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 $g = 9{,}8$ 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.

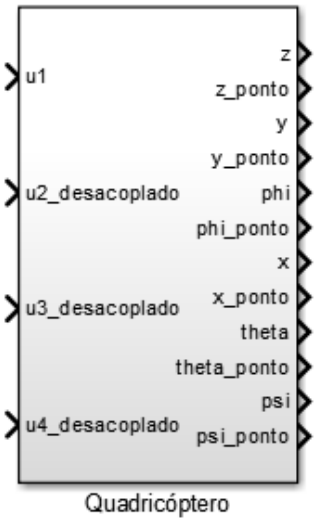
2 Desenvolvimento do Trabalho

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, ϕ, θ, ψ indicadas por x, y, z, phi, theta e 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 psi_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 $u1$

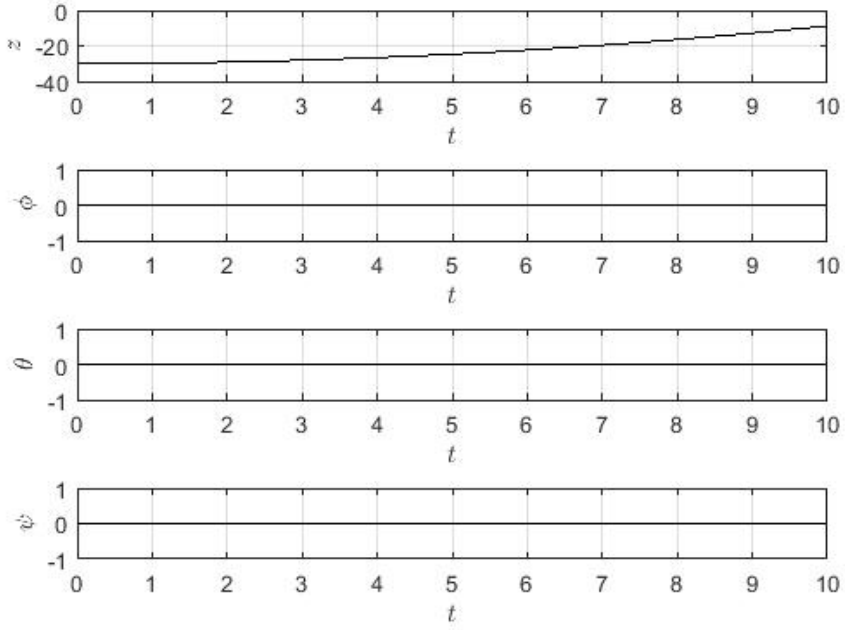
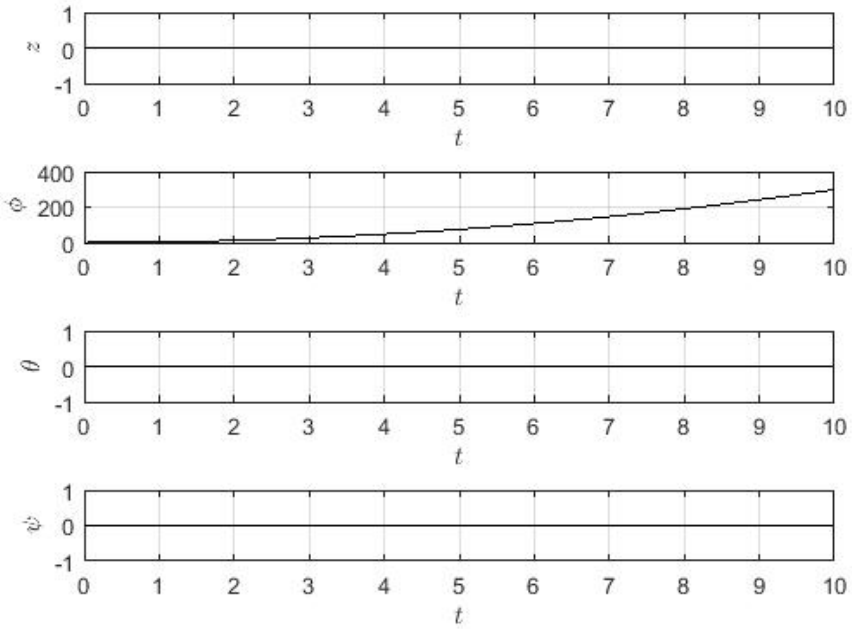


Figura 3 – Resposta das saídas z, ϕ, θ e ψ a um entrada em degrau em $u2\_desacoplado$



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 $u1$ somente interfira na variável de configuração z; $u2\_desacoplado$ em ϕ; $u3\_desacoplado$ em θ; e $u4\_desacoplado$ em ϕ.

