

## Resumo

A gama de aplicação de quadricópteros tem crescido substancialmente nos últimos anos, sendo eles utilizados inclusive para fins militares. Entretanto, para que o seu uso seja possível, se faz necessário o desenvolvimento de controladores que permitam seu funcionamento adequado de forma a permitir sua estabilidade. Para tanto, foram desenvolvidos ao longo deste trabalho, para estabilizar a atitude e altitude de um *drone*, dois controladores baseados em técnicas diferentes de Inteligência Computacional (IC): *fuzzy* e *neuro-fuzzy*. Os resultados obtidos por eles são comparados e mostra-se que ambos estabilizam o sistema de forma eficiente e, ainda, que com o controlador *neuro-fuzzy* obtiveram-se melhores resultados, além de ele oferecer melhor desempenho energético principalmente sobre um sistema com ruído de medição. Com isto, mostra-se que técnicas de IC podem ser aplicadas no projeto e implementação de controladores eficientes para sistemas não-lineares complexos e que o poder de aprendizado das RNAs é, de fato, capaz de melhorar a performance de um sistema *fuzzy*, o que pode ser constatado pelo melhor desempenho obtido pelo controlador *neuro-fuzzy*.

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

## Abstract

The range of applications of drones have grown substantially in the last years, as they have been used even for military services. However, in order to make their use possible, the development of controllers which allow their suitable operation to guarantee their stability is needed. Therefore, in this dissertation, two controllers based on Computational Intelligence (CI) techniques, fuzzy and neuro-fuzzy, were developed to stabilize a drone's attitude and altitude. The results that they obtained were compared and it's shown that both were able to stabilize the system with efficiency and yet that the neuro-fuzzy reached better results including a better energy efficiency even in the case in which a measurement noise was incorporated to the system. From it, one can see that CI techniques may be applied in the project and implementation of efficient controller for complex non linear systems and that the Neural Networks's power of learning is able to increase a fuzzy system's performance indeed, what can be seen from the better results obtained using the neuro-fuzzy controller.

**Keywords:** drone. neuro-fuzzy control. computational intelligence.

# Sumário

<b>1 – Metodologia</b> . . . . .	<b>1</b>
<b>2 – Desenvolvimento do Trabalho</b> . . . . .	<b>2</b>
2.1 Verificação do Desacoplamento das Entradas . . . . .	2
2.2 Controladores Fuzzy . . . . .	4
2.3 Controladores Neuro-Fuzzy . . . . .	6
2.4 Experimentos Realizados . . . . .	11
<b>3 – Resultados</b> . . . . .	<b>12</b>
3.1 Análise Consolidada dos Resultados . . . . .	24
<b>4 – Conclusão</b> . . . . .	<b>26</b>
4.1 Trabalhos futuros . . . . .	27
<b>Referências</b> . . . . .	<b>28</b>
 <b>Apêndices</b>	 <b>29</b>
<b>APÊNDICE A – Código para Criação de Modelo Neuro-Fuzzy para Altitude e Definição de Dados para Treinamento</b> . . . . .	<b>30</b>
<b>APÊNDICE B – Código para Criação de Modelo Neuro-Fuzzy para Atitude e Definição de Dados para Treinamento</b> . . . . .	<b>31</b>

# 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 do valor de  $z$  para verificar se o controle implementado se mostra robusto.

Em todos os casos, o algoritmo usado pelo Simulink® para a resolução de equações diferenciais ordinárias foi o Dormand-Price num contexto contínuo.

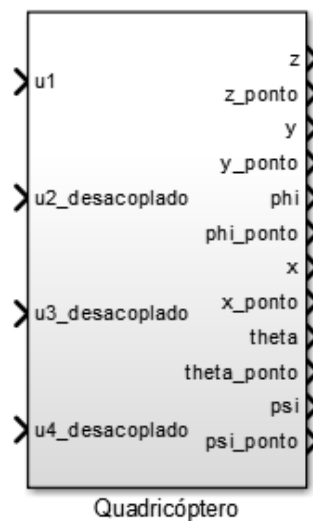
## 2 Desenvolvimento do Trabalho

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

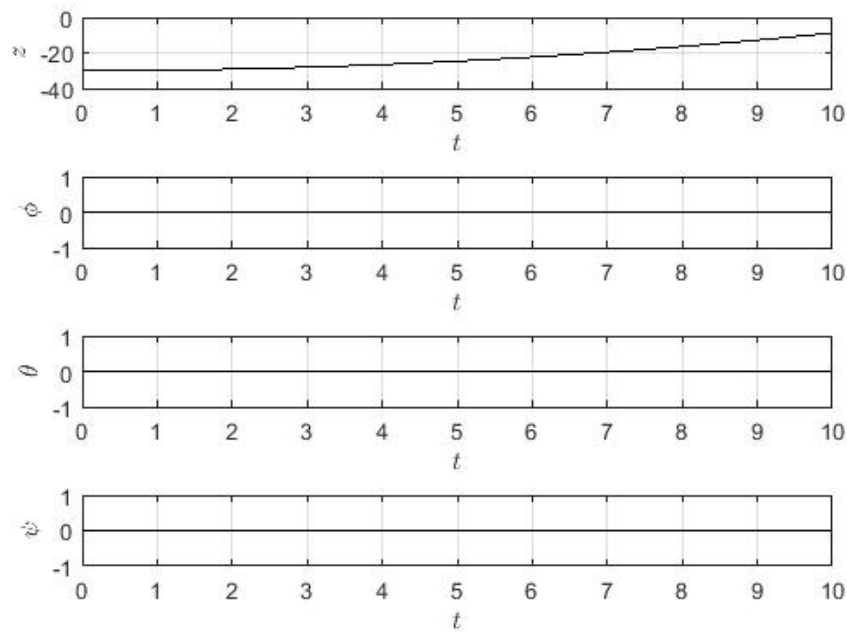
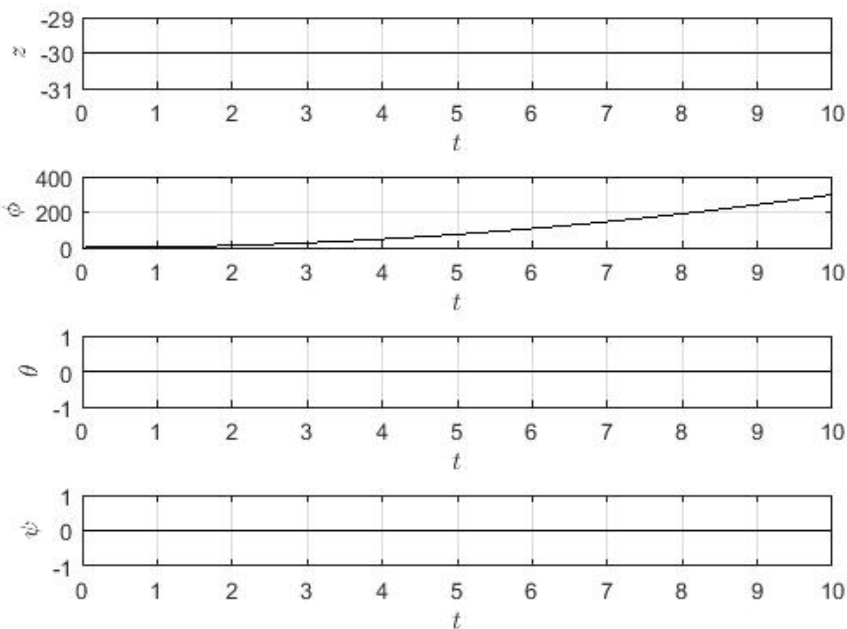
A representação do quadricóptero criada no Simulink® seguindo a modelagem de Balas (2007) é mostrada na Figura 1.

Figura 1 – Representação do quadricóptero no *Simulink*

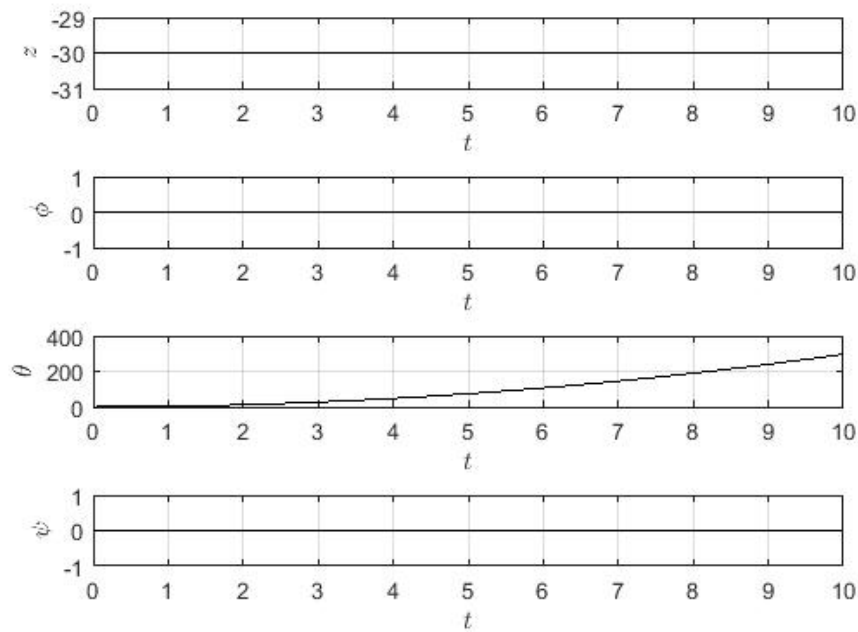
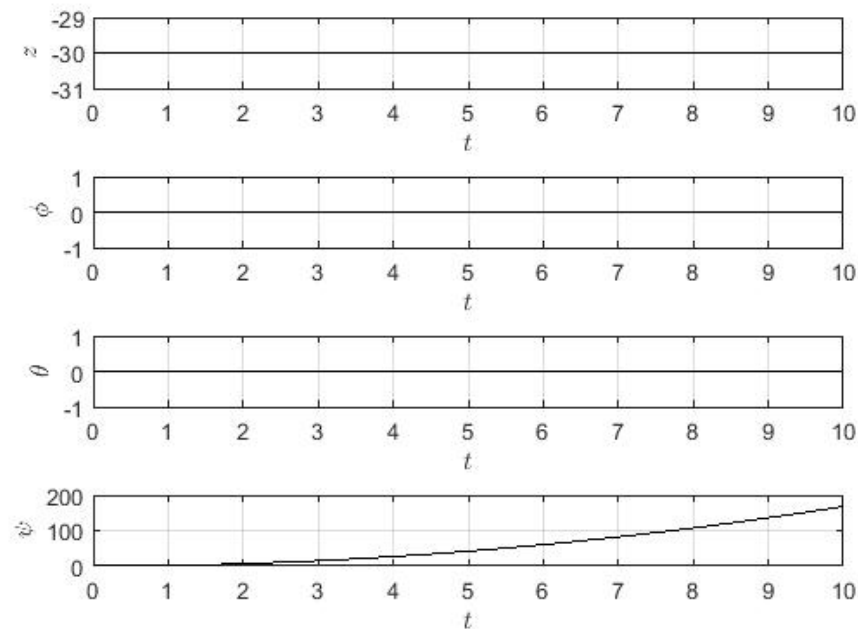


Como se pode ver, o sistema inclui os quatro sinais de entrada seguindo o desacoplamento desenvolvido ( $u1$ ,  $u2\_desacoplado$ ,  $u3\_desacoplado$  e  $u4\_desacoplado$ ) e com as doze saídas referentes às seis variáveis de configuração  $x, y, z, \phi, \theta, \psi$  indicadas por  $x, y, z, \phi, \theta, \psi$ , 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  $\psi\_ponto$ .

