
**UNIVERSITÀ POLITECNICA DELLE MARCHE**

Dipartimento di Ingegneria dell'Informazione

Corso di Laurea triennale in Ingegneria Informatica e dell'automazione

---

# Modellazione matematica e simulazione del sistema di controllo di un drone

## Mathematical modeling and simulation of a drone control system

Relatore: Chiar.mo
**Prof. Orlando Giuseppe**

Correlatore: Chiar.mo
**Prof. Ippoliti Gianluca**

Candidato:
**Stama Alfredo**

A.A. 2018/2019

# Contents

**1 Introduzione** **1**
1.1 Introduzione al drone quadricottero . . . . . 1

**2 Modellizzazione Matematica** **3**
2.1 Sistemi di Riferimento & Angoli di Eulero . . . . . 3
2.2 Cinematica Rotazionale . . . . . 6
2.3 Motori Brushless . . . . . 7
2.4 Dinamica Traslazionale . . . . . 9
2.5 Dinamica Rotazionale . . . . . 10
2.6 Modello Non Lineare . . . . . 11
2.7 Modello Lineare . . . . . 12
2.7.1 Linearizzazione . . . . . 13
2.7.2 Controllabilità e Osservabilità . . . . . 16

**3 Sintesi del Controllore** **17**
3.1 Strategia di Controllo tramite PID . . . . . 17
3.1.1 PID Posizione/Altitudine . . . . . 21
3.1.2 PID Assetto . . . . . 21
3.1.3 PID Velocità Angolari . . . . . 21
3.2 Applicazione del Controllore . . . . . 22

**4 Simulazione** **23**
4.1 Implementazione generale . . . . . 23
4.2 Script MATLAB . . . . . 23
4.3 Implementazione del PID Posizione/Altitudine . . . . . 24
4.4 Implementazione del PID Assetto . . . . . 26
4.5 Implementazione del PID Velocità Angolari . . . . . 27
4.6 Implementazione del Modello Lineare . . . . . 28
4.7 Implementazione del Modello Non Lineare . . . . . 28
4.8 Risultati simulazione . . . . . 31

**5 Conclusioni** **33**

**References** **34**

# 1 Introduzione

## 1.1 Introduzione al drone quadricottero

Un UAV (Unmanned Aerial Vehicle), comunemente noto come **drone**, è un velivolo caratterizzato dall'assenza di un pilota umano a bordo. Il loro utilizzo è ampiamente diffuso in ambito militare e, in particolare negli ultimi anni, sta prendendo piede anche in applicazioni civili quali sorveglianza aerea, telerilevamento e ricerca. Il vantaggio di un veicolo senza pilota risiede nel fatto che gli UAV possono essere utilizzati in situazioni di elevato pericolo per la vita, lì dove le condizioni di operazione risultano piuttosto ostiche o addirittura inaccessibili. Il disastro di Fukushima in Giappone nel 2011 ne è un esempio: dei droni *Global Hawk* sono stati utilizzati per monitorare i reattori dopo le esplosioni.

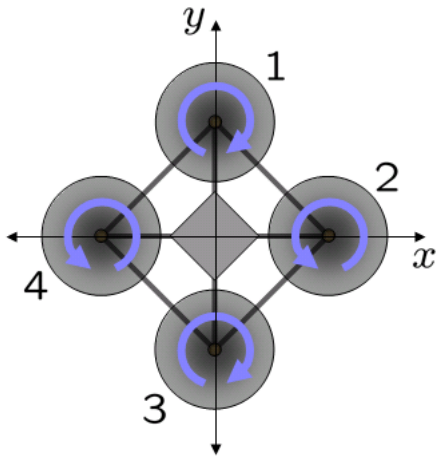


Figure 1: Verso di rotazione dei propulsori

Un drone quadricottero, o quadrirotore, è un aeromobile dotato di quattro eliche orizzontali equamente distanziate dal corpo centrale. L'unico svantaggio di questa tipologia di veivoli è dovuta alla bassa velocità di crociera. Tuttavia, la modalità di volo stabile unita alla sua capacità di *hovering* permette al quadricottero di presentarsi in maniera molto versatile e promettente. Le prime generazioni di quadricotteri erano adibite al trasporto di passeggeri, sfruttando il decollo e l'atterraggio verticale. Le generazioni più moderne, anche grazie alla loro dimensione ridotta, sono progettate per volare senza pilota utilizzando un sistema di controllo e dei sensori che permettono il volo sia in ambienti interni che esterni. Le eliche sono accoppiate a due a due in maniera opposta (1 e 3, 2 e 4 nella Figure (1)). Una coppia di motori *brushless* ruota in senso orario, l'altra in senso anti-orario. Dunque, se i rotori operano tutti alla stessa velocità, i momenti generati dalle coppie di motori si annullano a vicenda, permettendo al drone di mantenere una direzione durante il volo. Il quadricottero, quindi, a differenza dell'architettura tradizionale, non ha bisogno di un rotore di coda. L'imbardata, infatti, è controllata variando le velocità di una coppia di rotori, evitando così che il momento torcente generato dalla prima coppia venga contrastato dal momento relativo all'altra coppia di rotori. L'altitudine è controllata variando la spinta di ogni singolo motore di una stessa quantità. Per muovere il drone lateralmente, si fa girare un motore più rapidamente, diminuendo in contemporaneità la velocità del motore diametralmente opposto.

(a) Il drone varia l'altitudine variando la spinta di ogni singolo motore in egual misura

(b) Il beccheggio varia applicando più spinta ad un singolo motore e diminuendo quella del motore diametralmente opposto

(c) L'imbardata varia modificando la velocità di una coppia di motori

Il controllo di un drone quadricottero è un problema piuttosto interessante in quanto ci si trova di fronte ad un sistema *under-actuated*: con soltanto 4 input a disposizione (le velocità dei rotori), bisogna controllare 6 gradi di libertà ($x$, $y$ e $z$ lungo gli assi insieme a beccheggio, imbardata e rollìo). La dinamica necessaria ad ottenere un controllo di questo genere, risulta essere fortemente non lineare, specialmente se si tiene conto degli effetti aerodinamici.
In questo lavoro si eseguirà una modellazione matematica basata principalmente sui seguenti lavori: (Bouabdallah, 2007), (Selby, n.d.), ottenendo, seppur con varie semplificazioni, un modello non lineare. In seguito, il modello ottenuto sarà linearizzato come descritto in (Sabatino, 2015). Si lavorerà sul modello linearizzato per ottenere un controllore accettabile che verrà testato anche sul modello non lineare. Si procederà, infine, ad un confronto tra le performance dei due modelli.

# 2 Modellizzazione Matematica

Lo scopo di questo capitolo è quello di passare in rassegna le varie tappe necessarie alla modellazione matematica del quadrirotore. Verrà analizzata la cinematica e la dinamica, rotazionale e traslazionale del drone, ricavando in questo modo un totale di 12 equazioni differenziali che andranno a descriverne la dinamica. Prima di passare alla modellazione vera e propria, è necessario fare alcune assunzioni in merito al drone.

- La struttura del drone è una struttura rigida, simmetrica, il cui centro di massa coincide con il *body-frame* del veicolo.
- Le eliche dei propulsori sono rigide, quindi è trascurabile la deformazione dovuta alle alte velocità e alla flessibilità del materiale.
- La Terra è considerata piatta e non-rotante.
- Le velocità dei fluidi adiacenti al drone (vento) sono trascurabili.
- Il *ground-effect* è trascurabile.

Di seguito, inoltre, è riportato l'elenco di alcune delle variabili principali che verranno adoperate nei paragrafi successivi:

$x$ La posizione del drone nel quadro di riferimento inerziale (positiva verso Nord)

$y$ La posizione del drone nel quadro di riferimento inerziale (positiva verso Est)

$z$ La posizione del drone nel quadro di riferimento inerziale (positiva verso il basso)

$x_b$ La posizione del drone nel quadro di riferimento del corpo rigido (positiva verso Nord)

$y_b$ La posizione del drone nel quadro di riferimento del corpo rigido (positiva verso Est)

$z_b$ La posizione del drone nel quadro di riferimento del corpo rigido (positiva verso il basso)

$\phi$ Angolo di roll attorno $x_b$

$\theta$ Angolo di pitch attorno $y_b$

$\psi$ Angolo di yaw attorno $z_b$

$p$ Velocità angolare attorno $x_b$

$q$ Velocità angolare attorno $y_b$

$r$ Velocità angolare attorno $z_b$

## 2.1 Sistemi di Riferimento & Angoli di Eulero

Esistono diversi metodi utilizzati per rappresentare l'orientamento, due tra i più comuni sono gli **Angoli di Eulero** e i **Quaternioni**. La differenza sostanziale, sottolineata da (Tyler, n.d.), è la seguente: gli Angoli di Eulero, benché più semplici ed intuitivi, possono creare degli orientamenti ambigui in quanto le tre coordinate potrebbero non definire un unico orientamento. Questo problema, conosciuto come *gimbal-lock*, occorre quando il valore di alcuni angoli, come specificato da (CH-Robotics, n.d.), raggiunge valori critici. Infatti, se il valore dell'angolo di pitch $\theta$ dovesse raggiungere i 90 gradi, il sensore a bordo non riuscirebbe più a distinguere tra roll e yaw.

Figure 2: Orientamento ambiguo per le misurazioni tramite Angoli di Eulero.

Guardando infatti la Fig. (2), si può notare come l'orientazione rappresentata può essere frutto di due rotazioni differenti: un'imbardata seguita da un beccheggio oppure un beccheggio seguito da un rollìo. Si può ovviare a questo problema con due metodi differenti. Il primo metodo consiste nell'utilizzo dei Quaternioni, una tecnica di misura non affetta dal *gimbal-lock*. Il quaternione è un vettore di quattro elementi utilizzato per codificare qualsiasi rotazione in uno spazio tridimensionale. Il secondo metodo prevede di utilizzare gli Angoli di Eulero ad una condizione: gli angoli misurati dai sensori devono mantenersi piccoli. Per motivi di semplicità, sia pratica (implementazione) che teorica (linearizzazione basata sul *modello per piccole oscillazioni*), si è scelto di utilizzare gli Angoli di Eulero. Prima di passare alla descrizione degli Angoli di Eulero, è necessaria una introdzione al concetto di *quadro di riferimento inerziale* e di *quadro di riferimento del corpo rigido*.

1. **Quadro di riferimento inerziale.**
   È il quadro di riferimento fisso con il sistema Terra e le sue coordinate coincidono con le direzioni cardinali (Nord, Est). Vegono definite le coordinate $(x, y, z)$. La $z$ ha direzione positiva verso il basso.

2. **Quadro di riferimento del corpo rigido.**
   È il quadro di riferimento la cui origine coincide con il centro di massa del quadricottero. Vengono definite le coordinate $(x_b, y_b, z_b)$.

Ora, è necessario un metodo per convertire le coordinate da un sistema all'altro e viceversa. Ciò tornerà utile quando si vorranno leggere i valori dei sensori a bordo del drone per poi tradurli in valori inerziali o il contrario.
Gli Angoli di Eulero spiegano come ogni orientazione di un quadro di riferimento può essere ottenuta tramite la composizione di tre rotazioni attorno agli assi di riferimento. Il caso triviale, in cui non avviene nessuna rotazione, è descritto dalla seguente equazione con matrice di rotazione coincidente con la matrice identità $I_3$.

$$\begin{bmatrix} x_b \\ y_b \\ z_b \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{1}$$

Se, partendo dalla situazione in cui gli assi di riferimento dei due sistemi coincidono, si effettuasse una rotazione di un certo angolo attorno ad uno solo degli assi del quadro di riferimento del corpo rigido, si osserverebbe l'asse di rotazione rimanere parallelo all'asse corrispondente nel quadro inerziale. Gli assi rimanenti variano con una funzione delle componenti inerziali e dell'angolo di rotazione. Nella pagina seguente si osservano le tre tipologie di rotazioni possibili.

