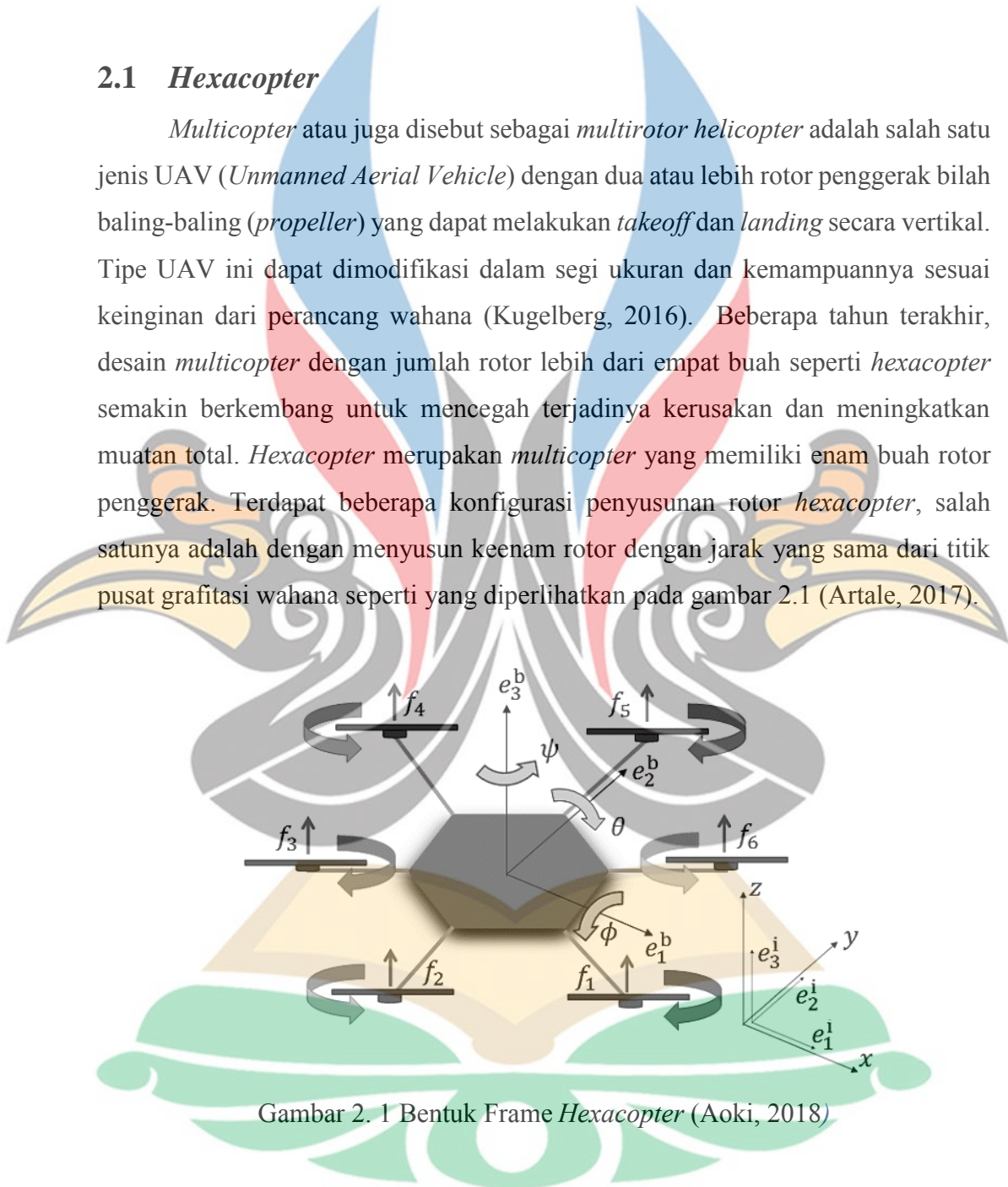


## BAB II

### TINJAUAN PUSTAKA

#### 2.1 *Hexacopter*

*Multicopter* atau juga disebut sebagai *multirotor helicopter* adalah salah satu jenis UAV (*Unmanned Aerial Vehicle*) dengan dua atau lebih rotor penggerak bilah baling-baling (*propeller*) yang dapat melakukan *takeoff* dan *landing* secara vertikal. Tipe UAV ini dapat dimodifikasi dalam segi ukuran dan kemampuannya sesuai keinginan dari perancang wahana (Kugelberg, 2016). Beberapa tahun terakhir, desain *multicopter* dengan jumlah rotor lebih dari empat buah seperti *hexacopter* semakin berkembang untuk mencegah terjadinya kerusakan dan meningkatkan muatan total. *Hexacopter* merupakan *multicopter* yang memiliki enam buah rotor penggerak. Terdapat beberapa konfigurasi penyusunan rotor *hexacopter*, salah satunya adalah dengan menyusun keenam rotor dengan jarak yang sama dari titik pusat grafitasi wahana seperti yang diperlihatkan pada gambar 2.1 (Artale, 2017).



Gambar 2. 1 Bentuk Frame *Hexacopter* (Aoki, 2018)

Gambar 2.1 menunjukkan sumbu inersia ( $e_j^b$ ) dan sumbu gerak ( $e_j^i$ ) ( $j = 1, 2, 3$ ) wahana di mana  $x, y, z$  merupakan posisi wahana,  $\Phi, \theta, \psi$  merupakan sudut euler, dan  $f_n$  ( $n = 1, \dots, 6$ ) merupakan gaya angkat yang dihasilkan tiap motor (Aoki, 2018)

### 2.1.1 Sudut Euler

Sudut euler adalah tiga sudut yang ditemukan oleh Leonhart Euler untuk mendeskripsikan orientasi dari benda yang kaku. Sudut euler biasanya disimbolkan dengan  $\Phi, \theta, \psi$ . Sudut euler merepresentasikan ketiga elemen rotasi terhadap titik sumbu sistem, di mana setiap orientasi dapat dicapai menggunakan ketiga elemen rotasi ini. Rotasi ini dimulai dari orientasi standar yang telah diketahui. Kombinasi orientasi ini dideskripsikan oleh matriks rotasi pada persamaan 2.1 –2.3.

$$R_x(\varphi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c(\Phi) & -s(\Phi) \\ 0 & s(\Phi) & c(\Phi) \end{bmatrix} \quad (2.1)$$

$$R_y(\theta) = \begin{bmatrix} c(\theta) & 0 & s(\theta) \\ 0 & 1 & 0 \\ -s(\theta) & 0 & c(\theta) \end{bmatrix} \quad (2.2)$$

$$R_z(\psi) = \begin{bmatrix} c(\psi) & -s(\psi) & 0 \\ s(\psi) & c(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2.3)$$

di mana  $R_{xyz}(\Phi, \theta, \psi)$  memiliki matriks seperti persamaan 2.4.

$$R = \begin{bmatrix} c(\theta)c(\psi) & s(\Phi)s(\theta)c(\psi) - c(\Phi)s(\psi) & c(\Phi)s(\theta)c(\psi) + s(\Phi)s(\psi) \\ c(\theta)s(\psi) & s(\Phi)s(\theta)s(\psi) + c(\Phi)c(\psi) & c(\Phi)s(\theta)s(\psi) - s(\Phi)c(\psi) \\ -s(\theta) & s(\Phi)c(\theta) & c(\Phi)c(\theta) \end{bmatrix} \quad (2.4)$$

(Francesco, 2015)