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  $z$ ,  $\phi$ ,  $\theta$  e  $\psi$  a um entrada em degrau em  $u_1$ Figura 3 – Resposta das saídas  $z$ ,  $\phi$ ,  $\theta$  e  $\psi$  a um entrada em degrau em  $u_{2\_desacoplado}$ 

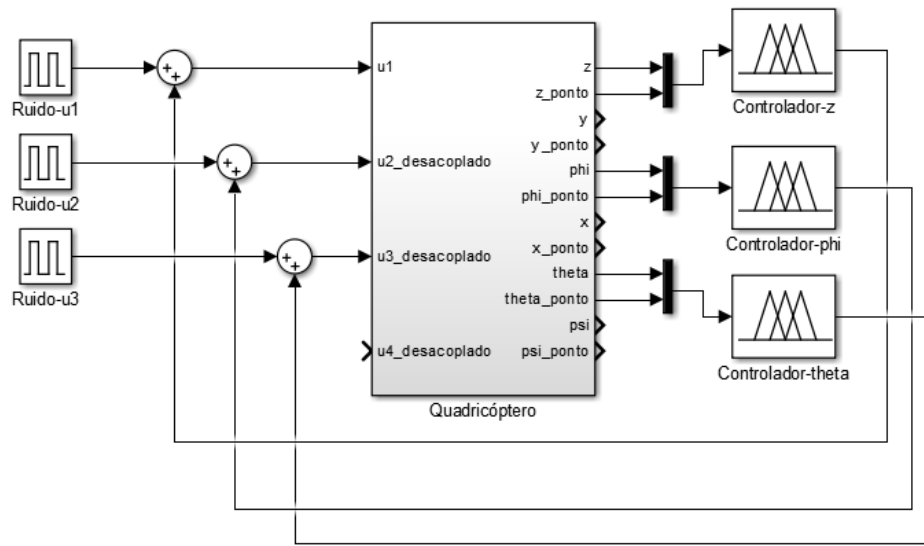
Como se pode ver, cada entrada afeta uma única saída e cada saída é afetada apenas por uma entrada. Com isso, mostra-se o desacoplamento existente que faz com que a entrada  $u_1$  somente interfira na variável de configuração  $z$ ;  $u_{2\_desacoplado}$  em  $\phi$ ;  $u_{3\_desacoplado}$  em  $\theta$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

Figura 4 – Resposta das saídas  $z$ ,  $\phi$ ,  $\theta$  e  $\psi$  a um entrada em degrau em  $u3\_desacoplado$ Figura 5 – Resposta das saídas  $z$ ,  $\phi$ ,  $\theta$  e  $\psi$  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.

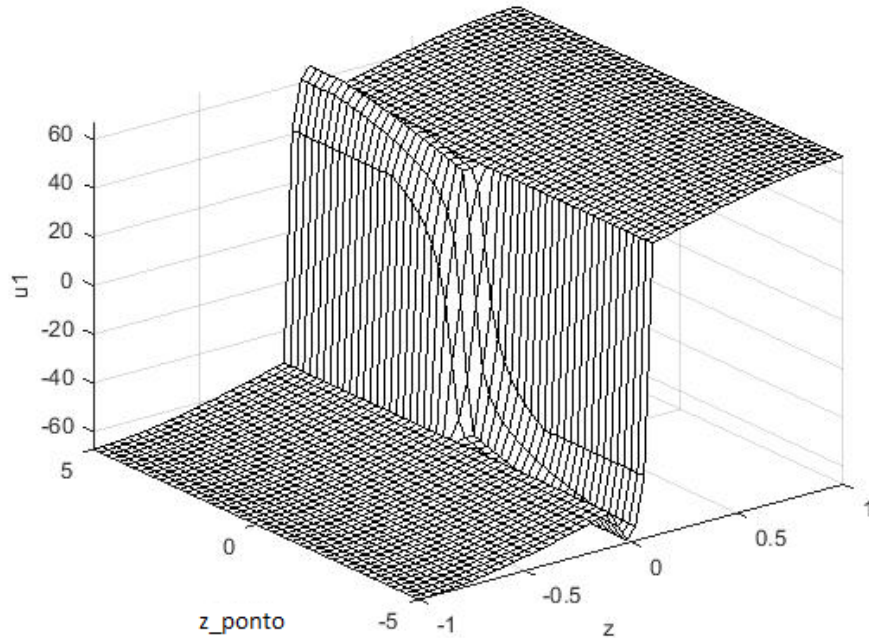
Quadro 1 – Regras fuzzy para modelagem do controle de altitude

$z$	$\dot{z}$	$u_1$
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 quadricóptero



( $\phi$  ou  $\theta$ ) 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.

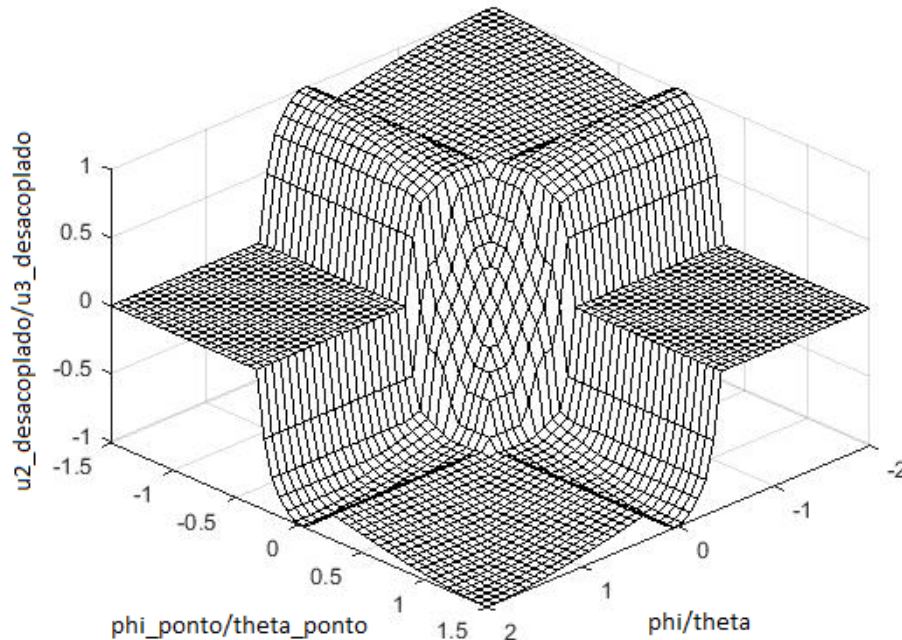
Quadro 2 – Regras fuzzy para modelagem do controle de atitude

$\phi/\theta$	$\dot{\phi}/\dot{\theta}$	$u_2/u_3$
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

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

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



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<sup>1</sup> pares de entradas e cada um deles foi submetido ao processo de inferência *fuzzy* utilizando o controlador previamente modelado e descrito na Seção 2.2. Dois terços desses dados foram utilizados para gerar o conjunto de treinamento, representado pela variável `train` e o um terço restante foi armazenado na variável `test` e utilizado para validação do treinamento. Então, utilizando o comando `mam2sug` do MATLAB, foi gerado um modelo fuzzy Sugeno a partir do Mamdani que havia sido modelado e este novo arquivo foi salvo sob o nome `fis_altitude_neuro.fis`.

Feito isto, utilizou-se o comando `anfisedit` para abrir o *Neuro-Fuzzy Designer* do MATLAB, cuja interface é mostrada na Figura 9. 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 10, pode ser obtida clicando no botão *Structure* logo

<sup>1</sup> Este valor foi arbitrado por corresponder a uma quantidade razoável para treinar a RNA sem que se alcance o sobre-parametrização, conhecido como *overfitting*.

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

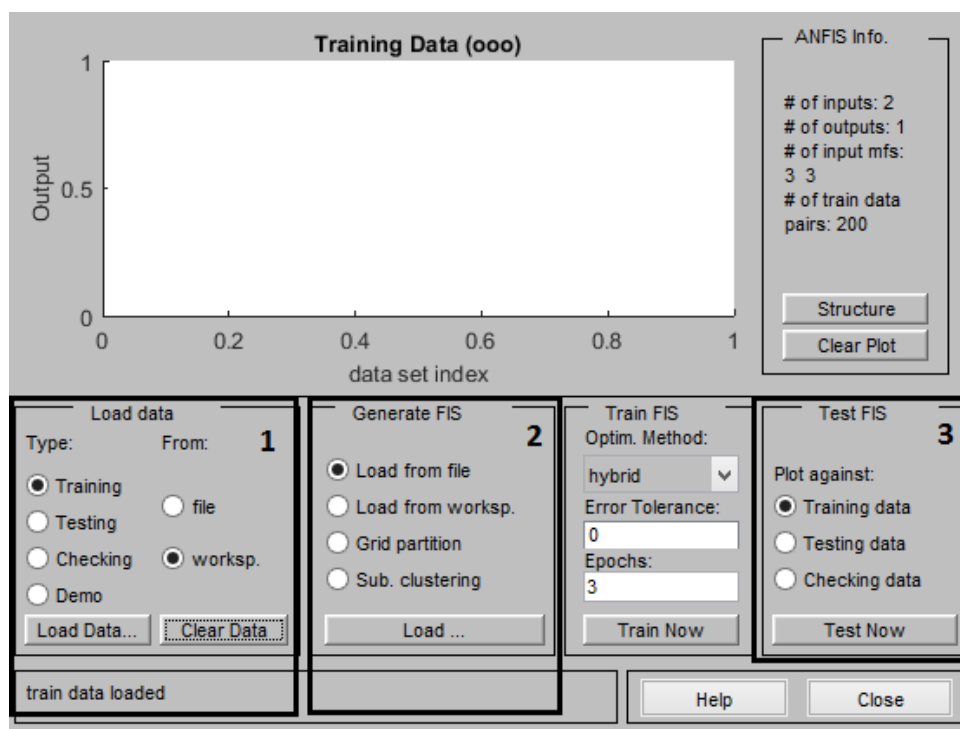
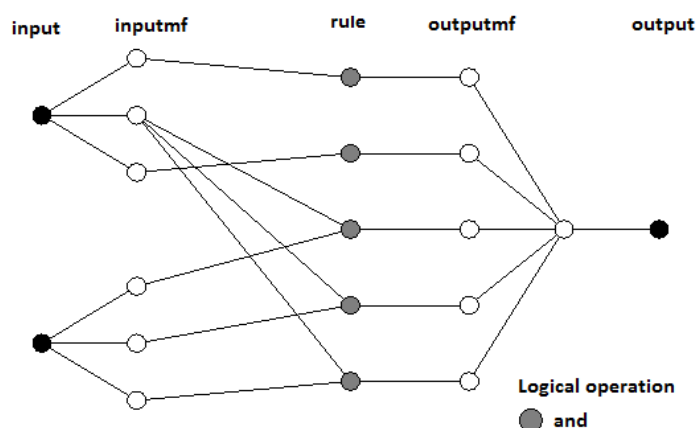