- Rotazione attorno $z_b$ di un angolo $\psi$

$$\begin{bmatrix} x_b \\ y_b \\ z_b \end{bmatrix} = \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}; \quad \begin{aligned} x_b &= x\cos(\psi) + y\sin(\psi) \\ y_b &= -x\sin(\psi) + y\cos(\psi) \\ z_b &= z \end{aligned}$$

- Rotazione attorno $y_b$ di un angolo $\theta$

$$\begin{bmatrix} x_b \\ y_b \\ z_b \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 & -sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}; \quad \begin{aligned} x_b &= x\cos(\theta) + z\sin(\theta) \\ y_b &= y \\ z_b &= x\sin(\theta) + z\cos(\theta) \end{aligned}$$

- Rotazione attorno $x_b$ di un angolo $\phi$

$$\begin{bmatrix} x_b \\ y_b \\ z_b \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}; \quad \begin{aligned} x_b &= x \\ y_b &= y\cos(\phi) + z\sin(\phi) \\ z_b &= -y\sin(\phi) + z\cos(\phi) \end{aligned}$$

La matrice con la quale si effettua la trasformazione è chiamata Matrice di Rotazione, indicata con la lettera $\mathbf{R}$. Questa matrice, essendo ortonormale, gode della seguente proprietà:

$$\mathbf{R}^T = \mathbf{R}^{-1} \tag{2}$$

È necessario sottolineare che, qualora si compia più di una rotazione in sequenza, l'ordine in cui queste vengono effettuate dev'essere consistente. A parità di angoli, una rotazione $\mathbf{R}(\phi)\mathbf{R}(\theta)\mathbf{R}(\psi)$ produce un risultato differente dalla rotazione $\mathbf{R}(\theta)\mathbf{R}(\phi)\mathbf{R}(\psi)$. La comodità delle trasformazioni tramite Matrice di Rotazione consiste nella possibilità di concatenare tre rotazioni ottenendo così un'unica matrice per passare direttamente dal sistema di riferimento inerziale al sistema di riferimento del corpo rigido e viceversa.

$$\begin{bmatrix} x_b \\ y_b \\ z_b \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{bmatrix} \begin{bmatrix} \cos(\psi) & 0 & -sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}; \tag{3}$$

$$\begin{bmatrix} x_b \\ y_b \\ z_b \end{bmatrix} = \begin{bmatrix} \cos(\theta)\cos(\psi) & \cos(\theta)\sin(\psi) & -\sin(\theta) \\ -\cos(\phi)\sin(\psi) + \sin(\phi)\sin(\theta)\cos(\psi) & \cos(\phi)\cos(\psi) + \sin(\phi)\sin(\theta)\sin(\psi) & \sin(\phi)\cos(\theta) \\ \sin(\phi)\sin(\psi) + \cos(\phi)\sin(\theta)\cos(\psi) & -\sin(\phi)\cos(\psi) + \cos(\phi)\sin(\theta)\sin(\psi) & \cos(\phi)\cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{4}$$

Per semplicità, si utilizzerà la notazione $C_i^j$ per indicare una matrice di rotazione dal quadro di riferimento $i$ al quadro di riferimento $j$. Nel caso in cui si volesse ottenere la

trasformazione di coordinate inversa, e cioè dal sistema di riferimento $j$, al sistema di riferimento $i$, tenendo conto dell'equazione (2), è possibile scrivere

$$C_j^i = (C_i^j)^T$$

Dunque, alla luce di ciò, indicando con $G$ il quadro di riferimento inerziale e con $b$ il quadro di riferimento del corpo rigido, l'equazione (4) può essere scritta come:

$$\begin{bmatrix} x_b \\ y_b \\ z_b \end{bmatrix} = C_G^b \begin{bmatrix} x \\ y \\ z \end{bmatrix}; \tag{5}$$

Nel caso in cui si volesse passare dal sistema di riferimento del corpo rigido al sistema di riferimento inerziale, basterebbe semplicemente calcolare:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = C_b^G \begin{bmatrix} x_b \\ y_b \\ z_b \end{bmatrix}; \tag{6}$$

Dove si ha che

$$C_b^G = (C_G^b)^T = \begin{bmatrix} \cos(\psi)\cos(\theta) & -\sin(\psi)\cos(\phi) + \cos(\psi)\sin(\theta)\sin(\phi) & \sin(\phi)\sin(\psi) + \cos(\phi)\sin(\theta)\cos(\psi) \\ \sin(\psi)\cos(\theta) & \cos(\phi)\cos(\psi) + \sin(\phi)\sin(\theta)\sin(\psi) & -\cos(\psi)\sin(\phi) + \sin(\psi)\sin(\theta)\cos(\phi) \\ -\sin(\theta) & \cos(\theta)\sin(\phi) & \cos(\theta)\cos(\psi) \end{bmatrix} \tag{7}$$

La matrice in questione permette di ricavare, dunque, le equazioni che descrivono la cinematica traslazionale del drone. Bisogna semplicemente calcolare le derivate delle posizioni nel quadro di riferimento del corpo rigido e ruotarle attraverso la matrice $C_b^G$ ottenendo così i loro valori nel quadro di riferimento inerziale.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = C_b^G \begin{bmatrix} \dot{x_b} \\ \dot{y_b} \\ \dot{z_b} \end{bmatrix} \tag{8}$$

Questa equazione descrive la cinematica traslazionale del drone, cioè il movimento lineare del drone lungo i tre assi del quadro di riferimento inerziale, indipendentemente dalle cause che lo hanno prodotto.

### 2.2 Cinematica Rotazionale

Dal momento che le velocità angolari sono definite nel quadro di riferimento del corpo rigido e gli Angoli di Eulero in quadri di riferimento intermedi, è possibile adoperare le matrici di rotazione definite sopra per determinare la relazione tra le due grandezze, esprimendola nel quadro di riferimento del corpo rigido, proprio come le velocità angolari.

$$\boldsymbol{\omega} = \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \mathbf{R}(\phi)\mathbf{R}(\theta) \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} + \mathbf{R}(\phi) \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} \tag{9}$$

Come si può osservare, per sapere quale sia il valore di $\dot{\psi}$ nel quadro di riferimento del corpo rigido, è necessario compiere le rotazioni rimanenti (l'ordine che si è scelto per calcolare la matrice di rotazione nel paragrafo precedente è $\mathbf{R}(\phi)\mathbf{R}(\theta)\mathbf{R}(\psi)$ ed è necessario che rimanga consistente). Ciò viene fatto moltiplicando $\dot{\psi}$ per le matrici di rotazione rimanenti $\mathbf{R}(\phi)\mathbf{R}(\theta)$. Lo stesso discorso viene ripetuto per $\dot{\theta}$ mentre $\dot{\phi}$, trattandosi dell'ultimo angolo, si trova già nel quadro di riferimento del corpo rigido e non ha bisogno di essere trasformato. Svolgendo i calcoli dell'eq. (9), si ricava:

$$\boldsymbol{\omega} = \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\sin(\theta) \\ 0 & \cos(\phi) & \sin(\phi)\cos(\theta) \\ 0 & -\sin(\phi) & \cos(\phi)\cos(\theta) \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \mathbf{S} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{10}$$

$$\mathbf{S}^{-1} = \begin{bmatrix} 1 & \sin(\phi)\tan(\theta) & \cos(\phi)\tan(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \frac{\sin(\phi)}{\cos(\theta)} & \frac{\cos(\phi)}{\cos(\theta)} \end{bmatrix}$$

Con l'inversa della matrice $\mathbf{S}$ si possono esprimere i valori delle derivate degli Angoli di Eulero.

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin(\phi)\tan(\theta) & \cos(\phi)\tan(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \frac{\sin(\phi)}{\cos(\theta)} & \frac{\cos(\phi)}{\cos(\theta)} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{11}$$

### 2.3 Motori Brushless

Il motore brushless è un motore elettrico, pilotato da un inverter, ampiamente utilizzato in applicazioni del genere. Il drone quadricottero monta quattro motori di questo tipo, ognuno dei quali viene alimentato da batterie elettriche trasportate a bordo del drone. Il controllo di questo genere di motori viene effettuato da un controller elettronico, uno per ogni motore presente. Qui di seguito si seguiranno i calcoli svolti in (Gibiansky, 2012) nella sezione *Motors*. Si ricaverà dapprima l'espressione della potenza per un motore brushless partendo dal calcolo dalla formula del momento meccanico $\tau$:

$$\tau = K_t(I - I_0) \tag{12}$$

In questa equazione, $K_t$ rappresenta una costante di proporzionalità, $I$ è la corrente e $I_0$ è la corrente senza carico. Dall'equazione è possibile ricavare l'espressione della corrente, che sarà uguale a:

$$I = \frac{\tau + K_t I_0}{K_t} \tag{13}$$

Ottenuta l'espressione della corrente, per arrivare ad esprimere la potenza, è necessario calcolare la ddp $V$ del motore con la seguente formula:

$$V = IR_m + E_m\omega \tag{14}$$

In questa equazione $R_m$ indica la resistenza del motore, $E_m$ la forza contro-elettromotrice

e $\omega$ la velocità angolare del motore. All'interno dell'equazione può essere sostituita l'espressione della corrente ricavata nell'equazione (13), ottenendo quindi:

$$V = \frac{\tau + K_t I_0}{K_t} R_m + E_m \omega \tag{15}$$

Alla luce di ciò, è possibile esprimere in maniera completa l'espressione della potenza meccanica del motore brushless.

$$\begin{aligned} P_m = IV = \frac{\tau + K_t I_0}{K_t} \Big( \frac{\tau + K_t I_0}{K_t} R_m + E_m \omega \Big) = \\ = \frac{\tau + K_t I_0}{K_t} \cdot \frac{\tau Rm + K_t I_0 Rm + E_m \omega K_t}{K_t} = \\ = \frac{(\tau + K_t I_0)(\tau Rm + K_t I_0 Rm + E_m \omega K_t)}{K_t^2} \end{aligned} \tag{16}$$

Ora viene fatta l'ipotesi che la resistenza e la corrente senza carico siano piuttosto piccole, cioè $R_m \approx I_0 \approx 0$. L'equazione (16) diventa come segue:

$$P_m = \frac{\tau E_m \omega}{K_t} \tag{17}$$

Prima di andare avanti, è necessario precisare che la potenza consumata dal drone per rimanere in volo stabile in un punto dello spazio equivale alla potenza necessaria ad ottenere una spinta muovendo una colonna d'aria. Si indica con $P_h$ la potenza necessaria a rimanere fermo in un punto dello spazio in volo.

$$P_h = T v_h \tag{18}$$

Dove $T$ rappresenta la spinta, di cui vogliamo ricavare infine un'espressione, e $v_h$ rappresenta la velocità indotta sul drone dal propulsore. Bisogna fare una precisazione, cioè che la spinta vada espressa come segue.

$$T = 2\rho A v_h^2 \tag{19}$$

In quest'ultima equazione, $\rho$ rappresenta la densità dell'aria, mentre $A$ l'area coperta dalle eliche del rotore. Ricavando la velocità indotta, si ottiene:

$$v_h = \sqrt{\frac{T}{2\rho A}} \tag{20}$$

Volendo riscrivere l'espressione di $P_h$, andando a sostituire l'espressione di $v_h$ appena ricavata al suo interno, si otterrebbe:

$$P_h = \frac{T^{\frac{3}{2}}}{\sqrt{2\rho A}} \tag{21}$$

Ricordando l'equazione della potenza meccanica (17), all'interno della quale è possibile sostituire la seguente espressione del momento meccanico $\tau = k_\tau T$, dove $k_\tau$ è una costante di proporzionalità, si ricava:

$$P_m = \frac{E_m k_\tau T \omega}{K_t} \tag{22}$$

Tenendo a mente il significato di $P_h$ espresso in precedenza, si può porre la seguente uguaglianza tra le equazioni (22) e (21):

$$P_m = P_h \tag{23}$$

Da cui si ottiene, isolando il termine $T$ della spinta e indicando con $k_T$ il coefficiente di spinta, l'espressione voluta:

$$T = \Big[\Big(\frac{E_m k_\tau \sqrt{2\rho A}}{K_t}\Big)\omega\Big]^2 = k_T \omega^2 \tag{24}$$