### 2.1.2 Model Matematis

Model matematis dari multicopter akan dibahas pada bagian ini dengan menggunakan persamaan Newton dan persamaan Euler untuk gerakan tiga dimensi pada benda yang kaku. Vektor  $[x \ y \ z \ \Phi \ \theta \ \psi]^T$  merupakan vektor yang berisi posisi linier dan posisi sudut dari wahana terhadap *earth frame*, dan vektor  $[u \ v \ w \ p \ q \ r]^T$  merupakan vektor yang berisi kecepatan linier dan kecepatan sudut terhadap *body frame*. Dari dinamika tiga dimensi wahana, kedua referensi *frame* dihubungkan oleh persamaan 2.5 dan persamaan 2.6.

$$V = R \cdot V_B \quad (2.5)$$

$$\omega = T \cdot \omega_B \quad (2.6)$$

di mana  $V = [\dot{x} \ \dot{y} \ \dot{z}]^T \in \mathbb{R}^3$ ,  $\omega = [\dot{\phi} \ \dot{\theta} \ \dot{\psi}]^T \in \mathbb{R}^3$ ,  $V_B = [u \ v \ w]^T \in \mathbb{R}^3$ ,  $\omega_B = [p \ q \ r]^T \in \mathbb{R}^3$ , dan  $\mathbf{T}$  adalah matriks transformasi untuk sudut angular seperti persamaan 2.7.

$$\mathbf{T} = \begin{bmatrix} 1 & s(\Phi)t(\theta) & c(\Phi)t(\theta) \\ 0 & c(\Phi) & -s(\Phi) \\ 0 & \frac{s(\Phi)}{c(\theta)} & \frac{c(\Phi)}{c(\theta)} \end{bmatrix} \quad (2.7)$$

Dengan  $t(\theta) = \tan(\theta)$ . Dengan menggunakan persamaan tersebut, maka model kinematic dari *hexacopter* ditunjukkan oleh persamaan 2.8.

$$\begin{cases} \dot{x} = w[s(\Phi)s(\psi) + c(\Phi)s(\theta)c(\psi)] - v[c(\Phi)s(\psi) - s(\Phi)s(\theta)c(\psi)] + u[c(\theta)c(\psi)] \\ \dot{y} = v[c(\Phi)c(\psi) + s(\Phi)s(\theta)s(\psi)] - w[s(\Phi)c(\psi) - c(\Phi)s(\theta)s(\psi)] + u[c(\theta)s(\psi)] \\ \dot{z} = w[c(\Phi)c(\theta)] + v[s(\Phi)c(\theta)] - u[s(\theta)] \\ \dot{\phi} = p + q[s(\Phi)t(\theta)] + r[c(\Phi)t(\theta)] \\ \dot{\theta} = q[c(\Phi)] - r[s(\Phi)] \\ \dot{\psi} = p \left[ \frac{s(\Phi)}{c(\theta)} \right] + r \left[ \frac{c(\Phi)}{c(\theta)} \right] \end{cases} \quad (2.8)$$

Hukum Newton menyatakan bahwa matriks berikut merupakan gaya total yang beraksi kepada multicopter

$$m(\omega_B \wedge V_B + \dot{V}_B) = f_B \quad (2.9)$$

di mana  $\wedge$  adalah perkalian *cross* dan  $f_B = [f_x \ f_y \ f_z]^T \in \mathbb{R}^3$  adalah gaya total. Persamaan euler memberikan total torsi yang diaplikasikan kepada *hexacopter* seperti persamaan 2.10.

$$\mathbf{I} \cdot \dot{\omega}_B + \omega_B \wedge (\mathbf{I} \cdot \omega_B) = m_B \quad (2.10)$$

di mana  $m_B = [m_x \ m_y \ m_z]^T \in \mathbb{R}^3$  adalah total torsi, dan  $\mathbf{I}$  adalah matriks inersia diagonal seperti yang ditunjukkan persamaan 2.11.

$$\mathbf{I} = \begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix} \in \mathbb{R}^3 \quad (2.11)$$

Sehingga model dinamis dari pada *frame hexacopter* adalah seperti yang ditunjukkan pada persamaan 2.12 (Francesco, 2015).

$$\begin{cases} f_x = m(\dot{u} + qw - rv) \\ f_y = m(\dot{v} + pw + ru) \\ f_z = m(\dot{w} + pv - qu) \\ m_x = \dot{p}I_x - qrI_y + qrI_z \\ m_y = \dot{q}I_y + prI_x - prI_z \\ m_z = \dot{r}I_z - pqI_x + pqI_y \end{cases} \quad (2.12)$$

### 2.1.3 Gaya dan Momen

Gaya eksternal terhadap frame wahana ditunjukkan dengan persamaan 2.13.

$$f_B = mg\mathbf{R}^T \cdot \hat{e}_z - f_t \hat{e}_3 + f_w \quad (2.13)$$

di mana  $\hat{e}_z$  adalah unit vektor inersia pada sumbu z,  $\hat{e}_3$  adalah unit vektor frame terhadap sumbu z, g adalah percepatan gravitasi bumi,  $f_t$  adalah gaya dorong total yang dihasilkan rotor, dan  $f_w = [f_{wx} \ f_{wy} \ f_{wz}]^T \in \mathbb{R}^3$  adalah gaya yang dihasilkan oleh angin wahana. Momen eksternal pada frame wahana dapat diperlihatkan pada persamaan 2.14.

$$m_B = \tau_B - G_a + \tau_w \quad (2.14)$$

di mana  $G_a$  merepresentasikan momen *gyroscopic* yang diakibatkan oleh rotasi gabungan dari keenam rotor dan frame wahana,  $\tau_B = [\tau_x \ \tau_y \ \tau_z]^T \in \mathbb{R}^3$  adalah torsi kontrol yang dihasilkan oleh kecepatan motor yang berbeda dan  $\tau_w = [\tau_{wx} \ \tau_{wy} \ \tau_{wz}]^T \in \mathbb{R}^3$  adalah torsi yang dihasilkan oleh angin dari wahana. Dikarenakan nilai  $G_a$  sangatlah kecil, maka momen *gyroscopic* dapat diabaikan. Untuk menyelesaikan model dinamis dari *frame body hexacopter*, persamaan tersebut didapatkan dengan mensubstitusikan persamaan 2.12 sehingga didapatkan persamaan model dinamis frame wahana hexacopter seperti yang ditunjukkan oleh persamaan 2.15.



$$\begin{cases} -mg[s(\theta)] + f_{wx} = m(\dot{u} + qw - rv) \\ mg[c(\theta)s(\phi)] + f_{wy} = m(\dot{v} - pw + ru) \\ mg[c(\theta)c(\phi)] + f_{wz} - f_t = m(\dot{w} + pv - qu) \\ \tau_x + \tau_{wx} = \dot{p}I_x - qrI_y + qrI_z \\ \tau_y + \tau_{wy} = \dot{q}I_y + prI_x - prI_z \\ \tau_z + \tau_{wz} = \dot{r}I_z - pqI_x + pqI_y \end{cases} \quad (2.15)$$

(Francesco, 2015)

### 2.1.4 Dinamika Aktuator

Gaya dorong yang dihasilkan oleh kecepatan motor dapat dituliskan dengan persamaan 2.16.