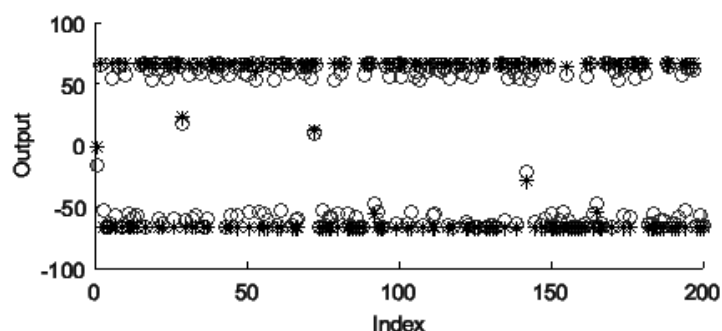
Figura 10 – Diagrama da RNA referente ao controlador neuro-fuzzy para altitude



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

A Figura 11 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 11 – 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 12 e 13 mostram o diagrama da RNA referente ao controlador neuro-fuzzy para atitude e o resultado obtido pelo seu treinamento respectivamente.

Figura 12 – Diagrama da RNA referente ao controlador neuro-fuzzy para atitude

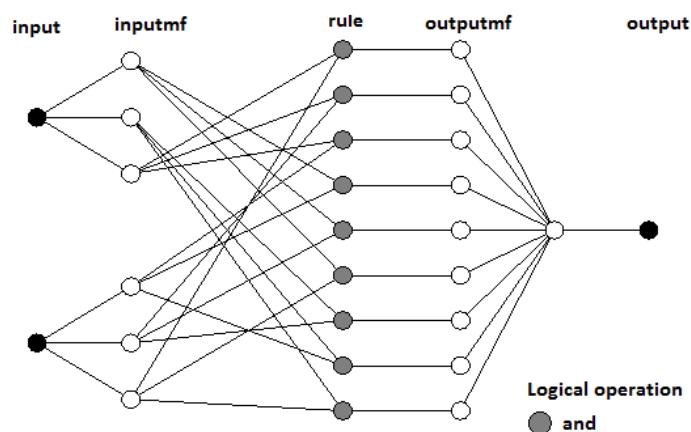
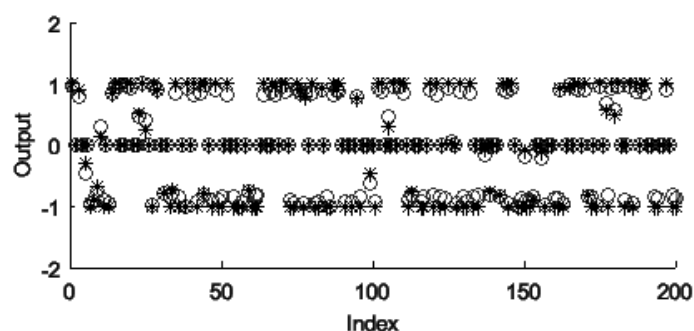


Figura 13 – Resultado obtido pelo treinamento da RNA para controle de atitude



O processo de treinamento determina o comportamento dos controladores neuro-fuzzy projetados, cujas superfícies de regras são exibidas nas Figuras 14 e 15.

Figura 14 – Superfície das regras do sistema de controle neuro-*fuzzy* para a altitude do quadricóptero

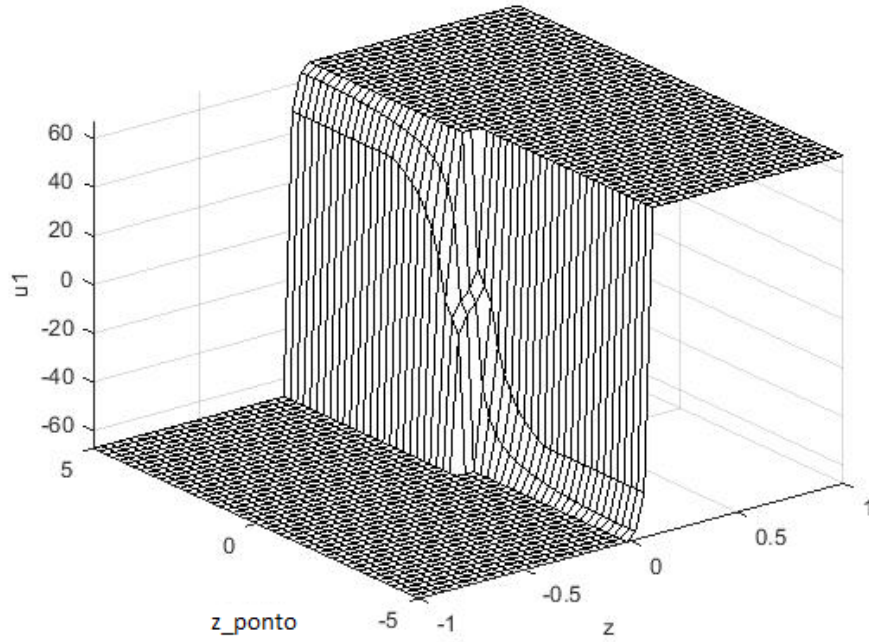
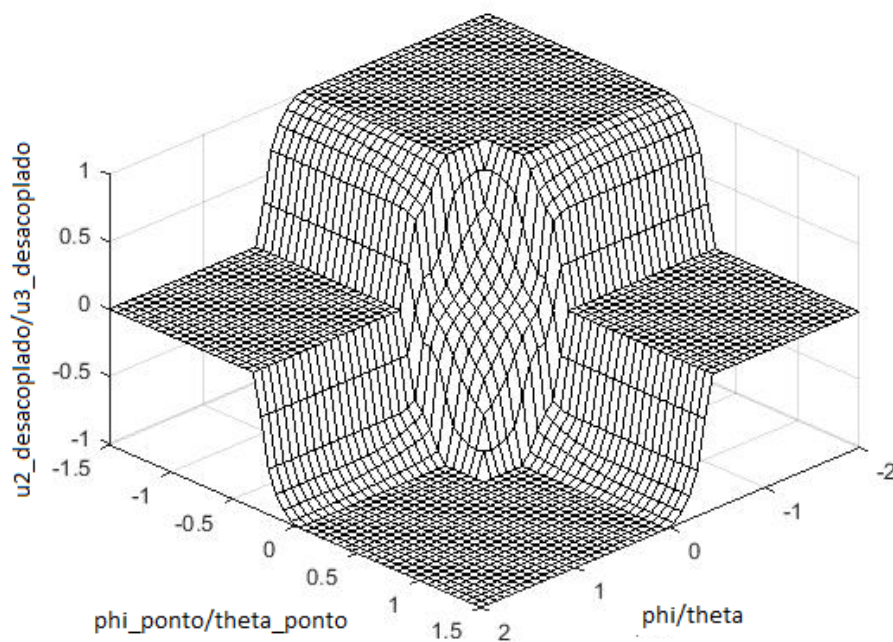


Figura 15 – Superfície das regras do sistema de controle neuro-*fuzzy* para a atitude do quadricóptero



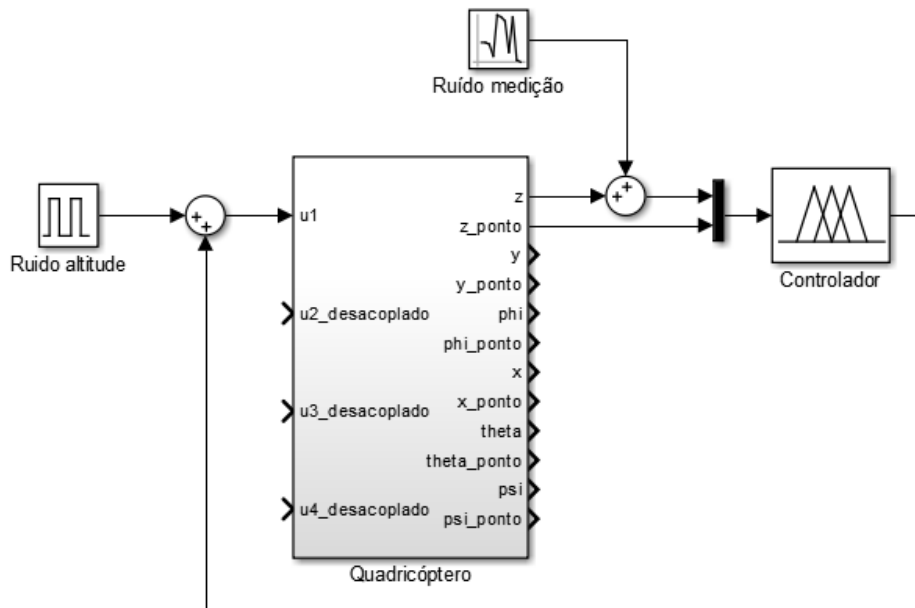
## 2.4 Experimentos Realizados

Uma vez projetados os controladores *fuzzy* e *neuro-fuzzy*, o sistema foi submetido a distúrbios em pulso em atitude e altitude para verificar o funcionamento deles sob condições similares às mostradas quando nenhum controle atua 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 quadricóptero é  $m = 5 \text{ kg}$ , valor este que foi escolhido por variar o parâmetro massa em mais de 100 %.