### 2.4 Dinamica Traslazionale

Le equazioni lineari del moto del drone (25) sono definite nel quadro di riferimento inerziale. La forza totale agente sul drone è la risultante della forza gravitazionale e della spinta totale dovuta alla propulsione dei motori. La spinta totale, definita nel quadro di riferimento del corpo rigido, è trasformata, con la matrice di rotazione definita sopra, in riferimento al quadro inerziale. È possibile, al fine di ottenere un modello più complesso, considerare anche le forze di trascinamento causate dagli attriti nel quadro di riferimento del corpo rigido, con costanti d'attrito differenti per ogni asse di riferimento.

$$m\ddot{X}^G = F_g - F_T^G \tag{25}$$

Con il termine $\ddot{X}^G$ si vuole indicare il vettore delle accelerazioni lungo gli assi longitudinali nel quadro di riferimento inerziale, che descrivono la dinamica traslazionale del drone. La massa è indicata con $m$.

$$\ddot{X}^G = \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix}$$

La forza di gravità ha componenti esclusivamente lungo l'asse **z**, infatti come si può vedere, le altre due componenti risultano nulle. L'accelerazione gravitazionale è indicata con $g$

$$F_g = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix}$$

Infine, la spinta totale dovuta alla propulsione dei motori, può essere espressa come di seguito, tenendo a mente che $F_T^G$ è la spinta totale nel quadro di riferimento inerziale, $F_T^b$ è la spinta totale nel quadro di riferimento del corpo rigido e $C_b^G$ rappresenta la matrice

di rotazione dal quadro di riferimento del corpo rigido al quadro di riferimento inerziale.

$$F_T^G = C_b^G F_T^b;$$

La spinta totale dovuta ai quattro motori $F_T^b$, che corrisponde al primo dei quattro input di controllo, può essere scritta come la somma delle spinte dei quattro motori:

$$F_T^b = U_1 = \sum_{i=1}^{4} T_i = k_T(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) \tag{26}$$

Andando a sostituire all'interno della (25) e dividendo ambo i membri per $m$ si ottiene la seguente equazione:

$$\ddot{X}^G = \frac{1}{m}\begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} - \frac{1}{m} C_b^G F_T^b; \tag{27}$$

Sostituendo la matrice di rotazione e svolgendo i calcoli:

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} -\frac{1}{m}(\cos(\phi)\cos(\psi)\sin(\theta) + \sin(\phi)\sin(\psi))F_T^b \\ -\frac{1}{m}(\cos(\phi)\sin(\psi)\sin(\theta) - \cos(\psi)\sin(\phi))F_T^b \\ -\frac{1}{m}(\cos(\phi)\cos(\theta))F_T^b + g \end{bmatrix} \tag{28}$$

E quindi, riscrivendo l'equazione (28) sostituendo il primo input di controllo si ottiene:

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} -\frac{1}{m}(\cos(\phi)\cos(\psi)\sin(\theta) + \sin(\phi)\sin(\psi))U_1 \\ -\frac{1}{m}(\cos(\phi)\sin(\psi)\sin(\theta) - \cos(\psi)\sin(\phi))U_1 \\ -\frac{1}{m}(\cos(\phi)\cos(\theta))U_1 + g \end{bmatrix} \tag{29}$$

### 2.5 Dinamica Rotazionale

Le equazioni del moto rotazionale del drone (30) sono definite nel quadro di riferimento del corpo rigido, in questo modo è possibile misurare le rotazioni attorno al centro di massa del drone. Inoltre, poiché è stata ipotizzata la simmetria del quadricottero, la matrice $J_b$ del momento di inerzia del corpo del drone risulta essere diagonale. Le equazioni, quindi, possono essere riassunte dalla seguente equazione:

$$J_b\dot{\omega} = \tau_m - \tau_g - (\omega \times J_b\omega) \tag{30}$$

Nell'equazione compare dapprima la matrice del momento di inerzia che, essendo diagonale, prevede che i prodotti d'inerzia siano nulli (gli elementi al di fuori della diagonale rappresentano i prodotti d'inerzia). $J_x$, $J_y$ e $J_z$ indicano i momenti d'inerzia.

$$J_b = \begin{bmatrix} J_x & 0 & 0 \\ 0 & J_y & 0 \\ 0 & 0 & J_z \end{bmatrix} \tag{31}$$

Con il vettore $\dot{\omega}$ si vuole indicare le accelerazioni angolari compiute dal drone attorno al suo centrro di massa.

$$\dot{\omega} = \begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix}; \tag{32}$$

Il momento meccanico $\tau_m$, esercitato sul drone dai motori, racchiude al suo interno, rispettivamente, il secondo, terzo e quarto input di controllo. Il parametro $l$ rappresenta la distanza del motore dal centro di massa del drone.

$$\tau_m = \begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} lk_T(\omega_4^2 - \omega_2^2) \\ lk_T(\omega_1^2 - \omega_3^2) \\ k_d(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2) \end{bmatrix}; \tag{33}$$

Il momento meccanico $\tau_g$, esercitato sul drone dall'effetto giroscopico, può essere trascurato poiché $J_r$, il momento di inerzia del rotore, è molto piccolo.

$$\tau_g = \begin{bmatrix} \dot{\theta} \\ -\dot{\phi} \\ 0 \end{bmatrix} J_r \sum_{i=1}^{4} (-1)^{i+1} \omega_i; \tag{34}$$

Infine, il seguente prodotto vettoriale descrive l'effetto giroscopico dovuto alla rotazione dei motori combinata alla rotazione del corpo. Ciò è causato dal fatto che l'asse di rotazione del rotore ruota con la velocità angolare del drone nel quadro di riferimento del corpo rigido.

$$(\omega \times J_b \omega) = \begin{bmatrix} \dot{\theta}\dot{\psi}(J_z - J_y) \\ \dot{\psi}\dot{\phi}(J_x - J_z) \\ \dot{\theta}\dot{\phi}(J_y - J_z) \end{bmatrix} \tag{35}$$

Dunque, isolando $\dot{\omega}$ nella (30), alla luce di quanto detto fin'ora, si ottiene:

$$\begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} \frac{1}{J_x}[(J_y - J_z)\dot{\theta}\dot{\psi} + U_2] \\ \frac{1}{J_y}[(J_z - J_y)\dot{\psi}\dot{\phi} + U_3] \\ \frac{1}{J_z}[(J_x - J_y)\dot{\theta}\dot{\phi} + U_4] \end{bmatrix} \tag{36}$$

### 2.6 Modello Non Lineare

Nei paragrafi precedenti sono state ricavate le equazioni che permettono di descrivere, in maniera sufficientemente accurata, la dinamica del drone quadricottero. Queste equazioni sono non-lineari e in seguito saranno semplificate usando specifiche assunzioni che permetteranno di sintetizzare un controllore adeguato. Le 12 equazioni, intanto, vengono riassunte qui di seguito.

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin(\phi)\tan(\theta) & \cos(\phi)\tan(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \frac{\sin(\phi)}{\cos(\theta)} & \frac{\cos(\phi)}{\cos(\theta)} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{10}$$

$$\begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} \frac{1}{J_x}[(J_y - J_z)\dot{\theta}\dot{\psi} + U_2] \\ \frac{1}{J_y}[(J_z - J_y)\dot{\psi}\dot{\phi} + U_3] \\ \frac{1}{J_z}[(J_x - J_y)\dot{\theta}\dot{\phi} + U_4] \end{bmatrix} \tag{17}$$

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} -\frac{1}{m}(\cos(\phi)\cos(\psi)\sin(\theta) + \sin(\phi)\sin(\psi))U_1 \\ -\frac{1}{m}(\cos(\phi)\sin(\psi)\sin(\theta) - \cos(\psi)\sin(\phi))U_1 \\ -\frac{1}{m}(\cos(\phi)\cos(\theta))U_1 + g \end{bmatrix} \tag{15}$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} \cos(\psi)\cos(\theta) & -\sin(\psi)\cos(\phi) + \cos(\psi)\sin(\theta)\sin(\phi) & \sin(\phi)\sin(\psi) + \cos(\phi)\sin(\theta)\cos(\psi) \\ \sin(\psi)\cos(\theta) & \cos(\phi)\cos(\psi) + \sin(\phi)\sin(\theta)\sin(\psi) & -\cos(\psi)\sin(\phi) + \sin(\psi)\sin(\theta)\cos(\phi) \\ -\sin(\theta) & \cos(\theta)\sin(\phi) & \cos(\theta)\cos(\psi) \end{bmatrix} \begin{bmatrix} \dot{x_b} \\ \dot{y_b} \\ \dot{z_b} \end{bmatrix} \tag{4}$$

Si può, quindi, scrivere il vettore delle variaibili di stato nel seguente modo:

$$\boldsymbol{X} = \begin{bmatrix} \phi & \theta & \psi & \dot{\phi} & \dot{\theta} & \dot{\psi} & \dot{x} & \dot{y} & \dot{z} & x & y & z \end{bmatrix}^T \in \mathbb{R}^{12} \tag{37}$$

### 2.7 Modello Lineare

La linearizzazione del sistema si compie attorno ad un punto di equilibrio e, per questioni di semplicità, avviene su un modello non lineare semplificato, denominato *modello per piccole oscillazioni*, che sarà ricavato nella prima parte di questo paragrafo. Questo modello si ottiene supponendo che il drone copra piccoli angoli di assetto e abbia imbardata nulla. Si ipotizzi, dapprima, che gli angoli che rappresentano l'assetto del drone, assumano dei valori molto piccoli, con conseguenza che la funzione coseno tenda all'unità, mentre la funzione seno tenda al suo argomento:

$$\phi \approx \theta \approx 0 \implies \begin{cases} \cos(\phi) \approx \cos(\theta) \approx 1 \\ \sin(\phi) \approx \sin(\theta) \approx \phi \approx \theta \approx 0 \end{cases} \tag{38}$$

Supposto ciò, la relazione (11) può essere scritta come segue nell'equazione (39), con la matrice che diventa diagonale. Ciò implica che, nel *modello per piccole oscillazioni*, le derivate degli Angoli di Eulero vengono a coincidere con le rispettive velocità angolari. Per comprendere a fondo questo concetto, bisogna considerare i frame intermedi attraverso i quali si effettuano le rotazioni definite dagli Angoli di Eulero. I quadri di riferimento intermedi diventano pressoché coincidenti in quanto le rotazioni che li differenziavano, ora diventano trascurabili.

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \approx \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{39}$$

Considerando l'angolo di imbardata approssimativamente nullo e avendo sostituito all'interno della relazione (29) le precedenti considerazioni sugli angoli di assetto, ottengo la relazione (41)

$$\psi \approx 0 \implies \cos(\psi) \approx 1 \quad \sin(\psi) \approx \psi \approx 0 \tag{40}$$

$$\begin{bmatrix}\ddot{x}\\ \ddot{y}\\ \ddot{z}\end{bmatrix} \approx \begin{bmatrix}-\frac{1}{m}(\theta\cos(\psi)+\phi\sin(\psi))U_1\\ -\frac{1}{m}(\theta\sin(\psi)-\phi\cos(\psi))U_1\\ -\frac{1}{m}U_1+g\end{bmatrix} \approx \begin{bmatrix}-\frac{1}{m}\theta U_1\\ \frac{1}{m}\phi U_1\\ -\frac{1}{m}U_1+g\end{bmatrix} \tag{41}$$

Per quanto riguarda le equazioni che descrivono la dinamica rotazionale del drone, valgono sempre le considerazioni fatte in questo paragrafo sugli angoli di assetto e sull'imbardata. Segue, dunque, che il termine che prevede il prodotto delle derivate degli angoli all'interno dell'eq. (36) si annulli in quanto gli angoli sono considerati pressoché nulli. (Vedi (38) e (40)). Le equazioni ottenute risultano lineari! Rimarranno, infatti, invariate tra il *modello per piccole oscillazioni* e il modello lineare. La dinamica rotazionale, come specificato nel relativo paragrafo 2.5, si occupa di descrivere le rotazioni attorno al centro di massa del corpo rigido ma, dal moento che queste vengono quasi completamente annullate, ne segue che il relativo modello risulta piuttosto semplice, quindi lineare.

