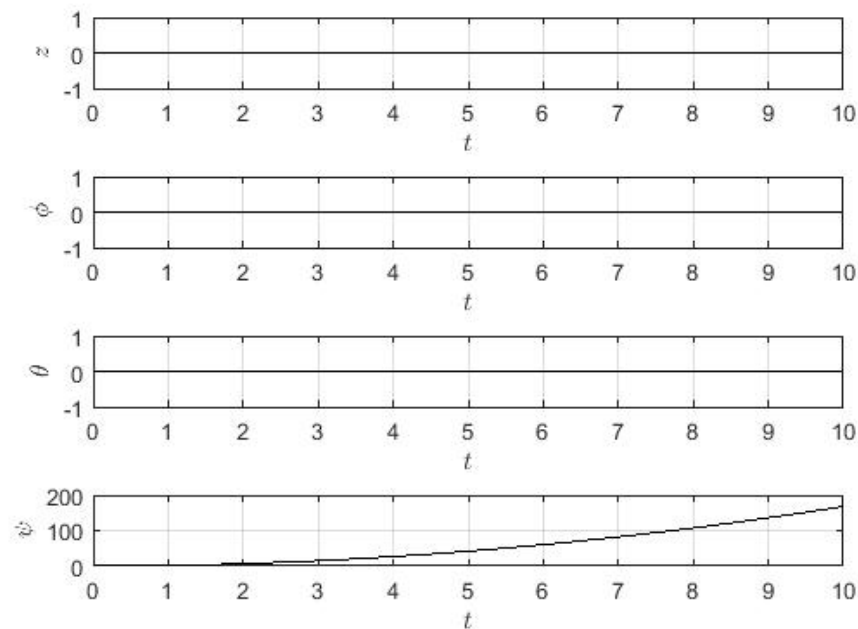


Como se pode ver, o sistema inclui os quatro sinais de entrada seguindo o desacoplamento desenvolvido ($u1$, $u2_desacoplado$, $u3_desacoplado$ e $u4_desacoplado$) e com as doze saídas referentes às seis variáveis de configuração $x, y, z, \phi, \theta, \psi$ indicadas por $x, y, z, \phi, \theta, \psi$, respectivamente; e suas respectivas variações $\dot{x}, \dot{y}, \dot{z}, \dot{\phi}, \dot{\theta}, \dot{\psi}$ representadas por $x_ponto, y_ponto, z_ponto, \phi_ponto, \theta_ponto$ e ψ_ponto .

Para mostrar o desacoplamento das variáveis, alternadamente foi aplicado um sinal de degrau a cada uma das entradas. Em cada um dos casos, somente uma entrada era submetida ao degrau, ao passo que as demais eram aterradas. As respostas, a cada um dos experimentos, das variáveis de configuração relativas à altitude e atitude do quadricóptero são mostradas nas Figuras 2, 3, 4 e 5 tomando como estado inicial um quadricóptero estável ($\phi = \theta = \psi = 0$ rad) a trinta metros de altura ($z = 30$ m).

Figura 2 – Resposta das saídas z , ϕ , θ e ψ a um entrada em degrau em u_1 Figura 3 – Resposta das saídas z , ϕ , θ e ψ a um entrada em degrau em $u_{2_desacoplado}$ 

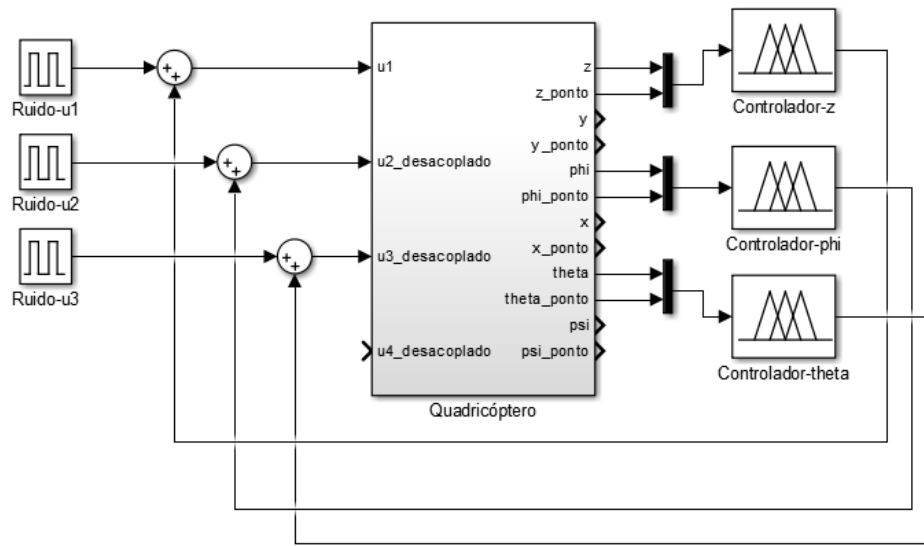
Como se pode ver, cada entrada afeta uma única saída e cada saída é afetada apenas por uma entrada. Com isso, mostra-se o desacoplamento existente que faz com que a entrada u_1 somente interfira na variável de configuração z ; $u_{2_desacoplado}$ em ϕ ; $u_{3_desacoplado}$ em θ ; e $u_{4_desacoplado}$ em ψ .