Por fim, foi testado o funcionamento do sistema quando um ruído de medição passa a fazer parte dele. Para tanto, um sinal aleatório de ruído foi somado ao valor real obtido de  $z$ , como mostrado na Figura 16.

Figura 16 – Representação do quadricóptero no *Simulink* na simulação envolvendo ruído de medição de  $z$



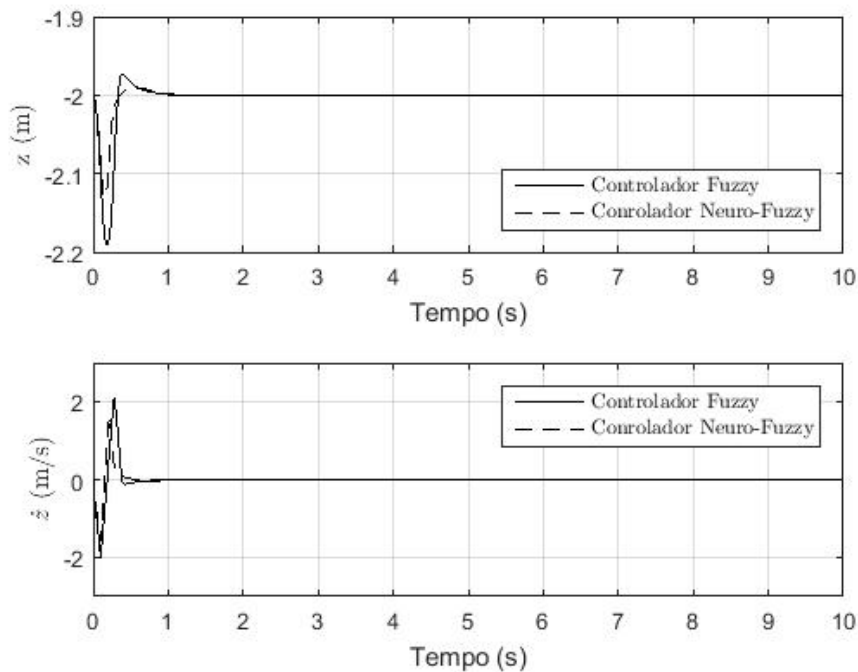
Os resultados obtidos são mostrados no capítulo seguinte.



### 3 Resultados

A Figura 17 mostra a posição no eixo vertical ( $z$ ) do *drone*, bem como sua variação no sistema em que atua o controlador *fuzzy* projetado. Como se pode ver, o distúrbio foi devidamente controlado, fazendo com que o quadricóptero retornasse à posição inicial  $z = -2$  m e também ao repouso<sup>1</sup> representado por  $\dot{z} = 0$  m/s. Nesta figura, entretanto, não fica tão clara a diferença de desempenho dos controladores *fuzzy* e *neuro-fuzzy*, aspecto que pode ser claramente verificado na Figura 18. Como se pode ver, tanto para  $z$  quanto para  $\dot{z}$ , o *neuro-fuzzy* apresenta desempenho melhor. No controle sobre a posição  $z$ , o controlador *neuro-fuzzy* apresentou redução do tempo de convergência em 29%, e da variação do sistema em 31%, além de eliminar a sobrelevação apresentada pelo *fuzzy*. Já sobre a velocidade  $\dot{z}$ , apresentou uma redução no tempo de convergência de 29% além melhorar a variação do sistema em 23%. A partir destes resultados, verifica-se que o controlador *neuro-fuzzy* fez com que o distúrbio fosse melhor absorvido e que sua correção ocorresse mais rapidamente.

Figura 17 – Comparação da resposta das saídas  $z$  e  $\dot{z}$  no controle de altitude *fuzzy* e *neuro-fuzzy* para o sistema com massa  $m = 2$  kg



Já a Figura 19 mostra a ação de ambos os controladores. O controlador *neuro-fuzzy* apresentou melhor resultado quanto ao gasto energético, apresentando um gasto 14% menor do que o *fuzzy*.

<sup>1</sup> i.e. velocidade nula

Figura 18 – Comparação em mais detalhes da resposta das saídas  $z$  e  $\dot{z}$  no controle de altitude fuzzy e neuro-fuzzy para o sistema com massa  $m = 2$  kg

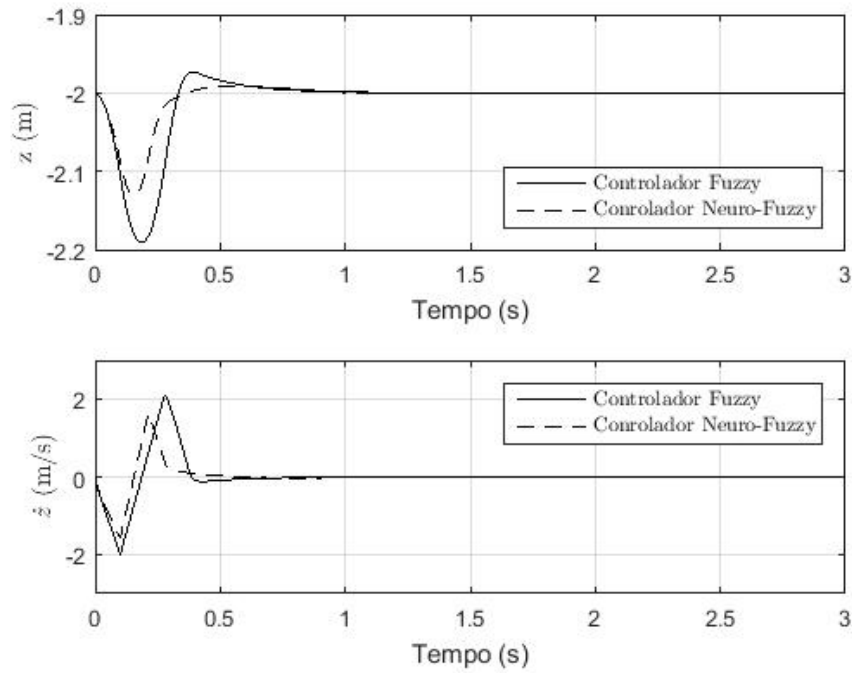
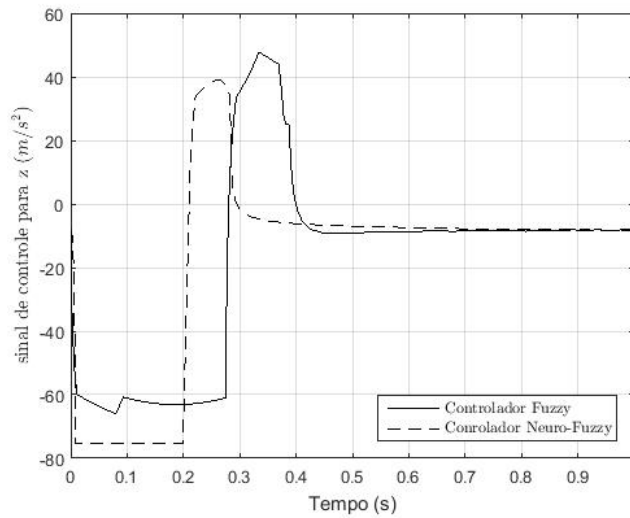


Figura 19 – Comparação da ação dos controladores *fuzzy* e *neuro-fuzzy* na estabilização em altitude do sistema com massa  $m = 2$  kg



Já as Figuras 20 e 21 mostram a estabilidade de atitude em torno dos eixos  $x$  e  $y$  (i.e em relação ao plano horizontal  $XY$ ), representados por  $\theta$  e  $\phi$  respectivamente. Como se pode ver, ambos os estados são devidamente controlados e, com isto, o *drone* volta à estabilidade horizontal, com ângulos e velocidades angulares nulas no estado permanente.

A partir das Figuras 22 e 23, que mostram as respostas obtidas em mais detalhes,



Figura 20 – Comparação da resposta das saídas  $\phi$  e  $\dot{\phi}$  no controle de atitude *fuzzy* e neuro-*fuzzy* para o sistema com massa  $m = 2$  kg

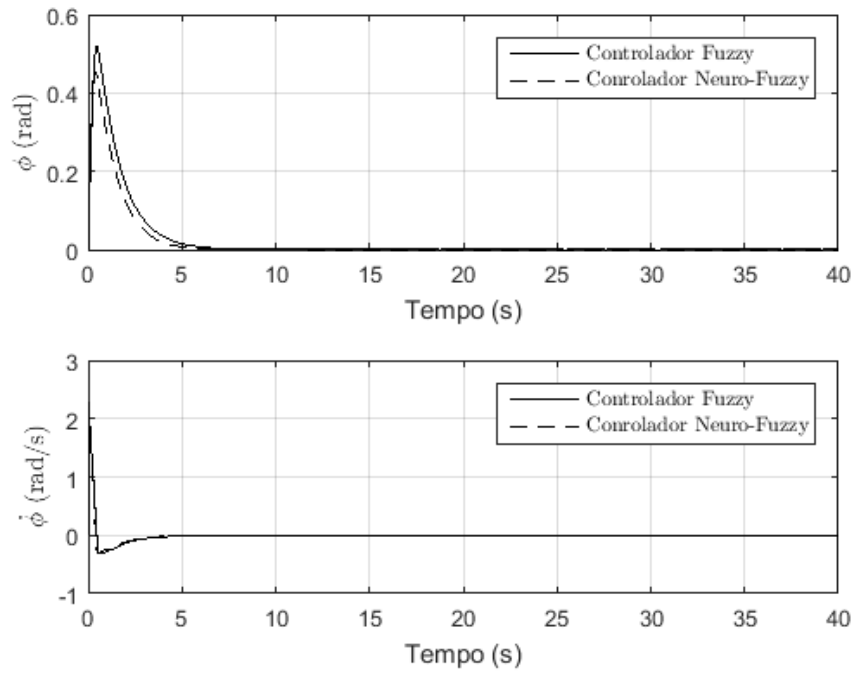
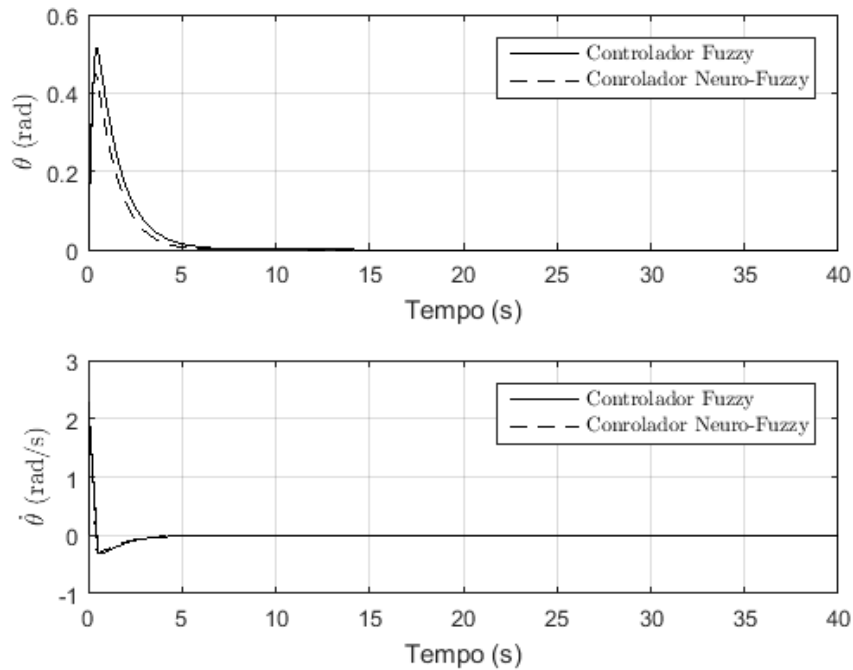