$$\begin{bmatrix}\ddot{\phi}\\ \ddot{\theta}\\ \ddot{\psi}\end{bmatrix} \approx \begin{bmatrix}\frac{1}{J_x}U_2\\ \frac{1}{J_y}U_3\\ \frac{1}{J_z}U_4\end{bmatrix} \tag{42}$$

La cinematica traslazionale, nel *modello per piccole oscillazioni*, rappresenta la via di mezzo tra il modello non lineare e quello lineare. Nel primo, si ha la matrice di rotazione nella sua forma completa $C_b^G$, nel secondo, la matrice di rotazione viene completamente ridotta ad una matrice identità, come si vedrà più avanti. Si è scelto, in questo passaggio, di non sostituire gli angoli direttamente con zero, ma di lasciare l'approssimazione della funzione seno con il suo argomento per rappresentare al meglio le piccole oscillazioni; la funzione coseno viene ridotta all'unità e quindi scompare l'argomento in quanto elemento neutro del prodotto. La relazione che se ne ricava è la seguente:

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{z}\end{bmatrix} \approx \begin{bmatrix}1 & -(\psi-\phi\theta) & \phi\psi+\theta\\ \psi & 1+\phi\theta\psi & -(\phi-\psi\theta)\\ -\theta & \phi & 1\end{bmatrix} \begin{bmatrix}\dot{x_b}\\ \dot{y_b}\\ \dot{z_b}\end{bmatrix} \tag{43}$$

### 2.7.1 Linearizzazione

Come detto in precedenza, per linearizzare il sistema è necessario un punto d'equilibrio, che supporremo essere un generico punto dello spazio con coordinate $(\bar{x}, \bar{y}, \bar{z})$. Si sta ipotizzando che il drone sia immobile in un generico punto dello spazio senza muoversi in una particolare direzione. Quindi, il vettore delle variabili di stato diventa:

$$\bar{\boldsymbol{X}} = \begin{bmatrix}0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \bar{x} & \bar{y} & \bar{z}\end{bmatrix}^T \in \mathbb{R}^{12}$$

La situazione descritta nello stato $\bar{\boldsymbol{X}}$ può essere ottenuta utilizzando un input costante:

$$\bar{\boldsymbol{u}} = \begin{bmatrix}mg & 0 & 0 & 0\end{bmatrix}^T \in \mathbb{R}^4$$

Un input del genere rappresenta la forza necessaria a contrastare la forza di gravità,

consentendo al drone di rimanere stabile in un punto dello spazio $(\bar{x}, \bar{y}, \bar{z})$, proprio come supposto dalle ipotesi. Infatti, inviando al drone il valore corrispondente alla forza di gravità nel primo input, la si annulla in quanto si generano forze uguali in modulo ma di verso opposto. Se scriviamo le equazioni del modello come $f(\boldsymbol{X}, \boldsymbol{u})$, si possono ricavare le matrici per il sistema lineare.

$$f(\boldsymbol{X}, \boldsymbol{u}) = \begin{cases} \dot{\phi} = p \\ \dot{\theta} = q \\ \dot{\psi} = r \\ \ddot{\phi} = \frac{U_2}{J_x} \\ \ddot{\theta} = \frac{U_3}{J_y} \\ \ddot{\psi} = \frac{U_4}{J_z} \\ \ddot{x} = -\frac{1}{m}\theta U_1 \\ \ddot{y} = \frac{1}{m}\phi U_1 \\ \ddot{z} = -\frac{1}{m}U_1 + g \\ \dot{x} = \dot{x_b} - \dot{y_b}(\psi - \phi\theta) + \dot{z_b}(\phi\psi + \theta) \\ \dot{y} = \dot{x_b}\psi + \dot{y_b}(1 + \phi\theta\psi) - \dot{z_b}(\phi - \psi\theta) \\ \dot{z} = -\dot{x_b}\theta + \dot{y_b}\phi + \dot{z_b} \end{cases} \tag{44}$$

Per scrivere le matrici canoniche della forma in spazio di stato del sistema, è necessario ricorrere alla matrice Jacobiana che ha la seguente forma:

$$J = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

$m$ indica il numero di equazioni mentre $n$ indica il numero di variabili di stato. Calcolando la matrice Jacobiana si è in grado di calcolare sia la matrice $\boldsymbol{A}$ che la matrice $\boldsymbol{B}$. Sia $m$ che $n$ hanno come valore 12 per la matrice $\boldsymbol{A}$; per la matrice $\boldsymbol{B}$ si ha che $n = 4$ come il numero di input a disposizione. L'ordine delle variabili $x_1, \cdots, x_{12}$, così come quello degli input $u_1, \cdots, u_4$ nel calcolo della matrice Jacobiana deve essere consistente con il vettore delle variabili di stato (37) e con il vettore degli input $\boldsymbol{u} = \begin{bmatrix} U_1 & U_2 & U_3 & U_4 \end{bmatrix}^T \in \mathbb{R}^4$ altrimenti si rischia di ottenere delle matrici con le colonne invertite, ottenendo equazioni non corrette. Procedendo quindi con i calcoli, si ottengono le suddette matrici:

$$\boldsymbol{A} = \left.\frac{\partial f(\boldsymbol{X},\boldsymbol{u})}{\partial \boldsymbol{X}}\right|_{\substack{\boldsymbol{X}=\bar{\boldsymbol{X}}\\ \boldsymbol{u}=\bar{\boldsymbol{u}}}} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -g & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ g & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

$$\boldsymbol{B} = \left.\frac{\partial f(\boldsymbol{X},\boldsymbol{u})}{\partial \boldsymbol{u}}\right|_{\substack{\boldsymbol{X}=\bar{\boldsymbol{X}}\\ \boldsymbol{u}=\bar{\boldsymbol{u}}}} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \frac{1}{J_x} & 0 & 0 \\ 0 & 0 & \frac{1}{J_y} & 0 \\ 0 & 0 & 0 & \frac{1}{J_z} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{-1}{m} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Si è certi della linearità del modello osservando che in ogni riga è presente al più soltanto un termine non nullo. Ciò esclude eventuali prodotti tra variabili all'interno di una singola equazione. Si può quindi calcolare l'equazione canonica del sistema in spazio di stato $\dot{\boldsymbol{X}} = \boldsymbol{A}\boldsymbol{X} + \boldsymbol{B}\boldsymbol{u}$:

$$\dot{\boldsymbol{X}} = \begin{cases} \dot{\phi} = p \\ \dot{\theta} = q \\ \dot{\psi} = r \\ \ddot{\phi} = \frac{U_2}{J_x} \\ \ddot{\theta} = \frac{U_3}{J_y} \\ \ddot{\psi} = \frac{U_4}{J_z} \\ \ddot{x} = -g\theta \\ \ddot{y} = g\phi \\ \ddot{z} = -\frac{1}{m}U_1 \\ \dot{x} = \dot{x_b} \\ \dot{y} = \dot{y_b} \\ \dot{z} = \dot{z_b} \end{cases} \tag{45}$$

L'equazione (45) rappresenta il sistema linearizzato usato per la sintesi del controllore.

### 2.7.2 Controllabilità e Osservabilità

La controllabilità e l'osservabilità sono due proprietà, l'una il duale dell'altra, fondamentali per un sistema di controllo in quanto ci assicurano di poter comandare il sistema dinamico a nostro piacimento e di poter osservare, a partire dai suoi output, gli stati interni del sistema. Per verificare queste proprietà avremo bisogno delle matrici $\boldsymbol{A}, \boldsymbol{B}$ e $\boldsymbol{C}$, quest'ultima coincide con la matrice identità $I_{12}$. Valutiamo la matrice di controllabilità del sistema. Il sistema lineare a tempo continuo è controllabile se e soltanto se la matrice di controllabilità ha rango pieno.

$$\mathscr{C} = \begin{bmatrix} \boldsymbol{B} & \boldsymbol{A} \cdot \boldsymbol{B} & \boldsymbol{A}^2 \cdot \boldsymbol{B} & \cdots & \boldsymbol{A}^{11} \cdot \boldsymbol{B} \end{bmatrix} \in \mathbb{R}^{12 \times 48} \tag{46}$$

Il sistema lineare a tempo continuo è osservabile, invece, se e soltanto se la matrice di osservabilità ha rango pieno. La matrice di osservabilità è data dalla seguente matrice:

$$\mathscr{O} = \begin{bmatrix} \boldsymbol{C} \\ \boldsymbol{C} \cdot \boldsymbol{A} \\ \boldsymbol{C} \cdot \boldsymbol{A}^2 \\ \vdots \\ \boldsymbol{C} \cdot \boldsymbol{A}^{11} \end{bmatrix} \in \mathbb{R}^{144 \times 12} \tag{47}$$

La verifica delle proprità è stata svolta tramite il software MATLAB. Le matrici hanno rango pieno, dunque, il sistema è sia controllabile che osservabile. Si procederà nel prossimo capitolo alla sinstesi del controllore.

# 3 Sintesi del Controllore

## 3.1 Strategia di Controllo tramite PID

Per sintentizzare un controllore si hanno a disposizione numerose tecniche. Una delle tecniche di controllo più comune consiste nell'utilizzo del controllore PID. Il controllore *Proporzionale-Integrale-Derivativo* è un sistema a retroazione negativa, cioè un sistema in grado di valutare il valore attuale dell'input rispetto ad un valore desiderato e di generare di conseguenza uno sforzo di controllo attuo a correggere l'errore facendolo tendere a zero. Questo genere di controllori è molto popolare in quanto risulta essere intuitivo e facile da implementare. I valori delle tre azioni *Proporzionale-Integrale-Derivativa* possono essere regolati in modo tale da ottenere una diversa reazione del sistema in base al comportamento richiesto.
Il controllo generale del quadricottero viene effettuato tramite l'utilizzo a cascata di diversi loop con controllori PID. Il loop più interno si occupa di controllare le velocità angolari attorno agli assi del quadro di riferimento del corpo rigido. Per controllare le velocità angolari è necessario calcolarne dapprima i valori desiderati. Il secondo loop più esterno, rappresentato dal blocco *PID Assetto*, infatti, si occupa di controllare l'assetto e di inviare il proprio output, ossia le velocità angolari desiderate, al loop più interno. Il controllore dell'assetto, inoltre, può ricevere informazioni sugli angoli desiderati direttamente da un comando remoto o da un terzo loop di controllo. Nel progetto è stata utilizzata quest'ultima soluzione: un terzo loop, il più esterno, dedicato al controllo della posizione. Inoltre, compreso nel loop più esterno, si trova anche un controllore PID dedicato a correggere l'altezza del drone, che riceve in ingresso i valori del sensore di altitudine del drone e produce in output il valore della spinta verticale. Quest'ultimo loop, dunque, controlla il moto traslazionale del drone ricevendo tre input arbitrari in ingresso $(x_{des}, y_{des}, z_{des})$, oltre a quelli retroazionati dal processo.

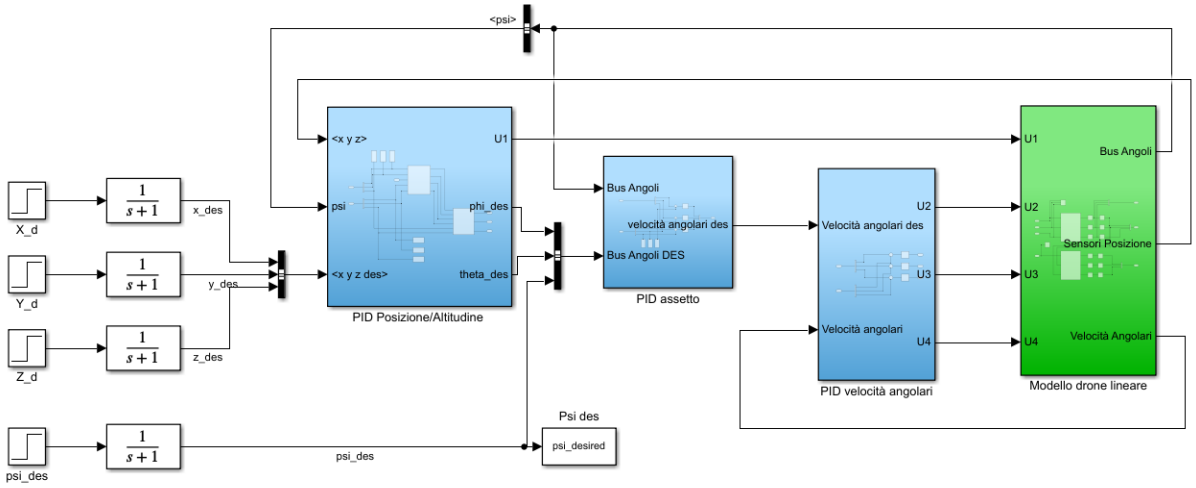


Figure 3: Schema a blocchi del modello lineare.