$$f_t = b \sum_{i=1}^6 \Omega_i^2 \quad (2.16)$$

Peningkatan kecepatan dari salah satu dari keenam motor akan menciptakan torsi terhadap sumbu  $x$ ,  $y$  dan  $z$  yang di mana akan menciptakan rotasi  $\varphi$ ,  $\theta$ ,  $\psi$ . Torsi merupakan gaya yang dikalikan dengan jarak dan motor-motor yang ada akan menghasilkan rotasi yang bergantung kepada pusat gravitasi wahana. Persamaan 2.17 – persamaan 2.19 adalah persamaan-persamaan torsi aktuator *hexacopter*.

$$\tau_x = bl \left( -\Omega_2^2 + \Omega_5^2 + \frac{1}{2}(-\Omega_1^2 - \Omega_3^2 + \Omega_4^2 + \Omega_6^2) \right) \quad (2.17)$$

$$\tau_y = bl \frac{\sqrt{3}}{2} (-\Omega_1^2 + \Omega_3^2 + \Omega_4^2 - \Omega_6^2) \quad (2.18)$$

$$\tau_z = d (-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2 - \Omega_5^2 + \Omega_6^2) \quad (2.19)$$

di mana  $\Omega$  merupakan kecepatan putar motor,  $b$  adalah konstanta gaya dorong motor,  $l$  adalah panjang lengan wahana, dan  $d$  adalah konstanta *drag* motor (Fogelberg, 2013).

### 2.1.5 Model State-Space

Vektor State wahana hexacopter dilambangkan pada persamaan 2.20.

$$\mathbf{x} = [\phi \ \theta \ \psi \ p \ q \ r \ u \ v \ w \ x \ y \ z]^T \in \mathbb{R}^{12} \quad (2.20)$$

Persamaan 2.8 dan 2.15 dapat dibuat menjadi fungsi *state-space* sehingga didapatkanlah persamaan 2.21.

$$\begin{cases}
 \dot{\Phi} = p + q[s(\Phi)t(\theta)] + r[c(\Phi)t(\theta)] \\
 \dot{\theta} = q[c(\Phi)] - r[s(\Phi)] \\
 \dot{\psi} = p \begin{bmatrix} s(\Phi) \\ c(\theta) \end{bmatrix} + r \begin{bmatrix} c(\Phi) \\ c(\theta) \end{bmatrix} \\
 \dot{p} = \frac{\tau_x + \tau_{wx}}{I_x} + qr \frac{I_y - I_z}{I_x} \\
 \dot{q} = \frac{\tau_y + \tau_{wy}}{I_y} + pr \frac{I_z - I_x}{I_y} \\
 \dot{r} = \frac{\tau_z + \tau_{wz}}{I_z} + pq \frac{I_x - I_y}{I_z} \\
 \dot{u} = rv - qw - g[s(\theta)] + \frac{f_{wx}}{m} \\
 \dot{v} = pw - ru + g[c(\theta)s(\Phi)] + \frac{f_{wy}}{m} \\
 \dot{w} = qu - pv + g[c(\theta)c(\Phi)] + \frac{f_{wz} - f_t}{m} \\
 \dot{x} = w[s(\Phi)s(\psi) + c(\Phi)s(\theta)c(\psi)] - v[c(\Phi)s(\psi) - s(\Phi)s(\theta)c(\psi)] + u[c(\theta)c(\psi)] \\
 \dot{y} = v[c(\Phi)c(\psi) + s(\Phi)s(\theta)s(\psi)] - w[s(\Phi)c(\psi) - c(\Phi)s(\theta)s(\psi)] + u[c(\theta)s(\psi)] \\
 \dot{z} = w[c(\Phi)c(\theta)] + v[s(\Phi)c(\theta)] - u[s(\theta)]
 \end{cases} \quad (2.21)$$

Di bawah ini kita mendapatkan dua bentuk alternatif dari model dinamis yang berguna untuk mempelajari pengontrolannya. Dengan menggunakan hukum Newton dapat dituliskan persamaan 2.22.

$$m\dot{V} = R \cdot f_B = mg\hat{e}_z - f_t R \cdot \hat{e}_3 \quad (2.22)$$

Sehingga didapatkan persamaan yang ditunjuka oleh persaman 2.23.

$$\begin{cases}
 \ddot{x} = -\frac{f_t}{m}[s(\Phi)s(\psi) + c(\Phi)c(\psi)s(\theta)] \\
 \ddot{y} = -\frac{f_t}{m}[c(\Phi)s(\psi)s(\theta) - s(\Phi)c(\psi)] \\
 \ddot{z} = g - \frac{f_t}{m}[c(\Phi)c(\theta)]
 \end{cases} \quad (2.23)$$

Dengan demikian, maka penyederhanaan dapat dilakukan dengan membuat  $[\dot{\Phi} \ \dot{\theta} \ \dot{\psi}]^T = [p \ q \ r]^T$ . Asumsi ini dikatakan benar untuk sudut pergerakan kecil. Maka model dinamis *hexacopter* pada *frame* inersianya ditunjukkan pada persamaan 2.24.

$$\begin{cases} \ddot{x} = -\frac{f_t}{m} [s(\phi)s(\psi) + c(\phi)c(\psi)s(\theta)] \\ \ddot{y} = -\frac{f_t}{m} [c(\phi)s(\psi)s(\theta) - s(\phi)c(\psi)] \\ \ddot{z} = g - \frac{f_t}{m} [c(\phi)c(\theta)] \\ \ddot{\phi} = \frac{I_y - I_z}{I_x} \dot{\theta}\dot{\psi} + \frac{\tau_x}{I_x} \\ \ddot{\theta} = \frac{I_z - I_x}{I_y} \dot{\phi}\dot{\psi} + \frac{\tau_y}{I_y} \\ \ddot{\psi} = \frac{I_x - I_y}{I_z} \dot{\phi}\dot{\theta} + \frac{\tau_z}{I_z} \end{cases} \quad (2.24)$$

Mendefinisikan ulang vektor *state-space* sebagai

$$x = [x \ y \ z \ \phi \ \theta \ \psi \ \dot{x} \ \dot{y} \ \dot{z} \ p \ q \ r]^T \in \mathbb{R}^{12} \quad (2.25)$$

Sehingga dimungkinkan menuliskan persamaan dari *state-space* dari *hexacopter* seperti yang ditunjukkan persamaan 2.26.

$$\dot{x} = f(x) + \sum_{i=1}^4 G_i(x)u_i \quad (2.26)$$

di mana

$$f(x) = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ q \frac{s(\phi)}{c(\theta)} + r \frac{c(\phi)}{c(\theta)} \\ q[c(\phi)] - r[s(\phi)] \\ p + q[s(\phi)t(\theta)] + r[c(\phi)t(\theta)] \\ 0 \\ 0 \\ g \\ \frac{I_y - I_z}{I_x} qr \\ \frac{I_z - I_x}{I_y} pr \\ \frac{I_x - I_y}{I_z} pq \end{bmatrix} \quad (2.27)$$

Dan

$$G_1(x) = [0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ g_1^7 \ g_1^8 \ g_1^9 \ 0 \ 0 \ 0]^T \in \mathbb{R}^{12} \quad (2.28)$$

$$G_2(x) = \left[ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ \frac{1}{I_x} \ 0 \ 0 \right]^T \in \mathbb{R}^{12} \quad (2.29)$$

$$G_3(x) = \left[ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ \frac{1}{I_y} \ 0 \right]^T \in \mathbb{R}^{12} \quad (2.30)$$

$$G_4(x) = \left[ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ \frac{1}{I_z} \right]^T \in \mathbb{R}^{12} \quad (2.31)$$