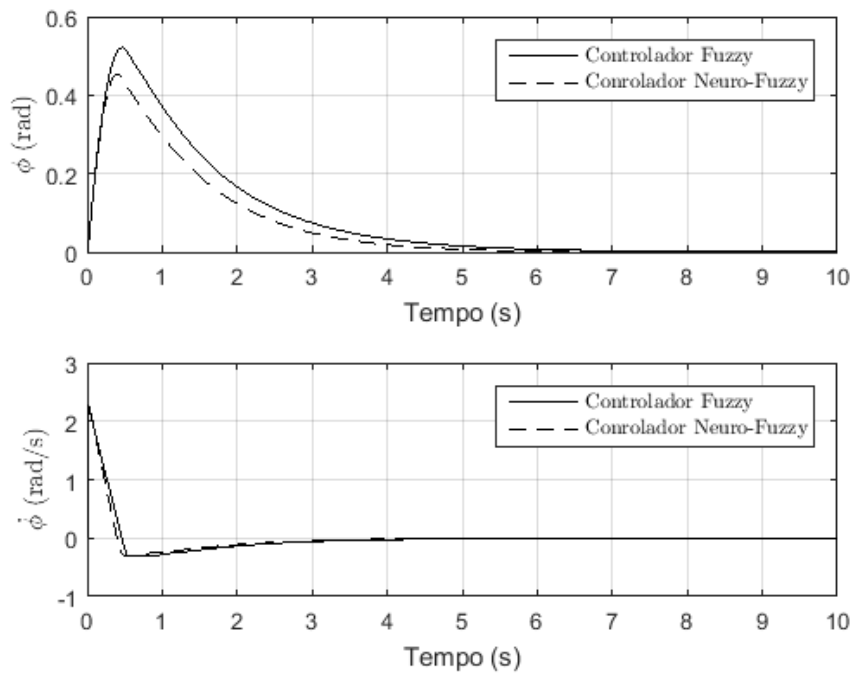
Figura 21 – Comparação da resposta das saídas  $\theta$  e  $\dot{\theta}$  no controle de atitude *fuzzy* e neuro-*fuzzy* para o sistema com massa  $m = 2$  kg



pode-se ver que, mais uma vez o controlador neuro-*fuzzy* mais uma vez teve desempenho superior ao *fuzzy*, fazendo com que os ângulos  $\phi$  e  $\theta$  convergissem 2% mais rápido, além de reduzir suas variações a 13%. Com relação às velocidades angulares

$\dot{\phi}$  e  $\dot{\theta}$ , foi capaz de reduzir o tempo de convergência em 3%, não afetando a variação nem a sobrelevação apresentada pelo sistema quando estabilizado pelo controlador *fuzzy*. Desta forma, verifica-se que o controle neuro-fuzzy levou o sistema a uma menor variação, representando que o ângulo máximo de inclinação alcançado pelo *drone* é menor e corrigido mais rapidamente.

Figura 22 – Comparação em mais detalhes da resposta das saídas  $\phi$  e  $\dot{\phi}$  no controle de atitude *fuzzy* e neuro-fuzzy para o sistema com massa  $m = 2$  kg



As Figuras 24 e 25 mostra a ação de ambos os controladores no processo de estabilização da atitude do *drone*. Como se pode ver, ambas as respostas são bastante parecidas, devido ao fato de o sistema ser praticamente simétrico em relação aos seus eixos x e y. Em ambos os casos, o controlador neuro-fuzzy apresentou resultado inferior ao fuzzy, consumindo 12% mais energia do que este.

Além da verificação da eficiência dos controladores atuando sobre o sistema para o qual foram projetados, eles foram testados num sistema em que um dos parâmetros foi acrescido de mais de 100 %, com a massa passando de 2,3 kg para 5 kg.

A resposta dos controladores *fuzzy* e neuro-fuzzy para altitude do *drone* nessas circunstâncias são mostradas nas Figuras 26 e 27, sendo que esta segunda é apenas uma forma melhor de comparar a ação dos dois controladores. Como se pode perceber, ambos os controladores levaram à estabilização do sistema, sendo que desta vez cada um obteve desempenho melhor sob determinados aspectos. No controle da posição vertical  $z$ , o neuro-fuzzy apresentou tempo de convergência 57% maior, em parte causado por uma sobrelevação, que não foi apresentada pelo *fuzzy*. Em contrapartida, o

Figura 23 – Comparação em mais detalhes da resposta das saídas  $\theta$  e  $\dot{\theta}$  no controle de atitude *fuzzy* e *neuro-fuzzy* para o sistema com massa  $m = 2$  kg

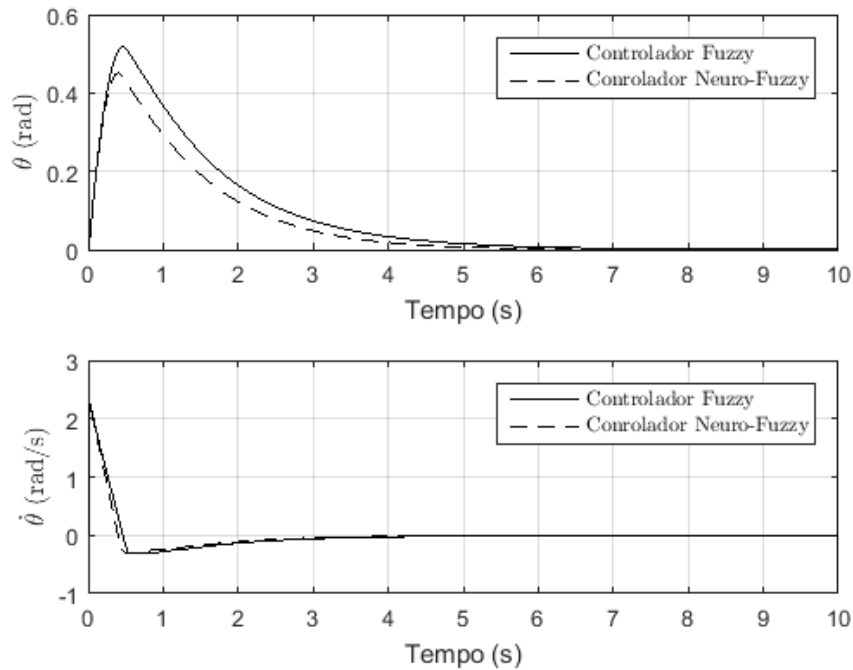
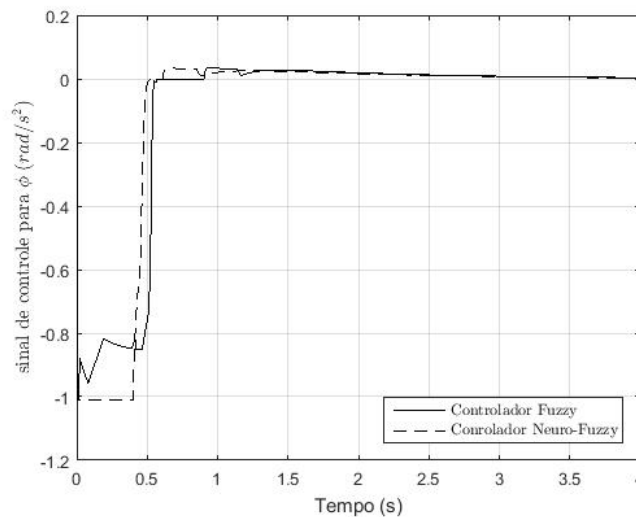


Figura 24 – Comparação da ação dos controladores *fuzzy* e *neuro-fuzzy* na estabilização em atitude do sistema com massa  $m = 2$  kg para variável  $\phi$



neuro-fuzzy apresentou uma menor variação, a reduzindo em 20% se comparado ao *fuzzy*. Com relação à velocidade  $\dot{z}$ , o controlador *neuro-fuzzy* apresentou aumento de 23% no tempo de convergência, mas melhorou o sistema nos quesitos variação e sobrelevação, as reduzindo em 16% e 33%, respectivamente. Esses resultados apontam que o quadricóptero, quando submetido ao controle *neuro-fuzzy*, apresentou movimentos mais suaves até ter sua altitude estabilizada, apesar de ter sido necessário mais tempo para que ela ocorresse.