Figura 4 – Resposta das saídas z , ϕ , θ e ψ a um entrada em degrau em $u3_desacoplado$ Figura 5 – Resposta das saídas z , ϕ , θ e ψ a um entrada em degrau em $u4_desacoplado$ 

2.2 Controladores Fuzzy

O projeto dos controlador *fuzzy* foi focado na estabilização de atitude e altitude do *drone* modelado de forma a se inserirem no sistema como é mostrado na Figura 6.

Figura 6 – Diagrama do sistema de controle de altitude utilizando controlador fuzzy



Como se pode ver, três controladores agem no sistema com o objetivo de torná-lo imune a distúrbios representados pelas entradas de ruídos. O Controlador-z diz respeito a um controlador de altitude ao passo que os Controlador-phi e Controlador-theta dizem respeito a controladores de atitude que, pelo fato de o quadricóptero ser simétrico em relação aos eixos x e y , puderam ser representados por um único controlador.

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).

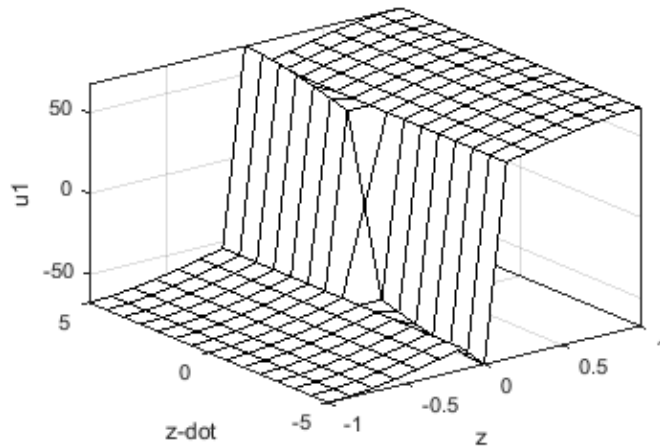
Utilizando o *Fuzzy Logic Toolbox* do MATLAB, cada variável linguística do controlador *fuzzy* foi dividida em três conjuntos: N (negativo), Z (zero) e P (positivo), tomando como base os trabalhos de [Maj e Butkiewicz \(2013\)](#) e [Gao et al. \(2014\)](#). As regras *fuzzy* definidas para este controlador são mostradas no Quadro 1, e a Figura 7 exibe seu equivalente em superfície.

Quadro 1 – 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

Figura 7 – 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 2 e podem ser vistas na superfície de regras mostradas na Figura 8.

Quadro 2 – Regras fuzzy para modelagem do controle de atitude

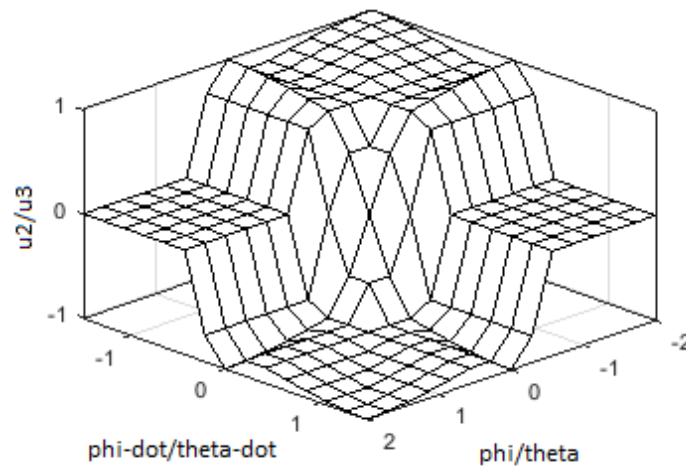
ϕ/θ	$\dot{\phi}/\dot{\theta}$	u_2/u_3
P	P	N
P	Z	N
P	N	Z
N	N	P
N	Z	P
N	P	Z
Z	Z	Z
Z	N	P
Z	P	N

2.3 Controladores Neuro-Fuzzy

2.4 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.