Dengan

$$g_1^7 = -\frac{1}{m} [s(\phi)s(\psi) + c(\phi)s(\theta)c(\psi)] \quad (2.32)$$

$$g_1^8 = -\frac{1}{m} [s(\phi)c(\psi) - c(\phi)s(\theta)s(\psi)] \quad (2.33)$$

$$g_1^9 = -\frac{1}{m} [c(\phi)c(\theta)] \quad (2.34)$$

(Francesco, 2015)

### 2.1.6 Model Linear

Vektor  $u$  merupakan vektor kontrol di mana  $u = [f_t \ \tau_x \ \tau_y \ \tau_z]^T \in \mathbb{R}^4$ . Proses linearisasi dikembangkan pada titik equilibrium  $\bar{x}$ , yang digunakan untuk memperbaiki input  $\bar{u}$  sebagai solusi sistem aljabar. Nilai dari vektor *state-space* yang digunakan untuk memperbaiki input konstan ditunjukkan oleh persamaan 2.35.

$$\hat{f}(\bar{x}, \bar{u}) = 0 \quad (2.35)$$

Dikarenakan fungsi  $\hat{f}$  tidak linear, permasalahan berkaitan dengan adanya keunikan solusi dari sistem muncul. Khususnya untuk sistem yang ada di mana solusi sulit ditemukan pada sistem tertutup dikarenakan fungsi trigonometri yang berhubungan satu dengan lain secara tidak mendasar. Oleh karena itu linearisasi dilakukan kepada model yang disederhanakan dengan osilasi yang kecil. Penyederhanaan ini didapatkan dengan memperkirakan fungsi sinus memiliki nilai 0 yang ada dan fungsi cosinus memiliki nilai uniti. Perkiraan bersifat valid jika nilai sudut kecil. Sistem yang didapatkan dideskripsikan dengan persamaan 2.36 seperti berikut.



$$\begin{cases}
 \dot{\phi} \approx p + q\phi\theta + r\theta \\
 \dot{\theta} \approx q - r\phi \\
 \dot{\psi} \approx p\phi + r \\
 \dot{p} \approx \frac{\tau_x + \tau_{wx}}{I_x} + qr \frac{I_y - I_z}{I_x} \\
 \dot{q} \approx \frac{\tau_y + \tau_{wy}}{I_y} + pr \frac{I_z - I_x}{I_y} \\
 \dot{r} \approx \frac{\tau_z + \tau_{wz}}{I_z} + pq \frac{I_x - I_y}{I_z} \\
 \dot{u} \approx rv - qw - g\theta + \frac{f_{wx}}{m} \\
 \dot{v} \approx pw - ru + g\phi + \frac{f_{wy}}{m} \\
 \dot{w} \approx qu - pv + g + \frac{f_{wz} - f_t}{m} \\
 \dot{x} \approx w(\phi + \theta) - v(\psi - \phi\psi) + u \\
 \dot{y} \approx v(1 + \phi\theta\psi) - w(\phi - \theta\psi) + u\psi \\
 \dot{z} \approx w + v\phi - u\theta
 \end{cases} \quad (2.36)$$

Model ini juga dapat ditulis dengan persamaan 2.37.

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \quad (2.37)$$

(Francesco, 2015)

### 2.1.7 Linearisasi

Seperti yang dijelaskan sebelumnya, untuk melakukan linearisasi titik *equilibrium* dibutuhkan. Titik tersebut ditunjukkan oleh persamaan 2,38

$$\bar{\mathbf{x}} = [0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ \bar{x} \ \bar{y} \ \bar{z}]^T \in \mathbb{R}^{12} \quad (2.38)$$

Dari persamaan tersebut kita dapat mencari titik *equilibrium* pada persamaan di atas dengan nilai input konstan seperti berikut

$$\bar{\mathbf{u}} = [mg \ 0 \ 0 \ 0]^T \in \mathbb{R}^4 \quad (2.39)$$

Perlu diingat bahwa nilai khusus ini merepresentasikan gaya yang dibutuhkan untuk menghilangkan berat dari wahana dan membuatnya *hovering*. Setelah ditentukan titik *equilibrium*  $\bar{\mathbf{x}}$  dan input  $\bar{\mathbf{u}}$  yang bersangkutan, kita mendapatkan matriks yang berhubungan dengan sistem linear yang ditunjukkan oleh persamaan 2.40 dan 3.41.

$$\mathbf{A} = \frac{\partial f(\mathbf{x}, \mathbf{u})}{\partial \mathbf{x}} \bigg|_{\substack{\mathbf{x} = \bar{\mathbf{x}} \\ \mathbf{u} = \bar{\mathbf{u}}}} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -g & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ g & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \quad (2.40)$$

$$\mathbf{B} = \frac{\partial f(\mathbf{x}, \mathbf{u})}{\partial \mathbf{u}} \bigg|_{\substack{\mathbf{x} = \bar{\mathbf{x}} \\ \mathbf{u} = \bar{\mathbf{u}}}} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1/I_x & 0 & 0 \\ 0 & 0 & 1/I_y & 0 \\ 0 & 0 & 0 & 1/I_z \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1/m & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (2.41)$$

Jika gangguan angin dianggap maka digunakan persamaan 2.42 dan 2.43 berikut

$$\mathbf{d} = [f_{wx} \ f_{wy} \ f_{wz} \ \tau_{wx} \ \tau_{wy} \ \tau_{wz}]^T \in \mathbb{R}^6 \quad (2.42)$$

$$\mathbf{D} = \frac{\partial f(\mathbf{x}, \mathbf{u})}{\partial \mathbf{x}} \bigg|_{\substack{\mathbf{x} = \bar{\mathbf{x}} \\ \mathbf{u} = \bar{\mathbf{u}}}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/I_x & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/I_y & 0 \\ 0 & 0 & 0 & 0 & 0 & 1/I_z \\ 1/m & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/m & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/m & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (2.43)$$

Sehingga model linear sistem dapat dilihat pada persamaan 4.44 dan 4.45

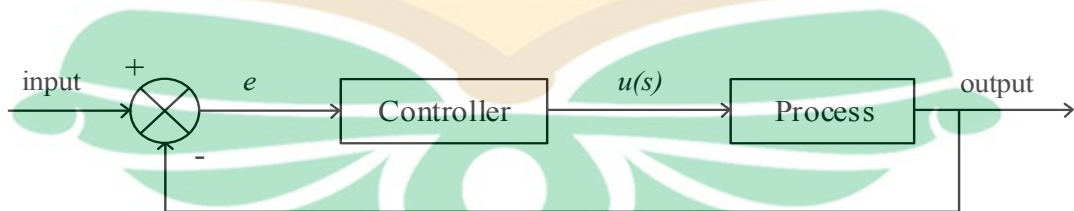
$$\dot{\mathbf{x}} = \mathbf{A} \cdot \mathbf{x} + \mathbf{B} \cdot \mathbf{u} + \mathbf{D} \cdot \mathbf{d} \quad (2.44)$$

$$\begin{cases}
 \dot{\phi} = p \\
 \dot{\theta} = q \\
 \dot{\psi} = r \\
 \dot{p} = \frac{\tau_x + \tau_{wx}}{I_x} \\
 \dot{q} = \frac{\tau_y + \tau_{wy}}{I_y} \\
 \dot{r} = \frac{\tau_z + \tau_{wz}}{I_z} \\
 \dot{u} = -g\theta + \frac{f_{wx}}{m} \\
 \dot{v} = g\phi + \frac{f_{wy}}{m} \\
 \dot{w} = \frac{f_{wz} - f_t}{m} \\
 \dot{x} = u \\
 \dot{y} = v \\
 \dot{z} = w
 \end{cases} \tag{2.45}$$

(Francesco, 2015)

## 2.2 *Propotional Integral Derivative (PID) Controller*

PID *controller* adalah pengendali yang paling banyak diaplikasikan, bahkan sistem kontrol pada industri yang kompleks mungkin menggunakan jaringan kontrol dengan pondasi dasarnya berupa modul PID (Jhonson, 2005). Pada dasarnya kontrol PID adalah mekanisme *loop* tertutup. Kontrol PID memiliki prinsip kerja untuk memonitor *error* antara variabel proses yang terjadi dengan *set point* yang diinginkan. Dari nilai *error* ( $e$ ) yang didapatkan maka sinyal untuk memperbaiki sistem dikalkulasikan lalu diberikan sebagai input untuk menyesuaikan proses yang diinginkan (Deepyaman, 2008).