Figura 25 – Comparação da ação dos controladores *fuzzy* e *neuro-fuzzy* na estabilização em atitude do sistema com massa  $m = 2$  kg para variável  $\theta$

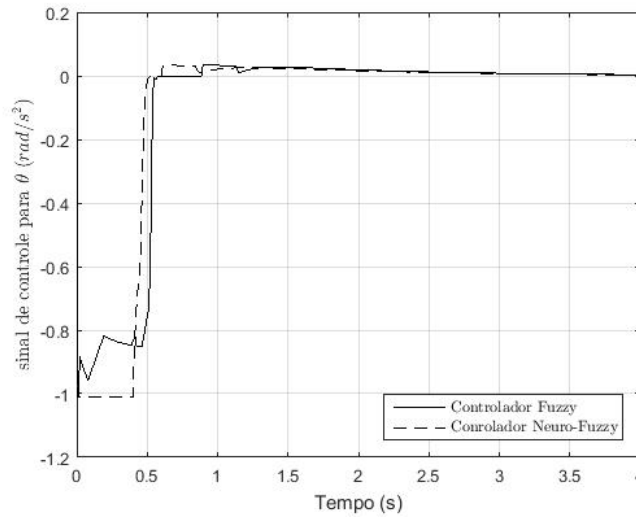
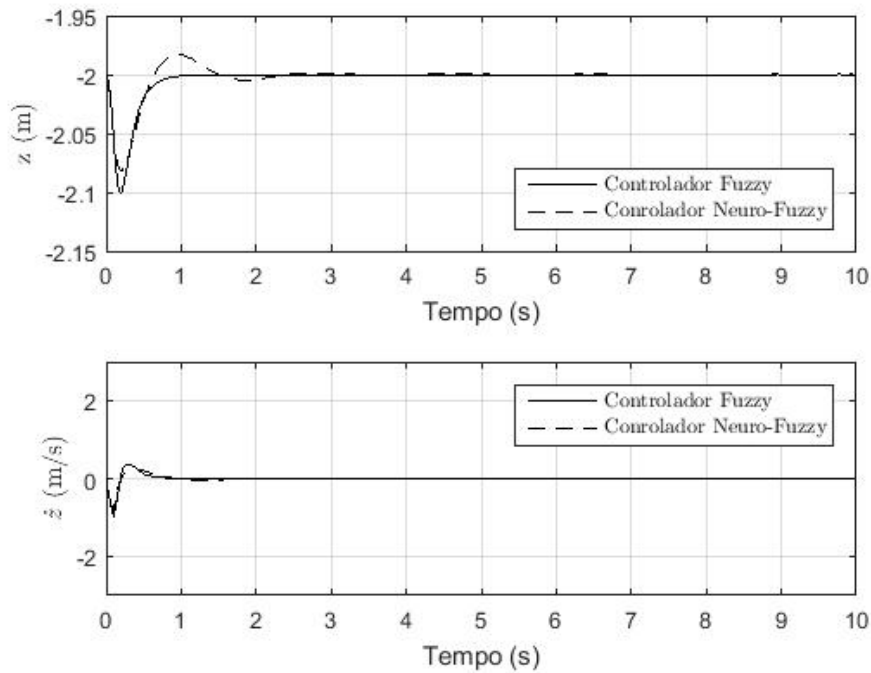


Figura 26 – Comparação da resposta das saídas  $z$  e  $\dot{z}$  no controle de altitude *fuzzy* e *neuro-fuzzy* para o sistema com massa  $m = 5$  kg



A Figura 28 mostra a ação de ambos os controladores sendo que o *neuro-fuzzy* apresentou um resultado ligeiramente superior ao *fuzzy* com relação ao gasto energético, consumindo 7% menos.

As Figuras 29 e 30 mostram os resultados obtidos pelos controladores de atitude no sistema com massa  $m = 5$  kg. Percebe-se que mais uma vez o sistema convergiu ao seu estado de estabilidade com os ângulos nulos e velocidades angulares também

Figura 27 – Comparação em mais detalhes da resposta das saídas  $z$  e  $\dot{z}$  no controle de altitude *fuzzy* e *neuro-fuzzy* para o sistema com massa  $m = 5$  kg

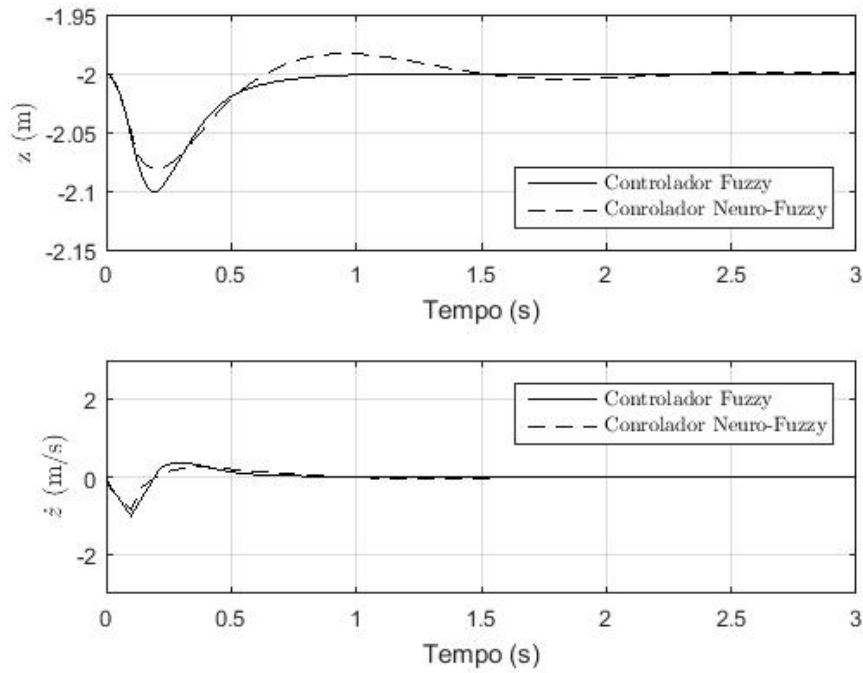
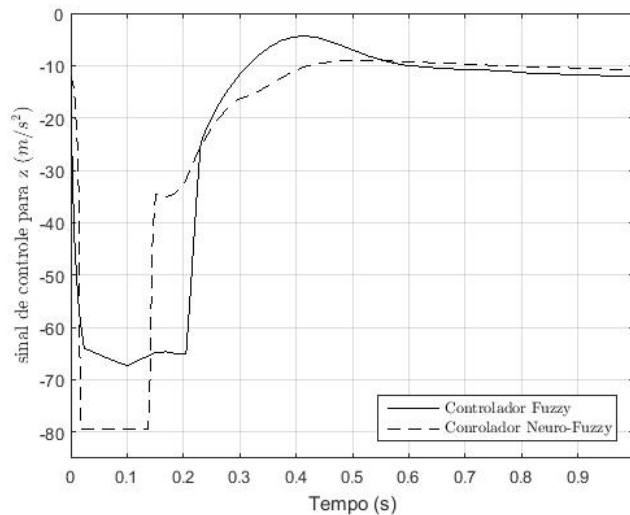


Figura 28 – Comparação da ação dos controladores *fuzzy* e *neuro-fuzzy* na estabilização em altitude do sistema com massa  $m = 5$  kg



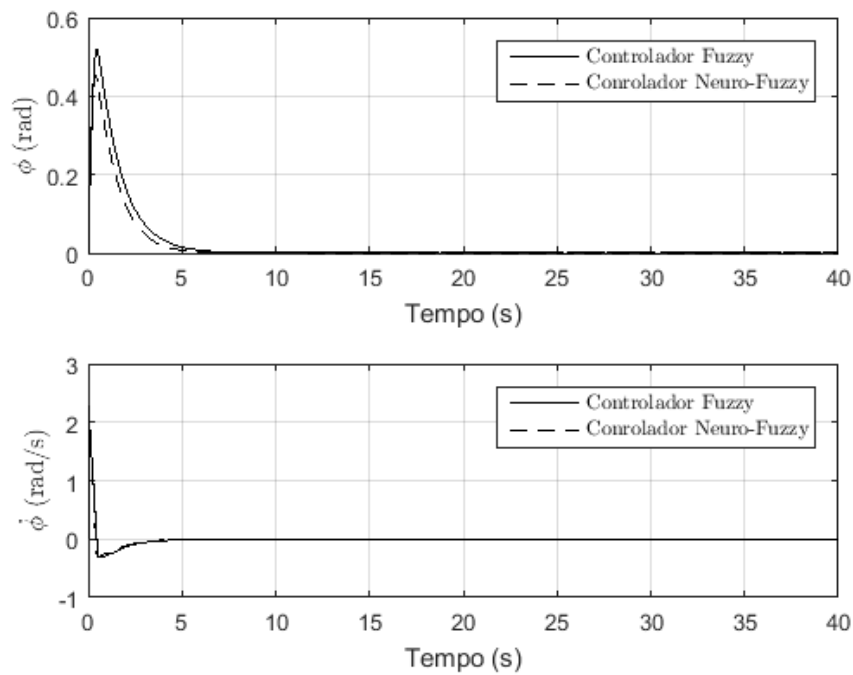
nulas, representando que o quadricóptero, após a ação de controle, tanto *fuzzy* quanto *neuro-fuzzy*, fica estável e com orientação plana<sup>2</sup>.

As Figuras 31 e 32 mostram em mais detalhes as repostas obtidas pelos controladores sobre a atitude do sistema com massa  $m = 5$  kg. A partir delas, nota-se que mais uma vez o controlador *neuro-fuzzy* apresentou desempenho levemente superior

<sup>2</sup> i.e paralela ao plano XY

ao *fuzzy*. Sobre os ângulos  $\phi$  e  $\theta$ , a convergência ocorreu 3% mais rapidamente e a variação apresentada reduziu 14%. Já sobre as velocidades angulares  $\dot{\phi}$  e  $\dot{\theta}$ , a redução de tempo de convergência com o *neuro-fuzzy* foi de 2%, mantendo a mesma sobrelevação e variação oferecidas pelo *fuzzy*. Com isto, mais uma vez o controle *neuro-fuzzy* fez com que o ângulo máximo de inclinação do *drone* fosse inferior ao alcançado pelo sistema controlado pelo *fuzzy*, e além de reduzir o tempo necessário para sua estabilização definitiva.

Figura 29 – Comparação da resposta das saídas  $\phi$  e  $\dot{\phi}$  no controle de atitude *fuzzy* e *neuro-fuzzy* para o sistema com massa  $m = 5$  kg



Por fim, as Figuras 33 e 34 mostram a ação de ambos os controladores no processo de estabilização da atitude do *drone*. Assim como para o sistema com massa  $m = 2$  kg, em ambos os casos o controlador *neuro-fuzzy* apresentou resultado inferior ao *fuzzy*. Desta vez, entretanto, o consumo foi 10% maior.

Já no teste de robustez a ruídos de medição dos controladores desenvolvidos, a Figura 35 mostra a resposta do sistema ao ruído de medição representado pela Figura 36. A mesma resposta é mostrada na Figura 37, porém com maiores detalhes, permitindo uma melhor comparação do sistema controlado pelos diferentes controladores.

Como se pode ver, o controlador *neuro-fuzzy* obteve uma resposta melhor se comparado ao *fuzzy*, apresentando uma redução em 39 % na variação do sistema, além de uma convergência 13 % mais rápida. Além disso, nota-se que, com o controlador *neuro-fuzzy*, o sistema ficou mais estável, apresentando menores variações, apresentando assim, uma melhor resposta ao sistema sujeito a ruídos.

