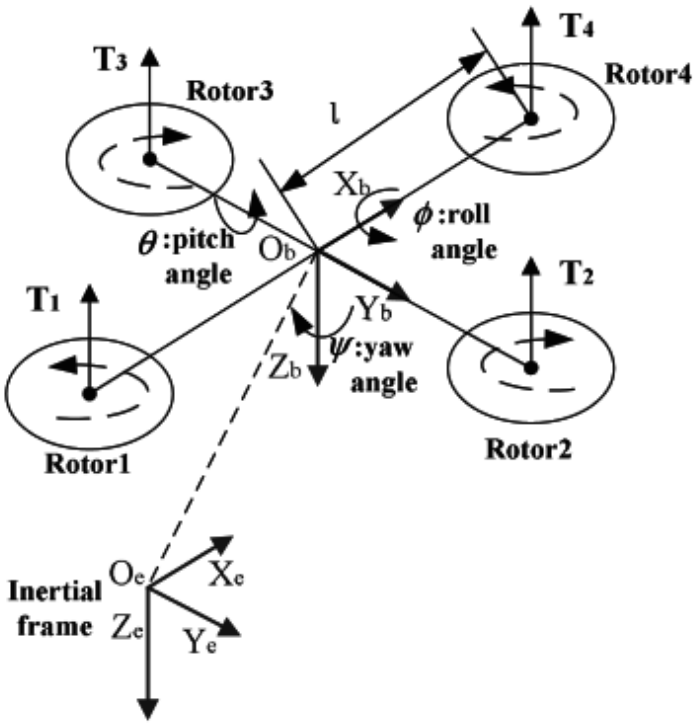
1 Modelagem de um Quadricóptero

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

Figura 1 – Representação de um quadricóptero



Fonte: Adaptado de Gao et al. (2014)

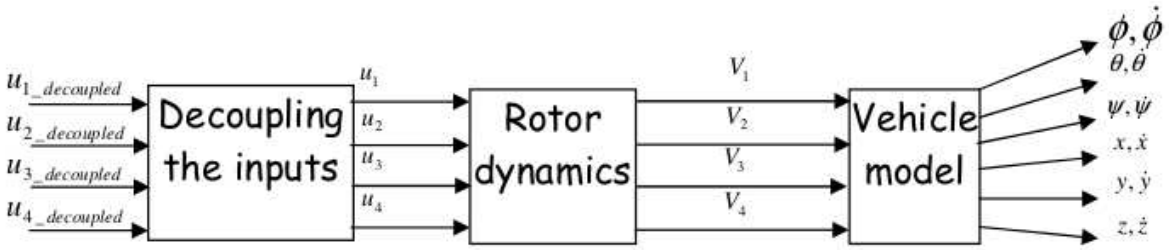
Neste diagrama, x, y e z indicam a orientação dos eixos, F_1, F_2, F_3 e F_4 são as forças geradas pela propulsão de cada um dos motores. As variáveis ϕ, θ, e ψ são os ângulos de rotação do quadricóptero em relação aos eixos x, y e z respectivamente. Estes ângulos são também chamados ângulos de *roll*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

A estabilidade de um 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 2

Figura 2 – Diagrama do sistema representando um quadrotor modelado por Balas (2007)



Fonte: Balas (2007, p. 49)

1.1 Modelagem Matemática

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

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

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

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

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

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

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

p. 5):

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

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

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

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

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

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

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

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

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, 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.

1.2 Desacoplamento das Entradas

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

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

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

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

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

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

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

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

1.2.1 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 matriz A, B, C e D sendo definida como:

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

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

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

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

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

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

Referências

BALAS, C. **Modeling and Linear Control of a Quadrotor**. Dissertação (Mestrado) — Cranfield University, Reino Unido, 2007. Citado 3 vezes nas páginas 2, 3 e 4.

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 na página 1.

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 1.

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');
```