Gambar 2. 2 Sistem *Loop* Tertutup (Deepyaman, 2008)

Kontrol PID juga disebut sebagai *three-term control* yang berarti kontrol PID merupakan gabungan dari tiga buah kontrol, yaitu *Propotional Control*,

*Integral Control*, dan *Derivative Control*. *Proportional Control* yang disimbolkan dengan P pada kontrol PID dengan *gain* yang disimbolkan dengan  $K_p$ , digunakan pada saat aksi kontrol berbanding lurus dengan sinyal *error* proses ( $e(t)$ ). Dalam bentuk domain waktu, *proportional control* direpresentasikan pada persamaan 2.46.

$$u_c(t) = K_p e(t) \quad (2.46)$$

*Integral Control* yang disimbolkan dengan I dengan nilai *gain* disimbolkan dengan  $K_i$ , digunakan pada saat dibutuhkannya kontrol yang memperbaiki *steady offset* (*Steady-state error*) dengan nilai sinyal referensi konstan. Dalam bentuk domain waktu, *integral control* direpresentasikan pada persamaan 2.47.

$$u_c(t) = K_i \int^t e(\tau) d\tau \quad (2.47)$$

Dan yang terakhir adalah *derivative control* yang disimbolkan dengan D dengan *gain* yang disimbolkan dengan  $K_d$ . Kontrol ini menggunakan tingkat perubahan dari sinyal *error* sebagai input dan memprediksi perilaku sistem dan memperbaikinya. *Derivative control* direpresentasikan pada persamaan 2.48.

$$u_c(t) = K_d \frac{de}{dt} \quad (2.48)$$

Gabungan dari ketiga kontrol ini disebut sebagai *PID controller* yang direpresentasikan pada persamaan 2.49.

$$u_c(t) = K_p e(t) + K_i \int^t e(\tau) d\tau + K_d \frac{de}{dt} \quad (2.49)$$

Jika persamaan 2.48 direpresentasikan menjadi bentuk domain laplace, maka persamaan 2.50 akan digunakan.

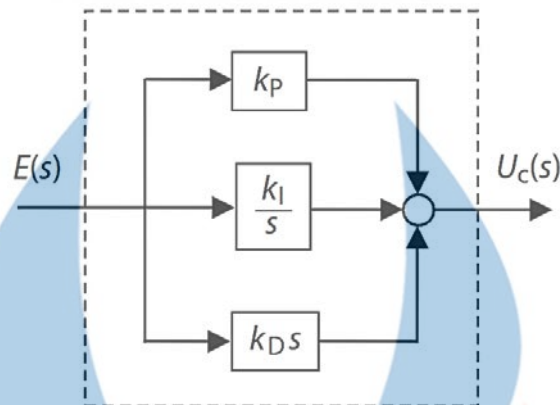
$$U(s) = \left[ K_p + K_i \frac{1}{s} + K_d s \right] E(s) \quad (2.50)$$

Persamaan 2.50 juga sering disebut sebagai *PID controller* dasar dikarenakan persamaan tersebut tidak menggunakan modifikasi tambahan yang biasa digunakan untuk mendapatkan kontrol PID yang dapat digunakan (Jhonson, 2005). Berikut



adalah gambar kontrol PID pada sistem tertutup

www.itk.ac.id



Gambar 2. 3 Desain PID Controller (Jhonson, 2005)

Efek dari setiap kontrol pada sistem tertutup dapat dilihat pada Tabel 2.1

Tabel 2. 1 Efek Kontrol PID

Efek Terhadap Performa				
	<i>Rise Time</i>	<i>Settling Time</i>	<i>Overshoot</i>	<i>Error Steady State</i>
P	Mengurangi	Perubahan Kecil	Meningkatkan	Mengurangi
I	Mengurangi	Meningkatkan	Meningkatkan	Menghilangi
D	Perubahan Kecil	Mengurangi	Mengurangi	Perubahan Kecil

\*)Hussein, 2015

### 2.3 Emperor Penguin Optimizer

Penguin kaisar (*aptenodytes forsteri*) merupakan spesies penguin yang tertinggi dan terberat. Penguin ini memiliki kepala, sirip/lengan, dan bagian punggung yang berwarna hitam, bagian perut berwarna putih, dada yang berwarna kuning pucat, serta warna kuning disekeliling telinganya. Penguin kaisar hidup di ekosistem tundra yang memiliki suhu yang sangat dingin. Untuk mempertahankan suhu tubuh yang hangat maka penguin kaisar akan membentuk sebuah kerumunan. Hal unik dalam pembentukan kerumunan ini adalah bahwa setiap penguin memiliki kesempatan yang sama untuk menghangatkan diri. Perilaku penguin kaisar ini yang dibuat menjadi persamaan matematis (Dhiman, 2018).

Penguin kaisar membentuk sebuah kerumunan untuk memaksimalkan suhu ambien pada kerumunan. Untuk memodelkan kondisi ini diasumsikan suhu ( $T$ ) bernilai 0 saat radius penguin ( $R$ ) lebih dari 1 dan suhu bernilai 1 saat radius lebih

kecil dari 1. Profil suhu ini mempengaruhi tingkat eksplorasi dari penguin kaisar. Persamaan 2.51 menunjukkan persamaan untuk menentukan profil suhu pada kerumunan ( $T'$ ). Variabel  $it$  merupakan nilai iterasi yang sedang dilakukan.

$$T' = \left(T - \frac{Maxiteration}{it - Maxiteration}\right) \quad (2.51)$$

Dengan nilai  $T$  adalah sebesar

$$T = \begin{cases} 0 & \text{jika } R > 1 \\ 1 & \text{jika } R < 1 \end{cases} \quad (2.52)$$

Profil suhu ( $T'$ ) digunakan untuk mencari nilai koefisien pencegah tabrakan antar penguin ( $\vec{A}$ ) seperti yang ditunjukkan oleh persamaan 2.53. Koefisien  $M$  digunakan untuk membuat jarak antar penguin agar tidak terjadi tabrakan yang diberi nilai 2.  $P_{grid}(accuracy)$  menyatakan akurasi grid pada polygon kerumunan dan  $rand$  adalah nilai acak yang memiliki nilai  $[0, 1]$ .

$$\vec{A} = \left(M \times \left(T' + P_{grid}(accuracy)\right) \times rand()\right) - T' \quad (2.53)$$

Dengan persamaan akurasi grid

$$P_{grid}(accuracy) = abs(\vec{P} - \vec{P}_{ep}) \quad (2.54)$$

Persamaan 2.54,  $\vec{P}$  menyatakan penguin yang memiliki posisi yang paling optimal dengan nilai fitness yang paling baik dan  $\vec{P}_{ep}$  menyatakan posisi penguin yang lain (Dhiman, 2018).