Nella Figure 3 si può osservare lo schema a blocchi che rappresenta quanto descritto in precedenza, in particolare il modo in cui sono stati organizzati i diversi blocchi. Nello specifico, nell'immagine è stato utilizzato il processo (blocco verde) che implementa le equazioni del modello lineare (maggiori dettagli nel par. 4.6). Lo schema a blocchi del modello non lineare differisce unicamente per il blocco in verde, in cui vengono implementate le equazioni non lineari del modello. I blocchi azzurri rappresentano il

controllore. Gli input $x_{des}, y_{des}, z_{des}$ e $\psi_{des}$ sono stati scelti a priori e possono essere modificati all'interno dello script `drone_quadricottero.m`, di cui si parlerà nel par. 4.2. I valori dei controllori PID sono stati regolati dapprima tramite *auto-tuning* di MATLAB e, in seguito, con piccole rifiniture manuali basate sulla teoria che segue più avanti nel capitolo.
La funzione *autotune* di SIMULINK, iniettando valori di test e misurando la risposta del processo, fornisce una prima stima per i valori $k_P$, $k_I$ e $k_D$, cioè i guadagni delle rispettive azioni del PID. Le tre azioni, infatti, vengono dapprima calcolate separatamente e in seguito vengono sommate.

$$u(t) = u_P(t) + u_I(t) + u_D(t) \tag{48}$$

I tre termini a secondo membro indicano rispettivamente l'azione proporzionale, integrale e derivativa e vengono calcolati come da definizione: per prima si osservi il termine proporzionale.

$$u_P(t) = k_P e(t) \tag{49}$$

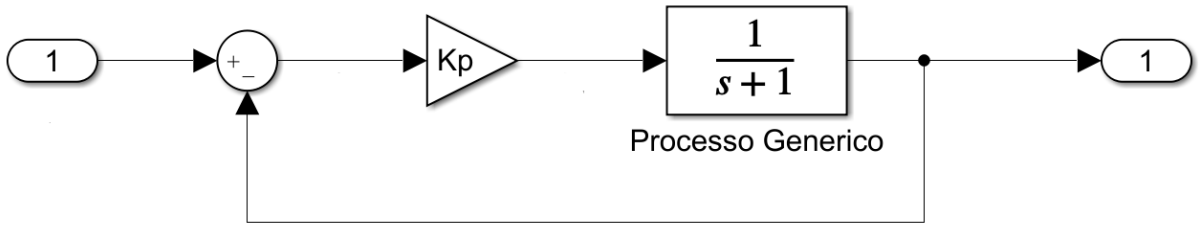


Figure 4: Controllore proporzionale

È chiaro che l'azione proporzionale in sé può stabilizzare soltanto i sistemi più semplici. Osservando la Fig. (4) e guardando l'equazione (49) si può facilmente verificare che maggiore sia $e(t)$, maggiore sarà l'azione di controllo $u_P(t)$ svolta dal controllore. Variando $k_P$ si varia il tempo di salita del sistema e la sovraelongazione, in particolare, aumentando il guadagno, si aumenta il numero di oscillazioni del sistema, rendendolo meno stabile.
Il secondo termine rappresenta l'azione integrale effettuata dal controllore.

$$u_I(t) = k_I \int_{t_0}^{t} e(\tau) d\tau \tag{50}$$

L'azione integrale è particolarmente importante nelle applicazioni, perché assicura un errore nullo a regime permanente per variazioni a gradino del riferimento $r(t)$. Come si può vedere nella fig. (5), l'azione integrale è stata accoppiata con l'azione proporzionale vista poc'anzi. Questo genere di controllori PI viene spesso adoperato nelle applicazioni dove non è richiesto un errore a regime completamente nullo e dove il carico varia lentamente. Qualora il carico dovesse variare in maniera improvvisa, il sistema potrebbe

essere portato all'instabilità, per questo è importante tarare il coefficiente $k_I$ opportunamente, ricordando che un errore $e(\tau)$ positivo porterà ad un incremento dell'integrale, viceversa per un errore $e(\tau)$ negativo. Inoltre, è di fondamentale importanza considerare l'effetto denominato *integral windup*: con la crescita del valore integrale dovuto, ad esempio, ad un gradino in ingresso al PID, il segnale di controllo può portare in saturazione l'attuatore. Ciò si può evitare con l'utilizzo di una delle due tecniche messe a disposizione dalla toolbox di SIMULINK. Per attivare l'anti-windup basta aprire il singolo PID dello schema a blocchi e andando sulla tendina *Output Saturation*, si può selezionare il metodo anti-windup: se si sceglie il metodo *back-calculation* il termine integrale viene scaricato opportunamente; se si sceglie il metodo *clamping*, si attiva la strategia di integrazione condizionata. Nel modello è stata scelta la tecnica *back-calculation*.

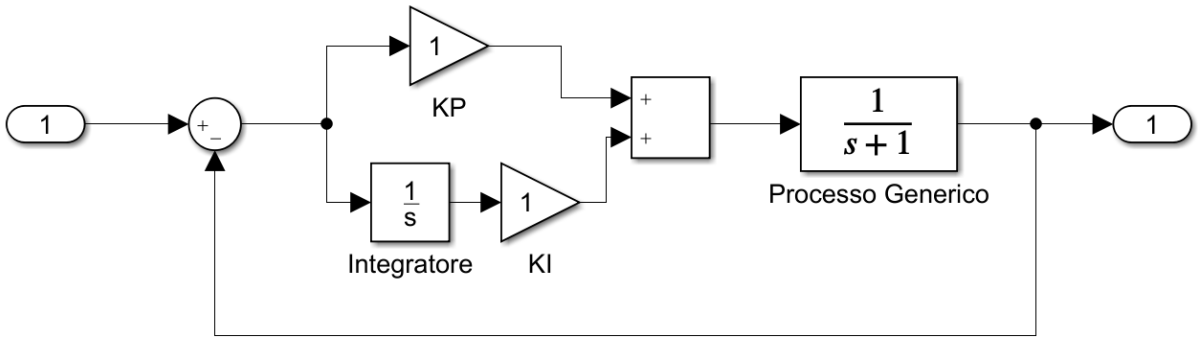


Figure 5: Controllore PI

Il terzo termine nel secondo membro dell'equazione (48) è quello derivativo e fornisce la derivata dell'errore rispetto al tempo.

$$u_D(t) = k_D \frac{\partial e(t)}{\partial t} \tag{51}$$

L'output di un controllore derivativo, o anticipatore, ha la caratteristica di variare con la velocità con cui varia l'errore. L'azione derivativa rischia di rendere un PID troppo sensibile alle variazioni. Come misura di precauzione si è soliti separare l'azione derivativa dal blocco PID per evitare divergenze dell'output del controllore in seguito a brusche variazioni dell'errore $e(t)$: l'azione derivativa riceve in input il valore di $y(t)$ e invia il suo output in un nodo sommatore con l'output del controllore PI. L'azione derivativa è molto importante in quanto garantisce un anticipo di fase di 90 gradi, attenuando eventuali ritardi nell'azione del controllore. Di seguito sono riportate le due possibili configurazioni di un controllore PID, dove l'azione derivativa riceve in input o $e(t)$ oppure $y(t)$.

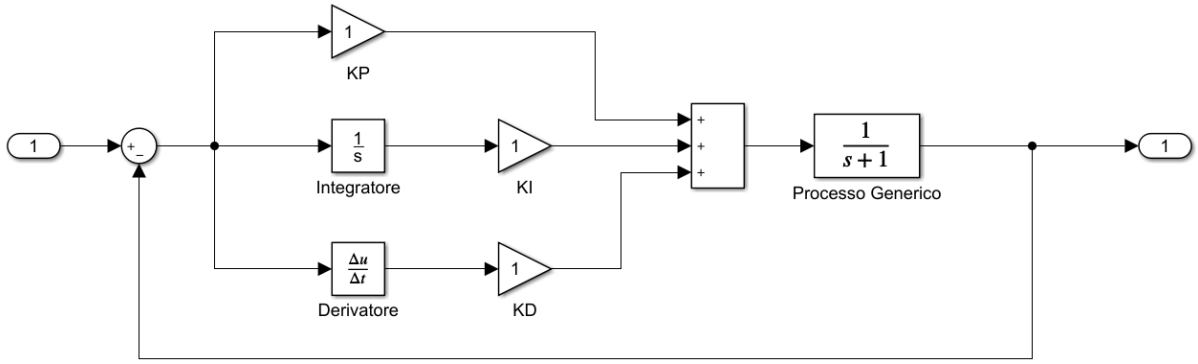


Figure 6: Controllore PID configurazione classicca

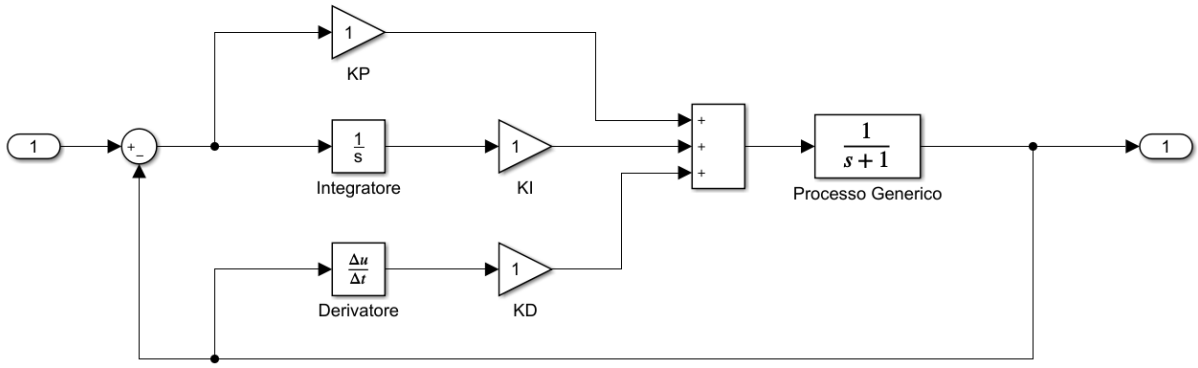


Figure 7: Controllore PID con derivatore separato

Volendo dunque riscrivere l'equazione (48) alla luce di quanto detto, si otterrebbe il seguente risultato:

$$u(t) = k_P e(t) + k_I \int_{t_0}^{t} e(\tau)d\tau + k_D \frac{\partial e(t)}{\partial t} \tag{52}$$

Raccogliendo il termine $k_p$ è possibile mettere in evidenza le costanti di tempo che giocano un ruolo fondamentale nella descrizione delle singole azioni di controllo. Si ottiene la seguente equazione:

$$u(t) = k_P\Big(e(t) + \frac{1}{\tau_I}\int_{t_0}^{t} e(\tau)d\tau + \tau_D \frac{de(t)}{dt}\Big) \tag{53}$$

Dove $\tau_I = \frac{k_P}{k_I}$ e $\tau_D = \frac{k_D}{k_P}$
In base a quanto detto poc'anzi, si può osservare meglio che andando ad aumentare il termine $\tau_I$, il peso del termine integrale si riduce, portando ad un miglioramento della stablitià del sistema al costo di un rallentamento del tempo di risposta; viceversa diminuendo $\tau_I$. Nel caso di $\tau_D$ la stabilità peggiora in entrambi casi, sia diminuendo che

aumentandone il valore. Però, accrescendo il suo valore, il tempo di risposta decrescebbe mentre, diminuendolo, il tempo di risposta si dilaterebbe.
Concluse queste osservazioni, si ha chiaro il funzionamento generale di un controllore PID. Il blocco PID(s), disponibile nella libreria di SIMULINK, permette di saltare l'implementazione manuale delle singole azioni fornendo un controllore pronto per essere regolato. I parametri dei controllori del drone sono stati varati dapprima automaticamente dal software e, in seguito, sulla base delle considerazioni appena espresse, sono stati fatti dei ritocchi ai parametri dei diversi blocchi PID per adattare la dinamica in maniera consona a quella di un drone quadricottero. È stato importante trovare un giusto compromesso tra rapidità di esecuzione e stabilità, giocando molto sull'aggressività del transitorio delle diverse variabili di stato. I macro-PID vengono analizzati nei seguenti paragrafi.