Figura 4 – Resposta das saídas z, ϕ, θ e ψ a um entrada em degrau em $u3\_desacoplado$

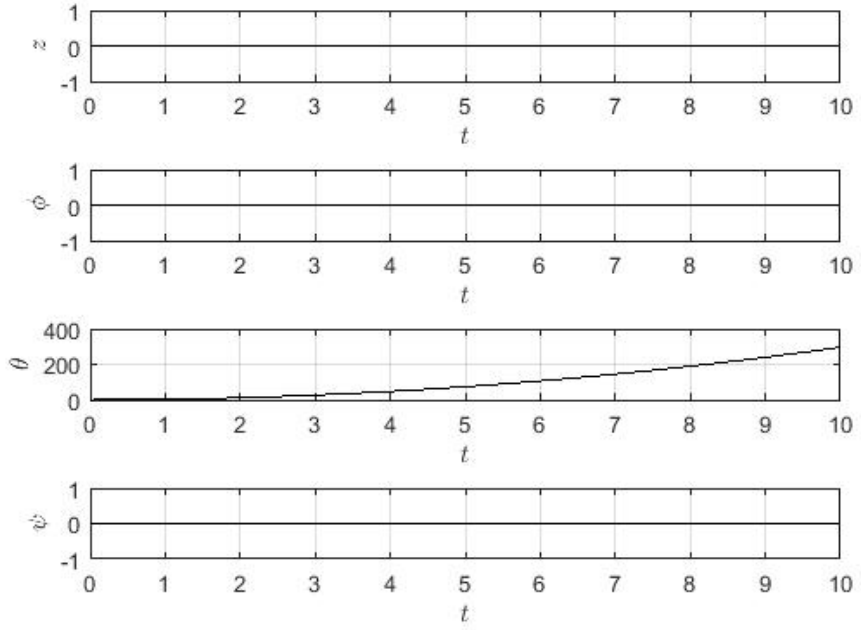
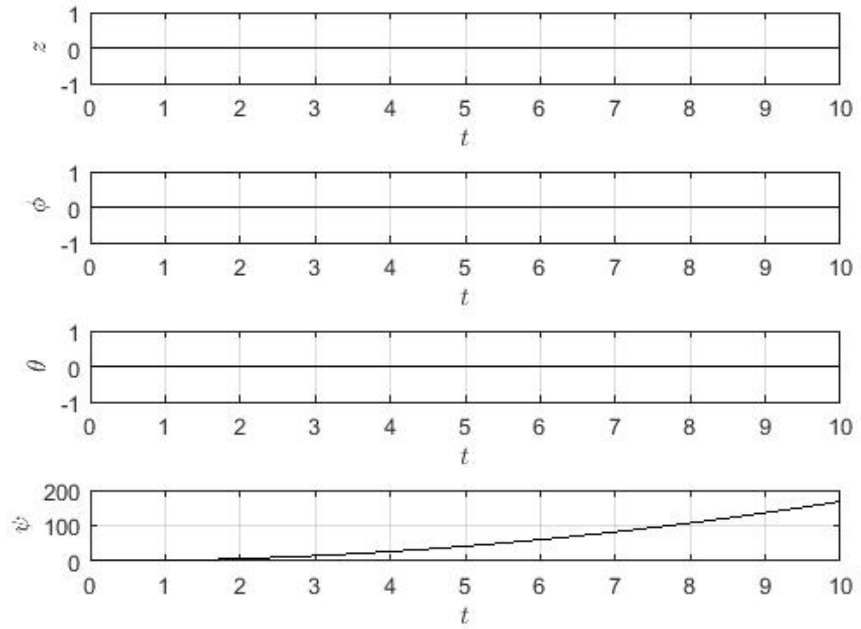


Figura 5 – Resposta das saídas z, ϕ, θ e ψ a um entrada em degrau em $u4\_desacoplado$



2.2 Controladores Fuzzy

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.

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

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

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');
```