Pergerakan penguin kaisar yang cenderung mendekati peinguin yang memiliki suhu yang lebih tinggi ( $fitness$  terbaik) dimodelkan sebagai fungsi vektor  $S()$ . Nilai dari vektor  $S()$  didapatkan dengan menggunakan persamaan 2.55. Konstanta  $e$  merupakan fungsi eksponensial dengan variabel  $f$  dan  $l$  merupakan parameter kontrol yang masing masing memiliki nilai diantara  $[2, 3]$  dan  $[1.5, 2]$ .

$$S(\vec{A}) = \left(\sqrt{f \cdot e^{-\frac{t}{l}} - e^{-t}}\right)^2 \quad (2.55)$$

Vektor  $S()$  digunakan untuk mendapatkan nilai jarak antara penguin yang ada dengan penguin terbaik ( $\vec{D}_{ep}$ ) yang didapatkan dengan menggunakan persamaan

2.56. Vektor  $\vec{C}$  merupakan konstanta pencegah tabrakan antar penguin seperti vektor  $\vec{A}$  yang memiliki nilai acak diantara [0, 1].

$$\vec{D}_{ep} = abs(S(\vec{A}) \cdot \vec{P}(x) - \vec{C} \cdot \vec{P}_{ep}(x)) \quad (2.56)$$

Setelah dilakukan perhitungan pergerakan penguin, maka dilakukan penentuan posisi penguin terbaru setelah dilakukannya iterasi dengan persamaan 2.57.

$$\vec{P}_{ep}(x+1) = \vec{P}(x) - \vec{A} \cdot \vec{D}_{ep} \quad (2.57)$$

EPO pada dasarnya merupakan algoritma *particle swarm optimization* dengan mengaplikasikan perilaku penguin kaisar. Setiap populasi yang dibuat bersama-sama mencari solusi yang optimal pada suatu permasalahan. Algoritma EPO memiliki *pseudocode* yang dapat dilihat pada Tabel 2.2 (Dhiman, 2018).

Tabel 2. 2 Pseudocode Algoritma EPO

- 
1. START
  2. Inialisasi populasi penguin
  3. Inialisasi Batas atas dan batas bawah semesta dan posisi setiap penguin  $\vec{P}_{ep}(x)$
  4. Inialisasi parameter  $T'$ ,  $R$ ,  $\vec{A}$ ,  $\vec{C}$ ,  $S()$ ,  $Max_{iteration}$
  5. while (setiap  $\vec{P}_{ep}(x)$ )
  6. Hitung *fitness* setiap penguin  $\vec{P}_{ep}$
  7. Perbarui *fitness function* sesuai dengan posisi penguin terbaik
  8. For  $it < Max_{iteration}$
  9.  $R = rand [0, 1]$ 
    - If ( $R > 1$ )
      - $T = 0$
    - Else
      - $T = 1$
  - End if
  10. Hitung  $T'$  dengan persamaan (2.51)
  11. Hitung vektor  $\vec{A}$  dan  $\vec{C}$  dengan persamaan (2.53) dan nilai acak [0, 1]
  12. Hitung  $\vec{D}_{ep}$  dengan persamaan (2.56)
  13. Hitung posisi terbaru penguin dengan persamaan (2.57)
  14. Memperbarui posisi penguin sesuai dengan batas semesta yang diberikan
  15. Perbarui *fitness function* sesuai dengan posisi penguin terbaik
  16.  $it = it + 1$
  17. end for
  18. end while
  19. Tampilkan nilai  $\vec{P}$  terbaik
  20. END
- 

\*)Dhiman, 2018

## 2.4 Particle Swarm Optimization

*Particle swarm optimization* (PSO) adalah algoritma berbasis populasi yang dikembangkan oleh Eberhart dan Kennedy berdasarkan perilaku dari sekelompok burung dan ikan pada tahun 1995. Untuk mencari solusi yang optimal, setiap partikel bergerak menuju arah posisi terbaik iterasi sebelumnya dan juga posisi terbaik dari semua iterasi (*global best*). PSO menggunakan variabel berupa kecepatan ( $V$ ) dan posisi ( $P$ ) yang diperbarui pada setiap iterasi. PSO memperbarui posisinya dengan menggunakan persamaan 2.58 dan 2.59 (Zhang, 2015).

$$V_i(t+1) = \chi \left( V_i(t) + c_1 r_1 (pbest(i,t) - P_i(t)) + c_2 r_2 (gbest(i,t) - P_i(t)) \right) \quad (2.58)$$

$$P_i(t+1) = P_i(t) + V_i(t+1) \quad (2.59)$$

Dengan nilai parameter penyempitan yang dapat dilihat pada persamaan 2.60.

$$\chi = \frac{2}{\left( 2 - (\varphi) - \sqrt{\varphi^2 - 4\varphi} \right)} \quad (2.60)$$

Di mana  $c_1$  dan  $c_2$  adalah koefisien percepatan dan  $r_1$  dan  $r_2$  adalah variabel random dengan nilai  $[0, 1]$  dan  $\varphi = c_1 + c_2$ . Algoritma PSO memiliki *pseudocode* seperti yang dapat dilihat pada Tabel 2.3 (Connor, 2017).

Tabel 2.3 *Pseudocode* Algoritma PSO

- 
1. START
  2. Inialisasi populasi partikel
  3. Inialisasi batas atas dan batas bawah semesta dan posisi setiap partikel  $P_i(t)$
  4. Inialisasi parameter  $\omega, c_1, c_2, V_i$
  5. while (setiap  $P_i(t)$ )
  6. Hitung *fitness* setiap partikel  $P_i(t)$
  7. Perbarui *fitness function* sesuai dengan posisi partikel terbaik
  8. For  $i < Max_{iteration}$
  9. Pilih nilai acak  $[0, 1]$  untuk nilai  $r_1$  dan  $r_2$
  10. Hitung  $V_i(t+1)$  dengan persamaan (2.58)
  11. Hitung posisi partikel terbaru  $P_i(t+1)$  dengan persamaan (2.59)
  12. Memperbaiki posisi partikel sesuai dengan batas semesta yang diberikan
  13. Perbarui *fitness function* sesuai dengan posisi penguin terbaik
  14.  $i = i+1$
  15. end for
  16. end while
  17. Tampilkan nilai  $P_i$  terbaik
  18. END
- 