#### 3.1.1 PID Posizione/Altitudine

Il blocco *PID Posizione/Altitudine* nella Figure 3 si occupa di due compiti fondamentali:

- **Correggere l'altitudine attuale**
  Leggendo i sensori di altitudine e confrontandone i valori con quelli desiderati si produce uno sforzo di controllo $U_1$ attuo a correggere la differenza tra i due valori misurati. $U_1$, cioè il primo input di controllo metterà in moto tutte e quattro le eliche del drone per raggiungere l'altitudine desiderata. Non è da ignorare che in questo modo si va ad influenzare, oltre che la spinta verticale, anche il movimento nel piano $xy$ come si vede nel modello non lineare (44), dove il termine $U_1$ compare in ben tre equazioni, quelle per l'accelerazione lineare lungo la $x$, la $y$ e la $z$. Nel modello linearizzato, la spinta verticale $U_1$ è disaccoppiata dal movimento laterale.
- **Correggere la traslazione dovuta a piccoli angoli di assetto**
  Un controllore dell'assetto (analizzato nel prossimo paragrafo) non può evitare di produrre del movimento traslazionale dovuto a piccole inclinazioni lungo gli assi $x$ e $y$ che, combinate con la spinta verticale $U_1$, spostano il drone lungo il piano $xy$. Per evitare che il movimento traslazionale risulti eccessivo, si utilizza un controllore per correggere questi movimenti. Il funzionamento di questo controllore è supportato dai sensori di posizione montati a bordo del drone (viene effettuata una trasformazione dal quadro di riferimento del corpo rigido a quello inerziale prima di valutare il confronto tra i valori misurati e quelli desiderati, che sono definiti nel quadro di riferimento inerziale). Gli output di questo controllore sono i valori desiderati di pitch e roll, che permetteranno di manovrare il drone sul piano $xy$.

#### 3.1.2 PID Assetto

In generale, il blocco *PID Assetto*, presente nella Figure 3, si occupa di confrontare i valori reali di $\phi$, $\theta$ e $\psi$ con quelli desiderati per correggere l'errore, calcolando così, come output, i valori delle velocità angolari desiderate. I valori degli angoi desiderati letti in input vengono per due terzi ($\phi_d$ e $\theta_d$) dal loop più esterno analizzato nel paragrafo precedente. Il terzo input, $\psi_d$, è scelto in maniera arbitraria come orientamento finale del drone al termine della simulazione. L'organizzazione specifica dei blocchi e i loro collegamenti possono essere visualizzati meglio nel capitolo successivo.

#### 3.1.3 PID Velocità Angolari

Il controllore delle velocità angolari riceve in input $\dot{\phi}_d$, $\dot{\theta}_d$ e $\dot{\psi}_d$ per poi fare un confronto con i valori delle velocità angolari misurate dal giroscopio. In questo modo vegono

rispettivamente prodotti i tre input di controllo: $U_2$ per il *rolling moment*, $U_3$ per il *pitching moment* e $U_4$ per lo *yawing moment* che, insieme alla spinta verticale $U1$, possono essere combinati in maniera opportuna per calcolare le velocità da inviare ai motori tramite gli ESC. Per fare ciò basta considerare il sistema dato dall'unione delle equazioni (26) e (33) e ricavare i valori di $\omega_i^2$, che verranno opportunamente trasformati nei relativi valore di tensione applicati ai motori.

$$\omega_1^2 = \frac{U_1}{4k_T} + \frac{U_3}{2lk_T} + \frac{U_4}{4k_d}$$
$$\omega_2^2 = \frac{U_1}{4k_T} - \frac{U_2}{2lk_T} - \frac{U_4}{4k_d}$$
$$\omega_3^2 = \frac{U_1}{4k_T} - \frac{U_3}{2lk_T} + \frac{U_4}{4k_d}$$
$$\omega_4^2 = \frac{U_1}{4k_T} + \frac{U_2}{2lk_T} - \frac{U_4}{4k_d}$$

## 3.2 Applicazione del Controllore

Il controllore è stato implementato e regolato in prima istanza sul modello lineare del paragrafo 2.7.1. Per via del carattere generale del controllore PID e della linearizzazione intorno ad un punto di lavoro generale, è stato possibile applicare, con qualche ulteriore regolazione, lo stesso controllore anche al modello non-lineare ottenendo un risultato piuttosto soddisfacente. Ulteriori analisi verranno effettuate nei capitoli successivi.
Poiché ci si è attenuti esclusivamente all'ambito simulativo, non ci si è preoccupati di parametrizzare al completo i motori dei droni. Ciò si traduce nell'assenza, all'interno dello schema a blocchi, di un limite inferiore e superiore alle velocità dei motori che possono variare da caso a caso. Qualora si volesse applicare il controllore ad un drone reale, bisognerebbe implementare un blocco, prima dell'ingresso dei segnali di controllo nel processo, che programmi le quattro equazioni ricavate poc'anzi e porre dei valori di saturazione alle velocità, in modo da non rischiare di rovinare i motori. Questa premessa non significa, però, che non si è tenuto conto dei valori assunti dagli sforzi di controllo in ingresso al processo: sottostimando la velocità massima di un motore, i parametri dei PID sono stati regolati in modo tale da produrre degli sforzi di controllo che dessero luogo ad una dinamica accettabile.

# 4 Simulazione

## 4.1 Implementazione generale

La simulazione verrà effettuata tramite MATLAB e Simulink, grazie ai quali la programmazione del controllore e del processo diventa molto semplice. Inoltre, sono presenti blocchi come lo *step* e lo *scope* che consentono di regolare e analizzare i diversi parametri. Le simulazioni, contenute nei file `schema_blocchi.slx` e `schema_blocchi_NL.slx`, sono lanciate dallo script di MATLAB `drone_quadricottero.m`. I risultati della simulazione vengono stampati a schermo, basta attendere che il software completi la simulazione la cui durata è impostata a 20.0 secondi simulati.
Il modello completo è quello rappresentato in Figure 3. Ora si procederà mostrando il contenuto dello script e dei singoli blocchi.

## 4.2 Script MATLAB

Lo script `drone_quadricottero.m` è molto semplice, il suo scopo consiste nel comunicare con Simulink affinché lanci le simulazioni e ne metta a grafico il risultato.
Le prime righe vengono utilizzate per pulire la Command Window e il Workspace, subito dopo vengono inizializzati i parametri del drone e le coordinate desiderate.

```
%% Inizializzazione Workspace
clear all
clc

%% Inizializzazione variabili
g = 9.81;      % Acc. gravitazionale (m/s^2)
m = 1.4;       % Massa quadricottero (kg)

Jx = .05;        % Momento di inerzia lungo l'asse X (kg-m^2)
Jy = .05;        % Momento di inerzia lungo l'asse Y (kg-m^2)
Jz = .24;        % Momento di inerzia lungo l'asse Z (kg-m^2)

X_des_GF = .5;          % Valore desiderato di X nel frame inerziale (metri)
Y_des_GF = 1;          % Valore desiderato di X nel frame inerziale (metri)
Z_des_GF = 2;          % Valore desiderato di X nel frame inerziale (metri)
psi_des = pi/6;        % Valore di psi desiderato (radianti)
```

Figure 8: Inizializzazione del Workspace e delle variabili

Vi è poi una sezione in cui vengono lanciate entrambe le simulazioni, dapprima quella lineare che salverà gli output all'interno dell'oggetto `SimOutLIN` e, in seguito, quella non lineare che andrà a salvare gli output nell'oggetto `SimOutNL`.

```
%% Lancio Simulazione (Lineare)
SimOutLIN = sim('schema_blocchi');

%% Lancio Simulazione (Non Lineare)
SimOutNL = sim('schema_blocchi_NL');
```

Figure 9: Lancio di entrambe le simulazioni

La parte finale dello script è infine dedicata alla rappresentazione grafica della simulazione

attraverso i dati contenuti negli oggetti `schema_blocchi.slx` e `schema_blocchi_NL.slx`. Viene dapprima presentato un grafico tridimensionale per l'andamento nello spazio e, successivamente, dei grafici bidimensionali per l'andamento della $x$, della $y$, della $z$ e di $\psi$.

```
25      % Grafico 3D
26 -    figure()
27 -    plot3(SimOutLIN.X_desired.signals.values,SimOutLIN.Y_desired.signals.values,SimOutLIN.Z_desired.signals.values) % Traiettoria desiderata
28 -    hold on
29 -    plot3(SimOutLIN.X_out.signals.values,SimOutLIN.Y_out.signals.values,SimOutLIN.Z_out.signals.values)            % Traiettoria lineare
30 -    hold on
31 -    plot3(SimOutNL.X_out.signals.values,SimOutNL.Y_out.signals.values,SimOutNL.Z_out.signals.values)               % Traiettoria non lineare
32 -    grid on
33 -    xlabel('X')
34 -    ylabel('Y')
35 -    zlabel('Z')
36 -    title('3D')
37
```

```
38      % Grafici 2D
39 -    figure()
40 -    plot(SimOutLIN.tout,SimOutLIN.X_desired.signals.values)         % Traiettoria desiderata
41 -    hold on
42 -    plot(SimOutLIN.tout,SimOutLIN.X_out.signals.values)             % Traiettoria lineare
43 -    hold on
44 -    plot(SimOutNL.tout,SimOutNL.X_out.signals.values)               % Traiettoria non lineare
45 -    title('X')
46
47 -    figure()
48 -    plot(SimOutLIN.tout,SimOutLIN.Y_desired.signals.values)         % Traiettoria desiderata
49 -    hold on
50 -    plot(SimOutLIN.tout,SimOutLIN.Y_out.signals.values)             % Traiettoria lineare
51 -    hold on
52 -    plot(SimOutNL.tout,SimOutNL.Y_out.signals.values)               % Traiettoria non lineare
53 -    title('Y')
54
55 -    figure()
56 -    plot(SimOutLIN.tout,SimOutLIN.Z_desired.signals.values)         % Traiettoria desiderata
57 -    hold on
58 -    plot(SimOutLIN.tout,SimOutLIN.Z_out.signals.values)             % Traiettoria lineare
59 -    hold on
60 -    plot(SimOutNL.tout,SimOutNL.Z_out.signals.values)               % Traiettoria non lineare
61 -    title('Z')
62
```

```
63 -    figure()
64 -    plot(SimOutLIN.tout,SimOutLIN.psi_desired.signals.values)         % Traiettoria desiderata
65 -    hold on
66 -    plot(SimOutLIN.tout,SimOutLIN.psi_out.signals.values)             % Traiettoria lineare
67 -    hold on
68 -    plot(SimOutNL.tout,SimOutNL.psi_out.signals.values)               % Traiettoria non lineare
69 -    title('PSI')
70
```

Figure 10: Codici utilizzati per graficare i risultati della simulazione

Il codice termina alla riga 69 dopo aver lanciato i grafici che verranno analizzati nel capitolo 5. Nel paragrafo successivo ci si occuperà di illustrare il codice a blocchi tramite il quale vengono calcolati i dati della simulazione.