Figura 8 – Superfície das regras do sistema de controle fuzzy para a atitude do quadrotor



Uma vez projetados os controladores fuzzy e neuro-fuzzy, o sistema foi submetido 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 $g = 9,81 \text{ m/s}^2$, $m = 2,3 \text{ kg}$ e $l = 0,5 \text{ m}$.

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 $m = 5 \text{ kg}$, valor este que foi escolhido por variar o parâmetro massa em mais de 100 %.

Os resultados obtidos são mostrados no capítulo seguinte.

Referências

BALAS, C. **Modeling and Linear Control of a Quadrotor**. Dissertação (Mestrado) — Cranfield University, Reino Unido, 2007. Citado 2 vezes nas páginas [1](#) e [2](#).

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 na página [5](#).

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 na página [5](#).

Apêndices

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

```

1      % le arquivo fis referente ao controle de altitude
2      fismat = readfis('fis_altitude.fis');
3
4      % define numero de casos a serem avaliados (treinamento + teste)
5      n = 300;
6      % define conjunto de n entradas aleatorias para o sistema fuzzy
7      % respeitando o range de cada entrada
8      input = zeros(n,2);
9      for i=1:n
10         z_value = rand * 2 - 1;
11         z_dot_value = rand * 10 - 5;
12         input(i,:) = [ z_value z_dot_value ];
13     end
14
15     % avalia resposta fuzzy para cada entrada
16     output= evalfis(input,fismat);
17
18     % define data como vetor relacionando cada conjunto de entradas ...
19     % a saida
20     % - obtida pelo sistema fuzzy
21     data = [];
22     for i=1:n
23         data(i,:) = [ input(i,:) output(i) ];
24     end
25
26     % define que 2/3 dos dados obtidos serao usados para treinamento
27     % e 1/3 sera usado para teste da rede
28     train = data(1:2*n/3,:);    % dados para treinamento
29     test = data(2*n/3+1:n,:);  % dados para validacao do sistema ...
30     treinado
31
32     % gera modelo fuzzy Sugeno a partir do Mamdani modelado
33     sugFIS = mam2sug(fismat);
34     % salva modelo Sugeno em disco com o nome fis_altitude_neuro.fis
35     writefis(sugFIS, 'fis_altitude_neuro.fis');

```

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

```

1      % le arquivo fis referente ao controle de atitude
2      fismat = readfis('fis_atitude.fis');
3
4      % define numero de casos a serem avaliados (treinamento + teste)
5      n = 300;
6      % define conjunto de n entradas aleatorias para o sistema fuzzy
7      % respeitando o range de cada entrada
8      input = zeros(n,2);
9      for i=1:n
10         phi_value = rand * 4 - 2;
11         phi_dot_value = rand * 3 - 1.5;
12         input(i,:) = [ phi_value phi_dot_value ];
13     end
14
15     % avalia resposta fuzzy para cada entrada
16     output= evalfis(input,fismat);
17
18     % define data como vetor relacionando cada conjunto de entradas ...
19     % a saída
20     % obtida pelo sistema fuzzy
21     data = [];
22     for i=1:n
23         data(i,:) = [ input(i,:) output(i) ];
24     end
25
26     % define que 2/3 dos dados obtidos serao usados para treinamento
27     % e 1/3 sera usado para teste da rede
28     train = data(1:2*n/3,:);      % dados para treinamento
29     test = data(2*n/3+1:n,:);    % dados para validação do sistema ...
30     treinado
31
32     % gera modelo fuzzy Sugeno a partir do Mamdani modelado
33     sugFIS = mam2sug(fismat);
34     % salva modelo Sugeno em disco com o nome fis_atitude_neuro.fis
35     writefis(sugFIS, 'fis_atitude_neuro.fis');

```