\*) Zhang, 2015



## 2.5 Integral Time Absolute Error (ITAE)

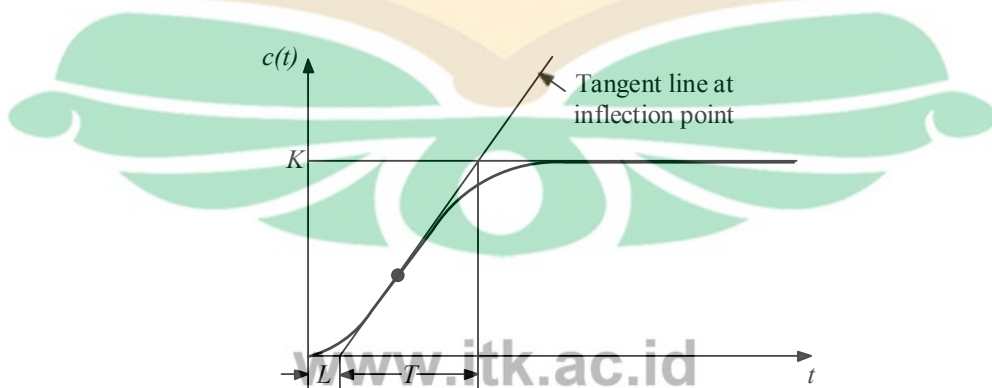
Memastikan stabilitas suatu plant merupakan suatu hal yang penting dalam perancangan suatu sistem kendali. Sebuah sistem tertutup dirancang agar output yang diinginkan dapat dipertahankan, walaupun sistem mengalami gangguan atau mengalami perubahan *setpoint*. Kontrol PID merupakan kontrol yang paling umum digunakan di dunia industri. Untuk mendapatkan respon sistem yang diinginkan, parameter kontrol PID hanya dapat diasumsikan baik secara perhitungan maupun secara *trial and error* (Shuaib, 2007). Untuk melihat baik atau tidaknya parameter kontrol digunakan, kriteria pembandingan ITAE digunakan seperti persamaan 2.61.

$$ITAE = \int_0^T t|e(t)|dt \quad (2.61)$$

ITAE memiliki keunggulan untuk menghasilkan *overshoot* dan osilasi yang lebih kecil dibandingkan dengan *Integral of Absolute Error* (IAE) dan *Integral Square Error* (ISE). Selain itu ITAE juga lebih sensitif dibandingkan dengan *Integral Time-square Error* (ITSE) (Maiti, 2008).

## 2.6 Metode Ziegler-Nichols

Ziegler dan Nichols mengusulkan aturan untuk menentukan nilai dari *gain propotional* ( $K_p$ ), waktu *integral* ( $T_i$ ), dan waktu *derivative* ( $T_d$ ) berdasarkan karakteristik respon transien dari *plant* yang diberikan. Terdapat dua metode dengan mengikuti aturan *tuning* Ziegler-Nichols. Pada metode pertama dilakukan dengan memberikan *input step* kepada sistem *open loop* sehingga output berbentuk seperti kurva berbentuk S seperti yang ditampilkan pada Gambar 2.4. Untuk menentukan nilai  $L$  dan  $T$  ditunjukkan pada Gambar 2.4.



Gambar 2. 4 Penentuan Nilai  $L$  dan  $T$  pada Kurva berbentuk S (Ogata, 2010)

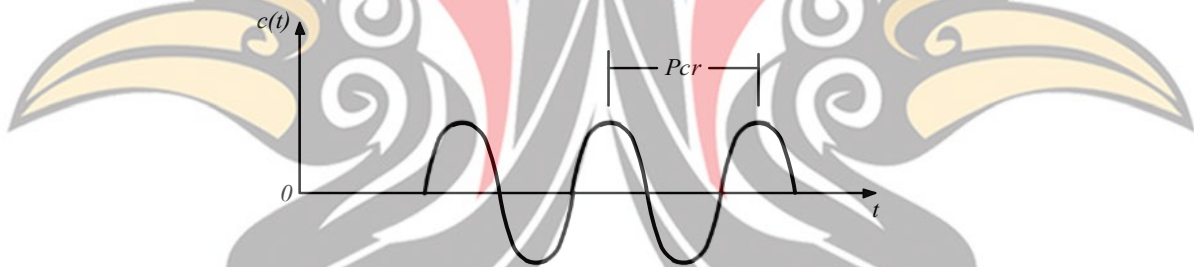
Kurva beerbentuk S ini dapat merepresentasikan dua konstanta, yaitu waktu tunda  $L$  dan waktu konstan  $T$ . Setelah parameter  $L$  dan  $T$  didapatkan maka nilai  $K_p$ ,  $T_i$  dan  $T_d$  dapat dicari dengan menggunakan persamaan pada Tabel 2.4 (Ogata, 2010).

Tabel 2. 4 Parameter Metode Ziegler-Nichols Pertama

Type Kontrol	$K_p$	$T_i$	$T_d$
P	$\frac{T}{L}$	$\infty$	0
PI	$0.9 \frac{T}{L}$	$\frac{L}{0.3}$	0
PID	$1.2 \frac{T}{L}$	$2L$	$0.5L$

\*)Ogata, 2010

Metode yang kedua dilakukan dengan menentukan nilai  $T_i = \infty$  dan  $T_d = 0$  dengan menggunakan nilai *gain proposional yang critical* ( $K_{cr}$ ) dan nilai periode osilasi *critical* ( $P_{cr}$ ) seperti yang ditampilkan pada Gambar 2.5.



Gambar 2. 5 Osilasi Sistem yang Berkelanjutan dengan Periode  $P_{cr}$  (Ogata, 2010)

Nilai  $K_{cr}$  dapat dicari dengan menggunakan kriteria kestabilan routh. Persamaan 2.62 adalah persamaan sistem orde tiga.

$$a_0s^3 + a_1s^2 + a_2s + a_3 = 0 \quad (2.62)$$

Dengan mengaplikasikan routh kepada persamaan 2.62 maka diperoleh persamaan 2.63.

$$\begin{array}{r}
 s^3 \quad \quad \quad a_0 \quad \quad \quad a_2 \\
 s^2 \quad \quad \quad a_1 \quad \quad \quad a_3 \\
 s^1 \quad \quad \quad \frac{a_1a_2 - a_0a_3}{a_1} \\
 s^0 \quad \quad \quad a_3
 \end{array} \quad (2.63)$$

di mana nilai  $K_p$  akan dimasukkan ke dalam persamaan 2.62 sehingga diperoleh nilai  $K_{cr}$ . Setelah diperoleh nilai  $K_{cr}$  dan  $P_{cr}$  maka Tabel 2.4 dapat digunakan untuk mendapatkan nilai  $K_p$ ,  $T_i$  dan  $T_d$  (Ogata, 2010)

Tabel 2. 5 Parameter Metode Ziegler-Nichols Kedua

Tipe Kontrol	$K_p$	$T_i$	$T_d$
P	$0.5K_{cr}$	$\infty$	0
PI	$0.45K_{cr}$	$\frac{1}{1.2}P_{cr}$	0
PID	$0.6K_{cr}$	$0.5P_{cr}$	$0.125P_{cr}$

\*)Ogata, 2010

## 2.7 Momen Inersia

Momen inersia adalah ukuran kelembaman suatu benda berotasi terhadap porisnya dengan satuan  $kg.m^2$ . Momen inersia berperan dalam dinamika rotasi yang menentukan hubungan momentum sudut dan kecepatan sudut, momen gaya dan percepatan sudut. Jika benda kaku terdiri dari beberapa partikel, kita dapat menghitung momen inersia terhadap poros putarnya dengan persamaan 2.64.