## 4.3 Implementazione del PID Posizione/Altitudine

Dopo aver aperto lo schema a blocchi nella Figure 3, eseguendo un doppio-click sul blocco *PID Posizione/Altitudine*, descritto nel par. 3.1.1, ci troveremo davanti alla schermata

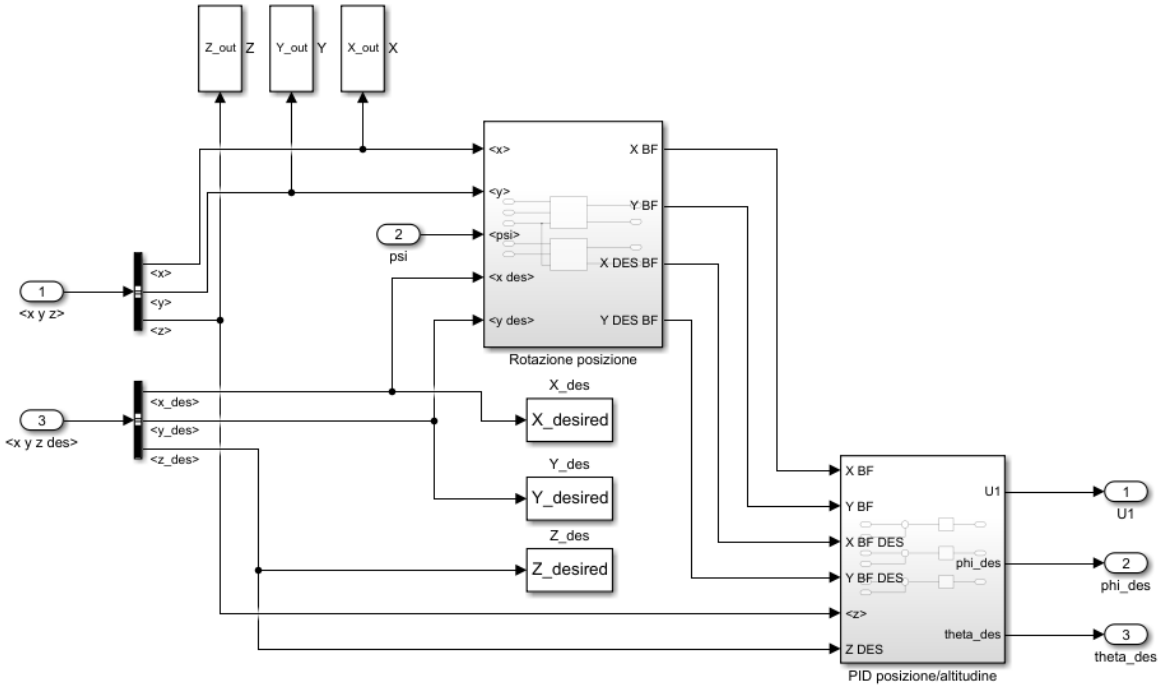


Figure 11: Schema a blocchi del controllore PID descritto nel par. 3.1.1.

rappresentata nella Figure 11. Si osservano due blocchi principali, *Rotazione Posizione* e *PID Posizione/Altitudine*. Il primo, tramite la matrice di rotazione attorno all'asse **z**, si occupa di convertire i valori della posizione nel piano $xy$ dal quadro di riferimento inerziale al quadro di riferimento del corpo rigido. Cliccando sul blocco *Rotazione Posizione* ci si trova davanti a quanto rappresentato nella Fig. (12). Osservando l'immagine, risaltano due blocchi: entrambi implementano la matrice di rotazione attorno all'asse **z**.

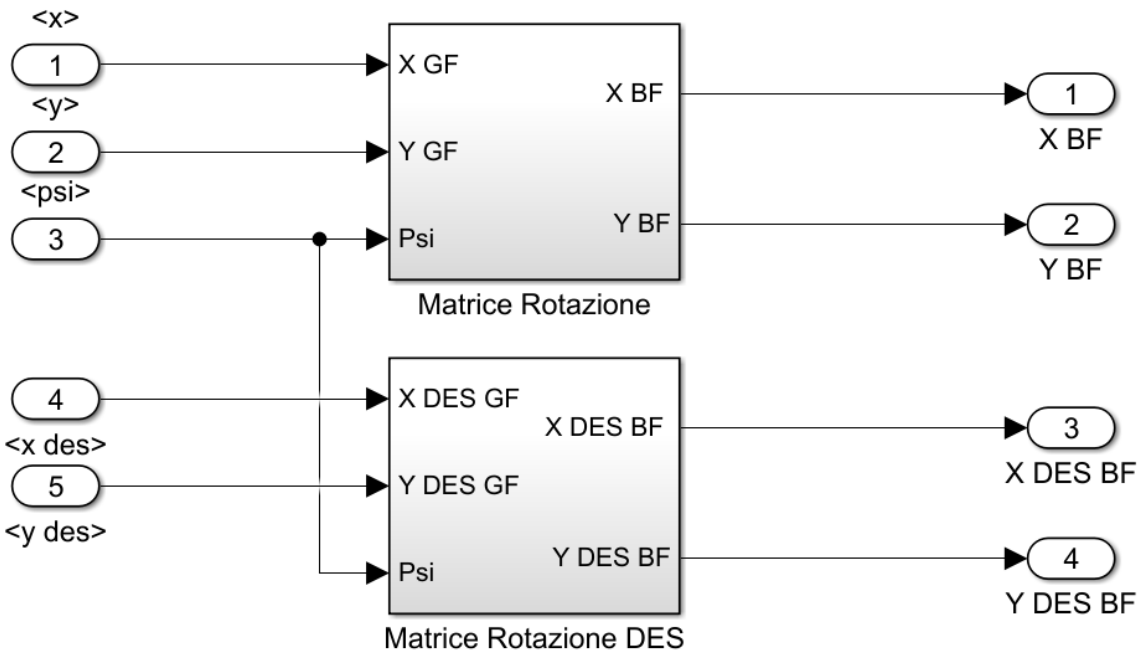


Figure 12: Interno del blocco Rotazione Posizione

Il blocco più in alto, *Matrice Rotazione* effettua la rotazione per i valori rilevati dai

sensori; il secondo blocco, *Matrice Rotazione DES*, è implementato allo stesso modo del blocco superiore ma, per ragioni di nomenclatura, si è dovuto scegliere un nome che non coincidesse. Ovviamente anche quest'ultimo blocco si occupa di effettuare la trasformazione, questa volta però, dei valori *desiderati* lungo il piano. Aprendo uno qualsiasi tra i due blocchi, si può osservare l'implementazione della matrice.

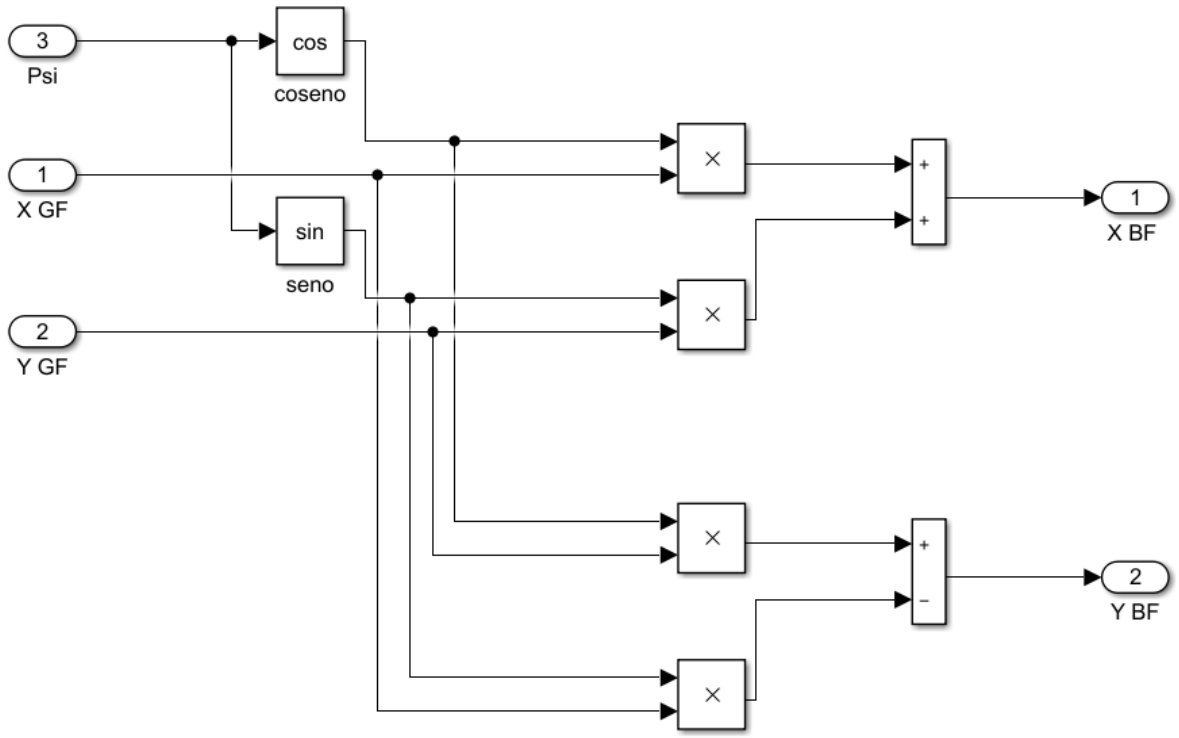


Figure 13: Interno del blocco Matrice Rotazione

I valori trasformati vengono poi mandati al blocco successivo, *PID Posizione/Altitudine*, che implementa, tramite gli appositi blocchetti PID, i controllori per la $x$, la $y$ e la $z$, generando rispettivamente $\theta_d$, $\phi_d$ e $U_1$ come si può osservare nella Fig. (14). Sono presenti, inoltre, dei blocchi *To Workspace* per comunicare i valori della posizione attuale in continuo aggiornamento e della posizione desiderata al workspace di MATLAB, all'interno dei relativi oggetti.

### 4.4 Implementazione del PID Assetto

Cliccando sul blocco denominato *PID Assetto*, descritto nel par. 3.1.2, la struttura al suo interno si rivela essere quella della Fig. (15). Gli input del blocco, anche in questo caso, sono due segnali bus. Il *Bus Angoli* contiene i valori dei sensori a bordo del drone, mentre il *Bus Angoli DES* racchiude i valori degli angoli desiderati, tra cui $\theta_d$, $\phi_d$ ricevuti dall'output del *PID Posizione/Altitudine*. Si è supposto, invece, per il valore $\psi_d$, che esso venga in input direttamente dal radiocomando. Infatti, nella Figure 8, si può notare come questo valore non è frutto di alcun cacolo, bensì di un valore arbitrario (in radianti) attribuito alla variabile `psi_des`. Una volta accoppiati opportunamente gli angoli desiderati con quelli reali, si elabora un rispettivo errore che verrà dato in input ai PID che si occupano di correggere l'apposito angolo. Vengono quindi prodotti, in output, e inviati all'interno di un bus, le velocità angolari desiderate.