Figura 30 – Comparação da resposta das saídas  $\theta$  e  $\dot{\theta}$  no controle de atitude *fuzzy* e neuro-*fuzzy* para o sistema com massa  $m = 5$  kg

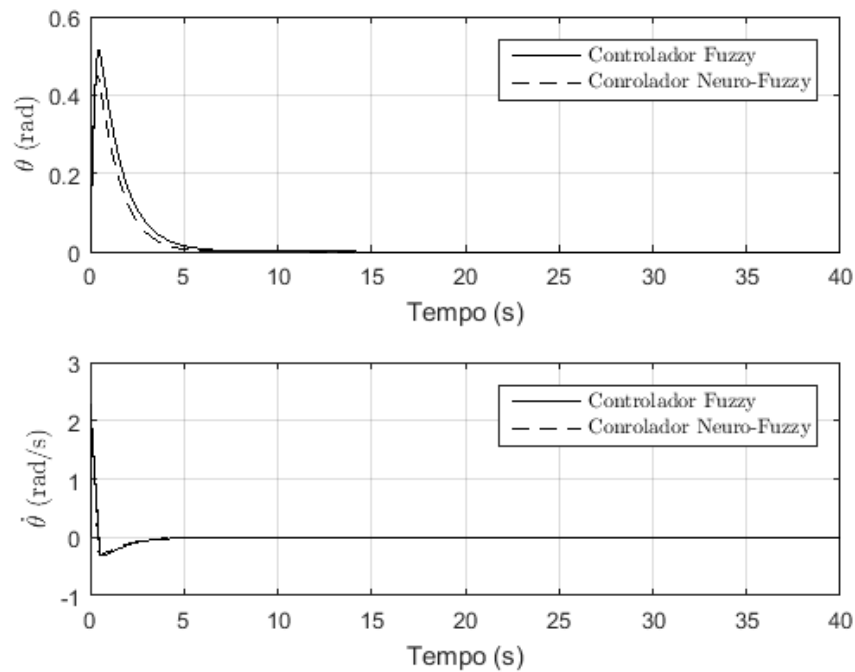
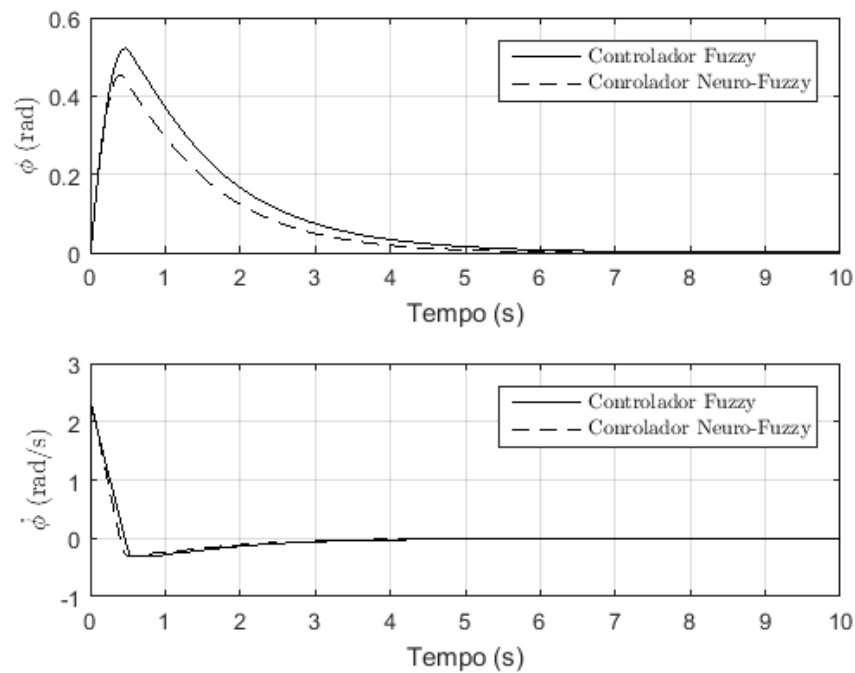


Figura 31 – Comparação da resposta das saídas  $\phi$  e  $\dot{\phi}$  no controle de atitude *fuzzy* e neuro-*fuzzy* para o sistema com massa  $m = 5$  kg



Por fim, a Figura 38 mostra a resposta dos dois controladores ao longo tempo. Como se pode ver, o controlador neuro-*fuzzy* apresentou eficiência energética 33 % superior ao *fuzzy* até o momento de convergência. Além disso, percebe-se que a ação do

Figura 32 – Comparação da resposta das saídas  $\theta$  e  $\dot{\theta}$  no controle de atitude *fuzzy* e neuro-*fuzzy* para o sistema com massa  $m = 5$  kg

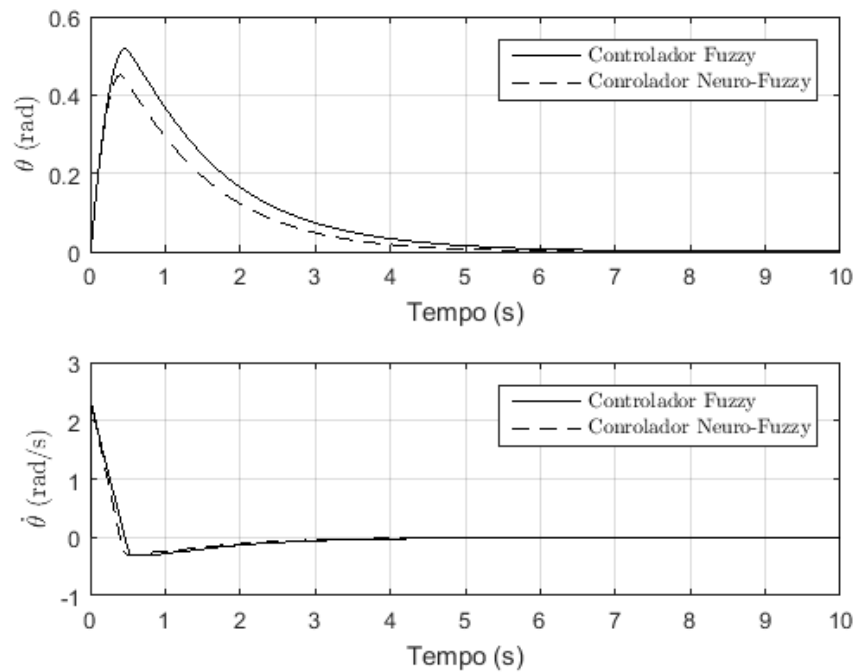
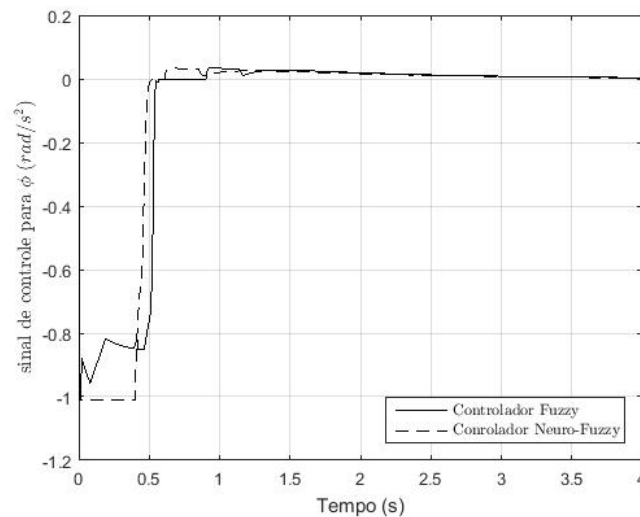


Figura 33 – Comparação da ação dos controladores *fuzzy* e neuro-*fuzzy* na estabilização em atitude do sistema com massa  $m = 5$  kg para variável  $\phi$



controlador neuro-*fuzzy* é muito mais sutil na absorção dos ruídos, levando assim a um grande ganho de desempenho energético a longo prazo.



Figura 34 – Comparação da ação dos controladores *fuzzy* e *neuro-fuzzy* na estabilização em atitude do sistema com massa  $m = 2$  kg para variável  $\theta$

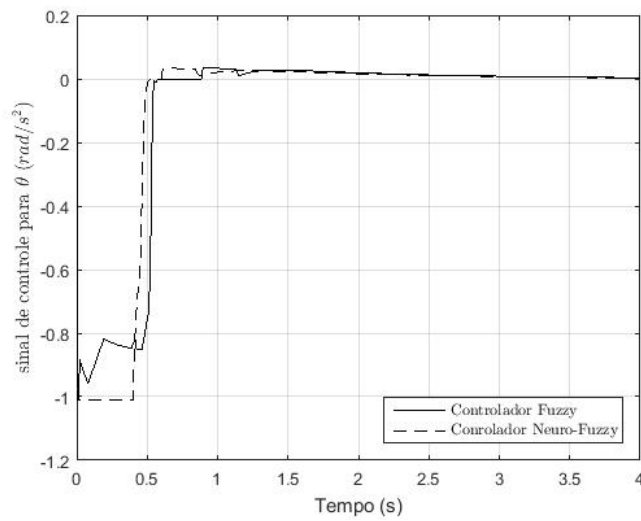


Figura 35 – Comparação da resposta das saídas  $z$  e  $\dot{z}$  no controle de altitude *fuzzy* e *neuro-fuzzy* para o sistema com massa  $m = 2$  kg sujeito a ruídos de medição da variável  $z$