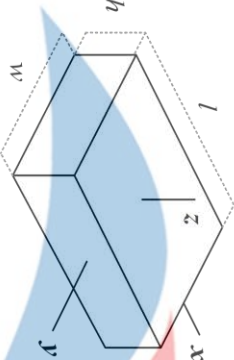
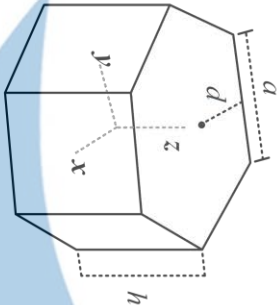
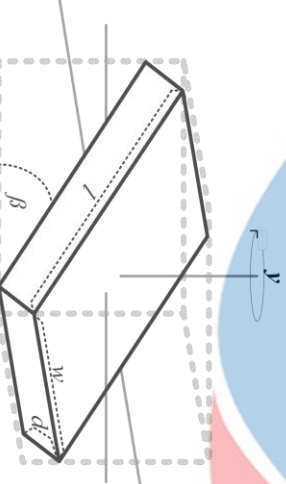
$$I = \sum m_i r_i^2 \quad (2.64)$$

Di mana  $m_i$  adalah massa partikel  $i$  dengan satuan  $kg$  dan  $r_i$  adalah jarak partikel  $i$  ke titik poros dengan satuan  $meter$ . Jika suatu benda kaku memiliki partikel yang sangat banyak, persamaan 2.64 tidak disarankan untuk digunakan. Untuk mengatasi permasalahan tersebut, maka digunakanlah persamaan 2.65.

$$I = \int r^2 dm \quad (2.65)$$

Seperti yang diketahui setiap benda memiliki nilai momen inersia yang berbeda-beda. Sesuai dengan persamaan 2.65 faktor utama yang mempengaruhi momen inersia suatu benda adalah persebaran massa dari suatu benda dan juga jarak benda tersebut terhadap poros yang ada. Oleh karena itu bentuk dari suatu benda juga mempengaruhi momen inersia benda tersebut (Walker, 2014). Tabel 2.6 menunjukkan persamaan inersia dari beberapa jenis bentuk benda dari poros putar yang dijelaskan.

Tabel 2. 6 Momen Inersia Beberapa Bentuk Benda

No.	Bentuk Benda	Bentuk	Persamaan	Keterangan
1		Balok dengan poros di pusat balok	$I_x = \frac{1}{12} M(w^2 + h^2)$ $I_y = \frac{1}{12} M(l^2 + h^2)$ $I_z = \frac{1}{12} M(l^2 + w^2)$	<p><math>M</math> = Massa</p> <p><math>w</math> = Lebar Balok</p> <p><math>l</math> = Panjang Balok</p> <p><math>h</math> = Tinggi Balok</p>
2		Perisma segi enam dengan poros di pusat balok	$I_x = I_y = M \left( \frac{a^2}{6} + \frac{d^2}{3} + \frac{h^2}{12} \right)$ $I_z = M \left( \frac{a^2}{6} + \frac{d^2}{3} \right)$	<p><math>M</math> = Massa</p> <p><math>a</math> = Sisi Segi Enam</p> <p><math>d = \frac{1}{2}</math> Tinggi Segi Enam</p> <p><math>h</math> = Tinggi Perisma</p>
3		balok miring dengan poros di sumbu y	$I = \frac{1}{12} M(l^2 \cos^2 \beta + d^2 \sin^2 \beta + w^2)$	<p><math>M</math> = Massa</p> <p><math>\beta</math> = Sudut kemiringan</p> <p><math>d</math> = Lebar Balok</p> <p><math>l</math> = Panjang Balok</p> <p><math>h</math> = Tinggi Balok</p>

\*)Bresciani, 2008; Panagapulos, 2015; Zanneiti, 2018



Jika kita ingin mengetahui nilai rotasi inersia dari suatu benda bermassa  $M$  terhadap sumbu tertentu, kita dapat menggunakan persamaan 2.64. Akan tetapi ada jalan pintas jika telah diketahui momen inersia  $I_m$  dari suatu benda yang sumbunya parallel terhadap pusat massa benda. Jarak dari poros putar dengan pusat massa benda adalah  $h$  ( $m$ ) sehingga didapatkanlah persamaan

$$I = I_m + Mh^2 \quad (2.66)$$

Persamaan 2.66 diketahui sebagai teori simbu parallel (Walker, 2014).

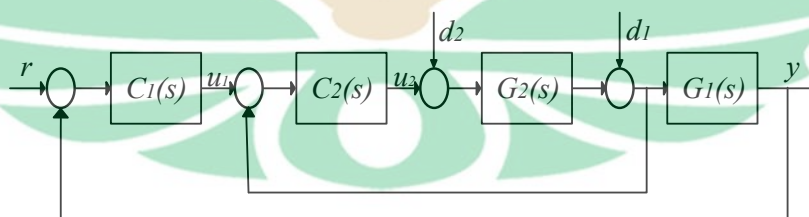
## 2.8 Mean Absolute Error

*Mean Absolute Error* (MAE) banyak digunakan untuk mengukur perbedaan antara nilai prediksi dan nilai aktual dari suatu sistem. MEA digunakan untuk mengevaluasi akurasi dari sistem yang direkomendasikan dengan persamaan 2.68 (Wang, 2018)

$$MAE = \frac{\sum_{n=1}^N |\hat{r}_n - r_n|}{N} \quad (2.68)$$

## 2.9 Desain Kontrol PID Seri

Kontrol seri adalah salah satu struktur kontrol kompleks yang marak ditemukan di dalam industri proses, dan diimplementasikan untuk meningkatkan respon dari sistem yang dikendalikan. Konfigurasi umum dari kontrol seri dapat dilihat pada Gambar 2.6 di mana terdapat dua loop kontrol yaitu kontrol *inner loop* yang mengontrol dinamika yang cepat dan *outer loop* yang mengontrol dinamika yang lambat.



Gambar 2. 6 Konfigurasi Kontrol Seri (Arrieta, 2008)

Perancangan kontrol *inner loop* memiliki cara yang sama dengan perancangan

kontrol PID pada umumnya. Perancangan kontrol PID *outerloop* dapat dilakukan setelah kontrol *inner loop* telah didapatkan. Untuk perancangan kontrol *outer loop* digunakan fungsi alih yang ditunjukkan oleh persamaan 2.69 (Arrieta, 2008).

$$G_e(s) = H_2(s)G_1(s) = \left( \frac{C_2(s)G_2(s)}{1 + C_2(s)G_2(s)} \right) G_1(s) \quad (2.69)$$

## 2.10 Posisi Penelitian

Tabel 2.7 merupakan penelitian yang telah dilakukan sebagai gambaran utama dalam tugas akhir yang akan dilakukan.

Tabel 2. 7 Tabel Posisis Penelitian

No.	Nama dan Tahun Publikasi	Judul Penelitian	Hasil
1.	Huu-Khoa Tran dan Juing-Shian Chiou, 2016	<i>PSO-Based Algorithm Applied to Quadcopter Micro Air Vehicle Controller Design</i>	Penerapan algoritma PSO ke dalam kontrol menunjukkan <i>settling time</i> yang cepat dan kestabilan sistem meningkat
2.	Jack Connor, Mehdi Syedmahmoudian dan Ben Horan, 2017	<i>Using Particle Swarm Optimization for PID Optimization for Altitude on a Quadcopter</i>	Algoritma PSO dapat digunakan untuk <i>tuning</i> kontrol PID dengan hasil yang dapat meningkatkan kinerja dari sistem dan menghemat waktu dalam pencarian nilai parameter kontrol
3.	Huu-Khoa Tran dan Thanh Nam Nguyen, 2018	<i>Flight Motion Controller Design Using Genetic Algorithm for a Quadcopter</i>	Penerapan algoritma GA kedalam kontrol PID menunjukkan hasil yang lebih baik dibandingkan dengan kontrol PID biasa untuk melakukan <i>trajectory tracking</i>
4.	Irvan Kurnia Wicaksono, 2020	Perancangan Kontrol <i>Trajectory Tracking</i> Menggunakan Algoritma EPO Pada <i>Hexacopter</i> Untuk Gerakan <i>Longitudinal</i>	