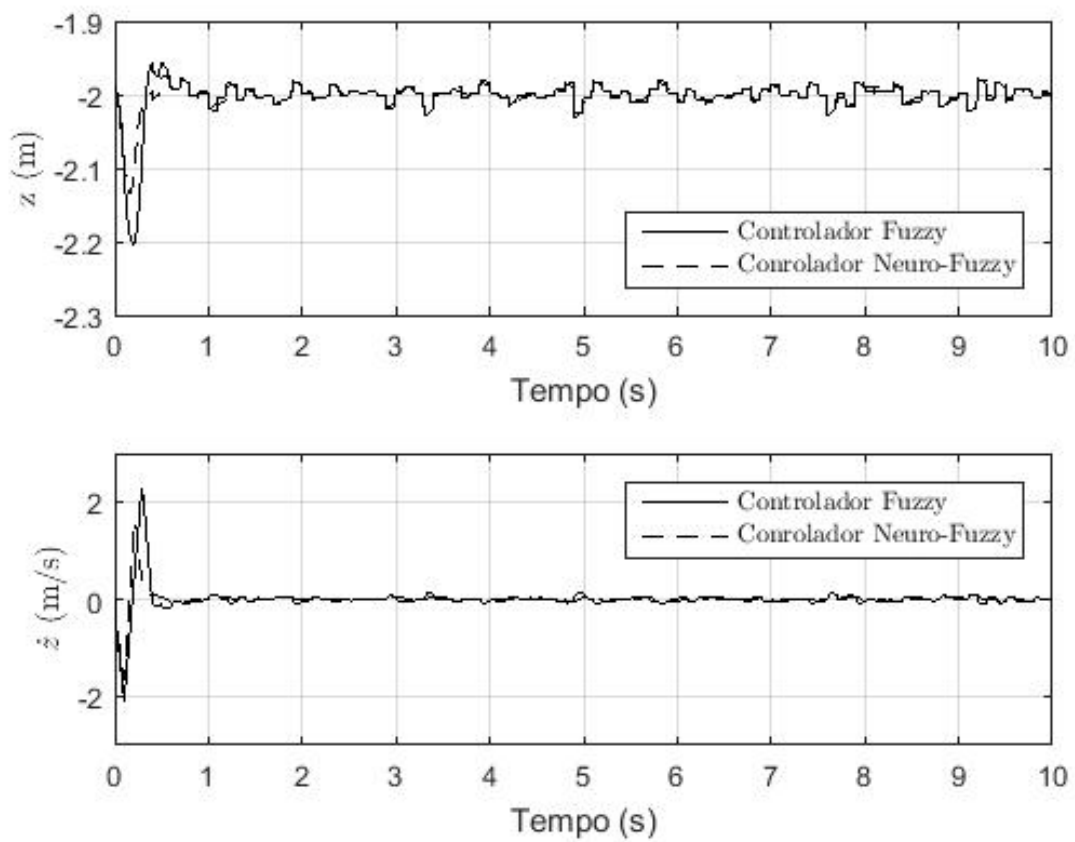


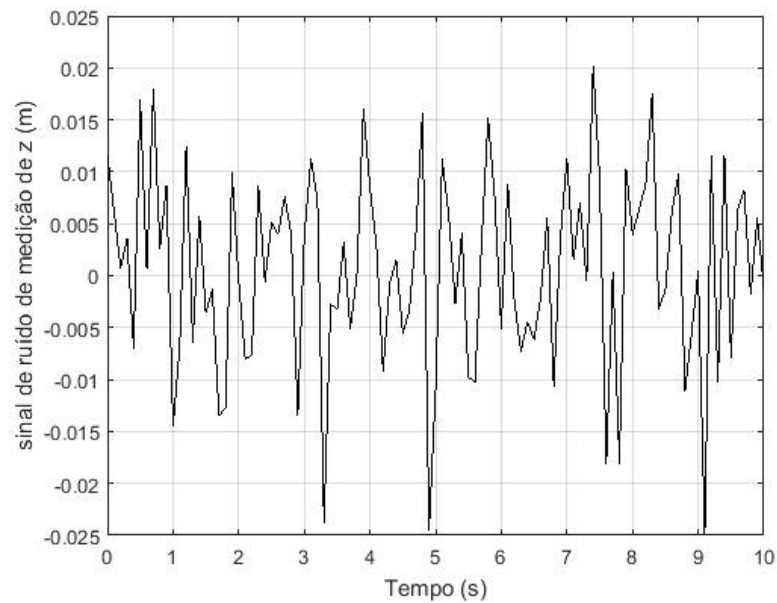
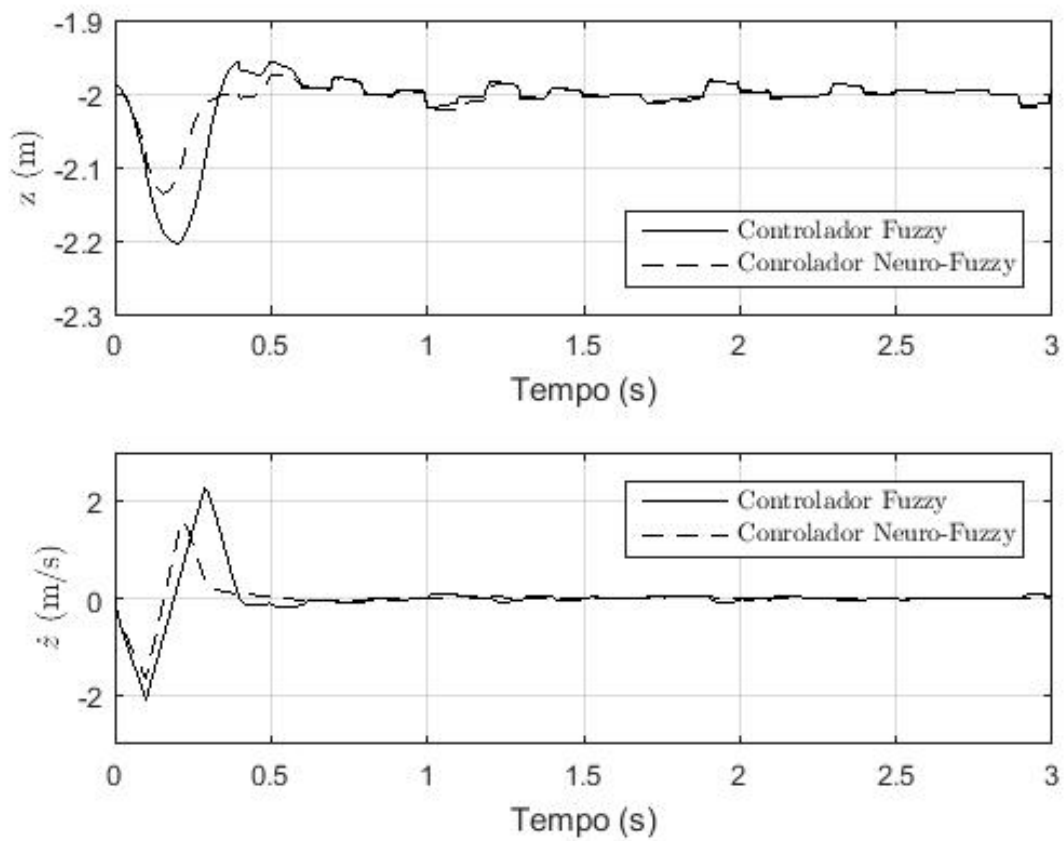
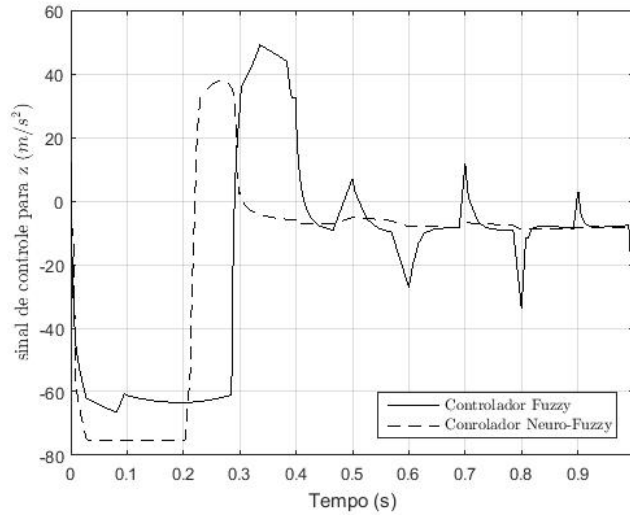
Figura 36 – Sinal de ruído de medição sobre o valor real da variável  $z$ Figura 37 – Comparação em mais detalhes da resposta das saídas  $z$  e  $\dot{z}$  no controle de altitude *fuzzy* e *neuro-fuzzy* para o sistema com massa  $m = 2$  kg sujeito a ruídos de medição da variável  $z$ 

Figura 38 – Comparação da ação dos controladores *fuzzy* e *neuro-fuzzy* na estabilização em altitude do sistema com massa  $m = 2$  kg sujeito a ruídos de medição da variável  $z$



### 3.1 Análise Consolidada dos Resultados

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

Em quase todos os casos, nota-se um comportamento do controlador neuro-fuzzy superior ao do *fuzzy*, o que já era esperado tendo em vista que o primeiro alia o poder do segundo às vantagens das RNAs, fazendo com que, a partir de um treinamento supervisionado utilizando o próprio modelo *fuzzy*, possa se construir um controle mais abrangente e com resposta melhorada. Além disto, verificou-se uma melhora no consumo energético no controle de altitude ao utilizar o controlador neuro-*fuzzy*. No controle de atitude, entretanto, o controlador neuro-*fuzzy* apresentou pior eficiência energética.

Já no experimento envolvendo ruídos de medição, o controlador neuro-*fuzzy* de altitude se mostrou muito superior ao *fuzzy*, apresentando menor variação e menor tempo de convergência além de uma eficiência energética muito melhor, atuando de forma muito sutil para as correções dos ruídos incluídos no sistema ao passo que o *fuzzy* gasta muito mais energia para fazer cada uma dessas correções.

Estes resultados mostram que, de fato, as técnicas de Inteligência Computacional podem ser aplicadas para projetar controladores eficientes e robustos para atuar sobre

sistemas multivariável e que, além disto, o poder de treinamento dos ANFISs realmente é capaz de fazer com que o desempenho de controladores neuro-*fuzzy* seja melhorado se comparado ao daqueles puramente *fuzzy*, tanto com relação à qualidade de resposta quanto à eficiência energética.

## 4 Conclusão

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

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

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

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

No experimento computacional incluindo ruídos de medição, ambos os controladores se mostraram capazes de lidar com eles. Neste ponto, o controlador *neuro-fuzzy* apresentou desempenho bastante superior, levando a uma convergência mais rápida e com menor variação além de consumir menos energia.

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

Desta forma, mostra-se que se podem usar técnicas de IC para controlar, de forma eficiente, sistemas não lineares complexos e, além disso, que controladores neuro-*fuzzy* podem ser utilizados para melhorar o desempenho de controladores *fuzzy* apesar de, em algumas situações, piorar a resposta se comparado a estes.

## 4.1 Trabalhos futuros

Os resultados obtidos nesta dissertação abrem espaço para diferentes frentes de trabalho, tais como:

- Implementação dos controladores propostos sobre um sistema físico ;
- Investigação mais completa sobre a resposta dos controladores desenvolvidos quando atuando sobre sistemas sujeitos a diferentes tipos de ruído, tais como de medição e de atuação;
- Comparar os resultados obtidos pelos controladores desenvolvidos aos obtidos por outros controladores que implementam técnicas de IC e outros que seguem técnicas tradicionais de controle.

## 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');

```