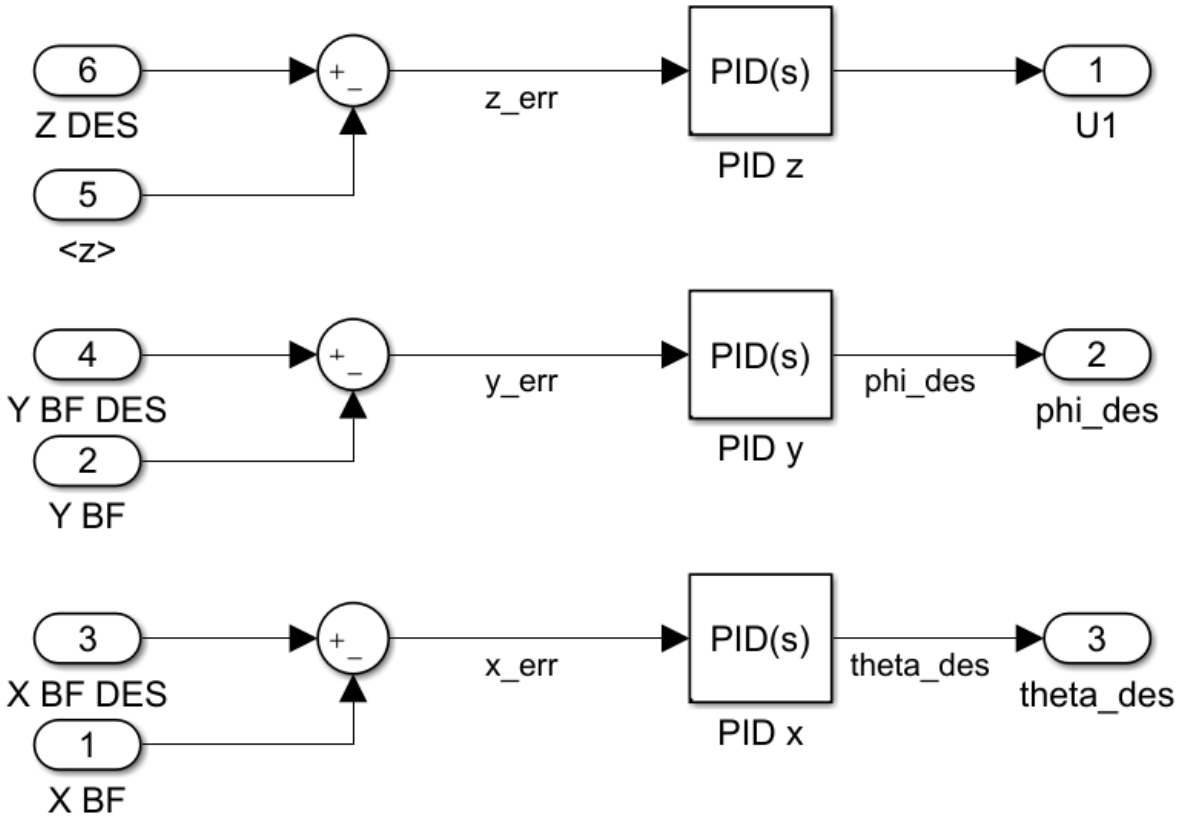


Figure 14: PID della posizione e dell'altitudine

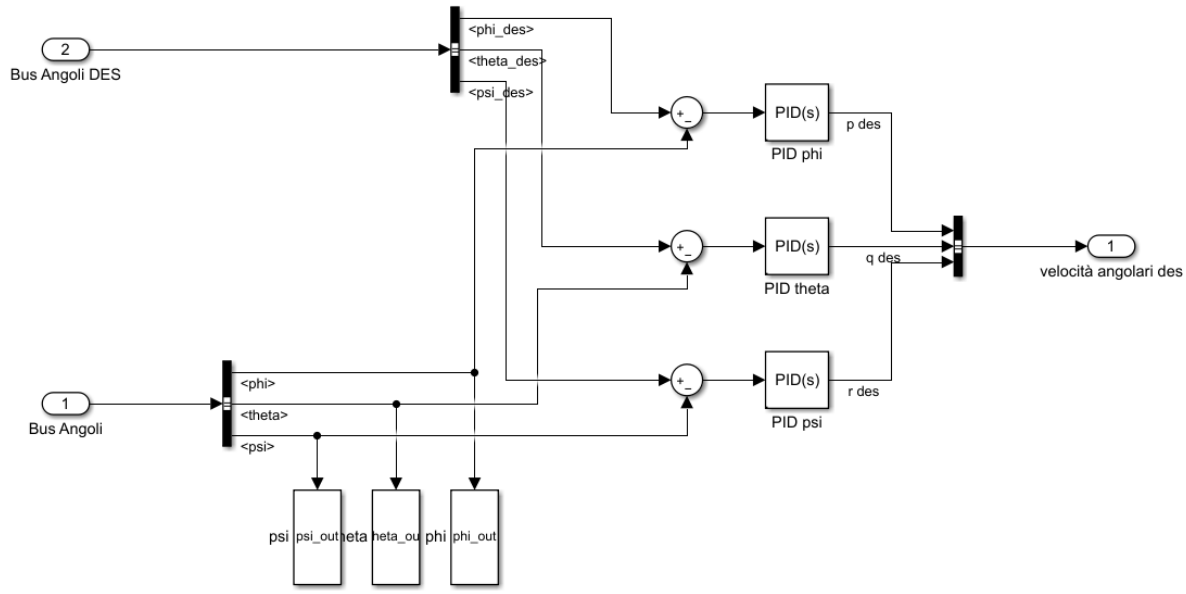


Figure 15: Schema a blocchi del controllore PID descritto nel par. 3.1.2.

## 4.5 Implementazione del PID Velocità Angolari

Come si può vedere nella Figure 16, il PID dedicato al controllo delle velocità angolari, ricalca lo schema logico del *PID Assetto*. Gli ingressi sono ancora due bus, uno per i valori desiderati (cioè l'output del blocco *PID Assetto*) e l'altro per i valori provenienti dai sensori a bordo del drone. Dopo aver mandato le rispettive coppie di valori nel

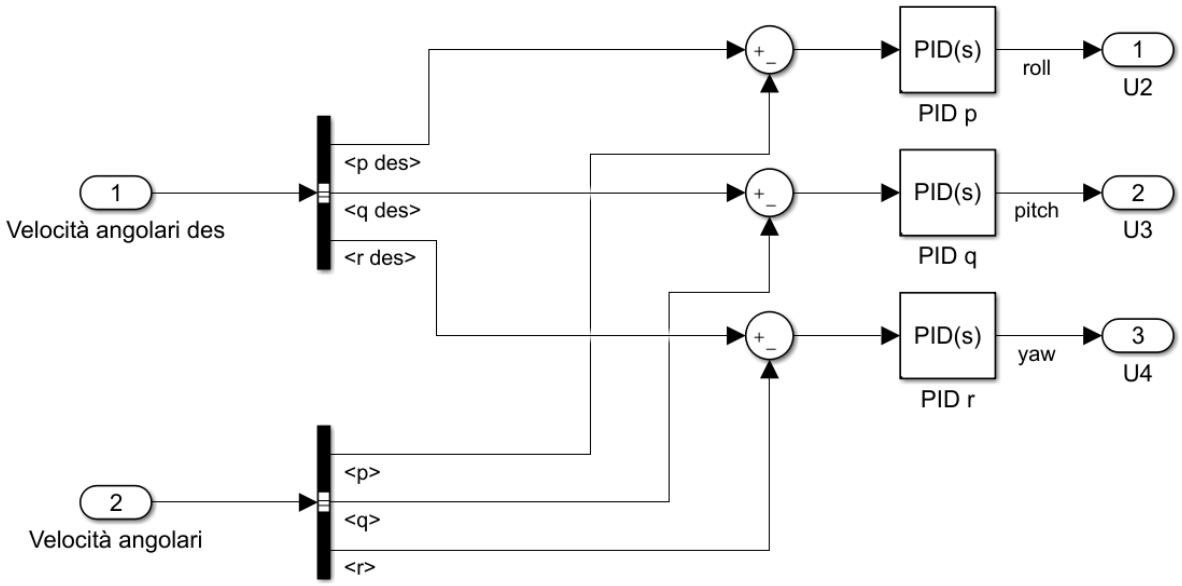


Figure 16: Schema a blocchi del controllore PID descritto nel par. 3.1.3.

nodo sommatore, l'errore calcolato va in input ai relativi PID. Gli output dei controllori corrispondono dunque agli sforzi di controllo restanti, $U_2$, $U_3$ e $U_4$ che, insieme a $U_1$, calcolato all'inizio della cascata dal PID relativo alla $z$, vengono dati in input al processo.

## 4.6 Implementazione del Modello Lineare

Nello schema riportato nella Figure 17, si può osservare l'implementazione generale delle equazioni lineari ricavate nel paragrafo 2.7.1. Vengono utilizzati degli *scope* per osservare l'andamento degli sforzi di controllo in input.
Analizzando il blocco *Dinamica Rotazionale* all'interno della Fig. (18), si può notare come vengono implementate le equazioni (36). Gli output del blocco *Dinamica Rotazionale* vengono mandati attraverso due terne di integratori in serie per ottenere, in ordine, $p$, $q$ ed $r$ e in seguito, dopo l'ultimo integratore, $\phi$, $\theta$ e $\psi$. La prima terna rappresenta, come si può osservare nella Fig. (17), il primo bus di output con le velocità angolari; la seconda terna rappresenta il secondo bus di output con gli angoli del drone. In particolare, $\phi$ e $\theta$ tornano utili all'interno del blocco *Dinamica Traslazionale* insieme all'input di controllo $U_1$. Si può osservarne il contenuto nella Fig. (19). Gli output del suddetto blocco vengono mandati attraverso un'altra coppia di terne di integratori in serie, ottenendo l'ultimo bus di output con le posizioni inerziali.

## 4.7 Implementazione del Modello Non Lineare

Nella Figure 20, si può osservare lo schema, certamente più complesso, del modello non lineare che implementa le equazioni del paragrafo 2.6. All'interno dello schema troviamo ancora il blocco *Dinamica Rotazionale*, questa volta però, il suo interno risulta più labirintico per via della maggiore complessità della equazioni (36). Infatti, facendo doppio click sul blocco ci si trova lo schema in Fig. (21). Il prodotto della Dinamica Rotazionale viene doppiamente integrato per poi essere mandato in input nel blocco *Dinamica Traslazionale*; inoltre, tra la prima e la seconda terna di integratori i valori $\dot{phi}$, $\dot{\theta}$ e $\dot{\psi}$ vengono inviati al blocco *Cinematica Rotazionale* per calcolare, insieme a $\phi$ e $\theta$, le velocità angolari come descritto nell'equazione (10).

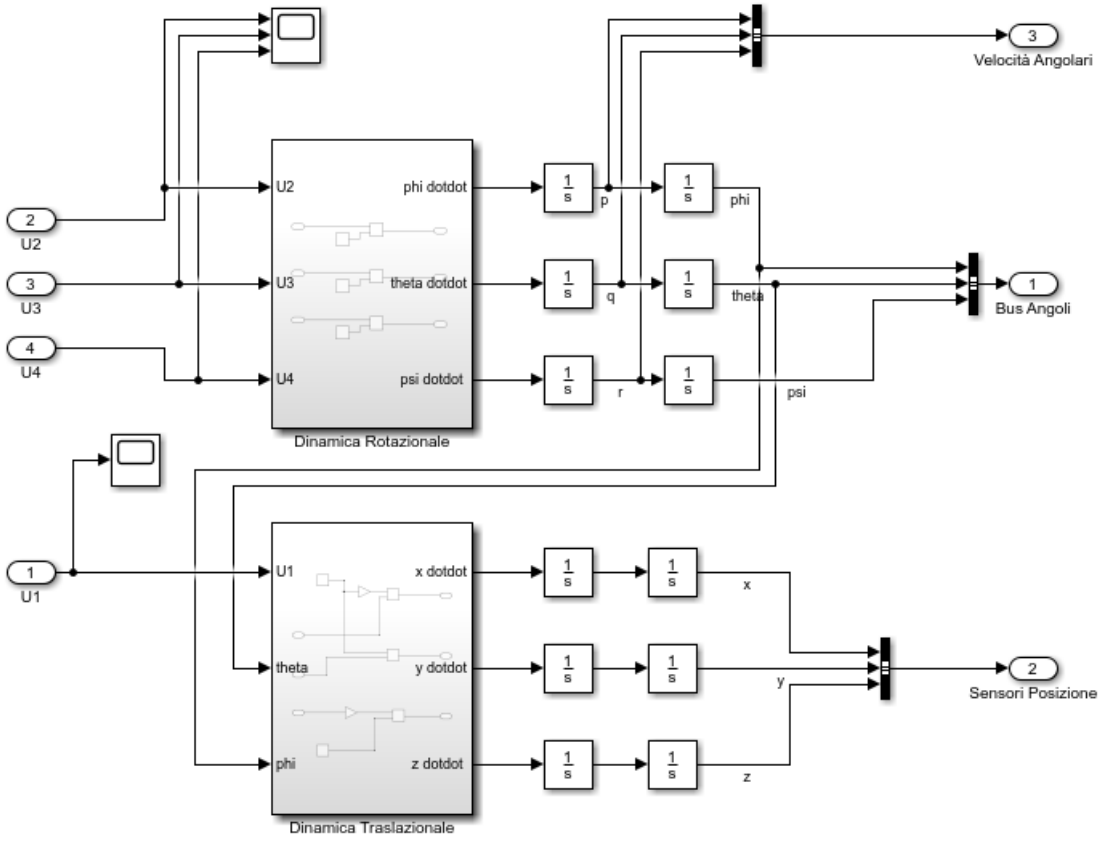


Figure 17: Implementazione del modello linearizzaato

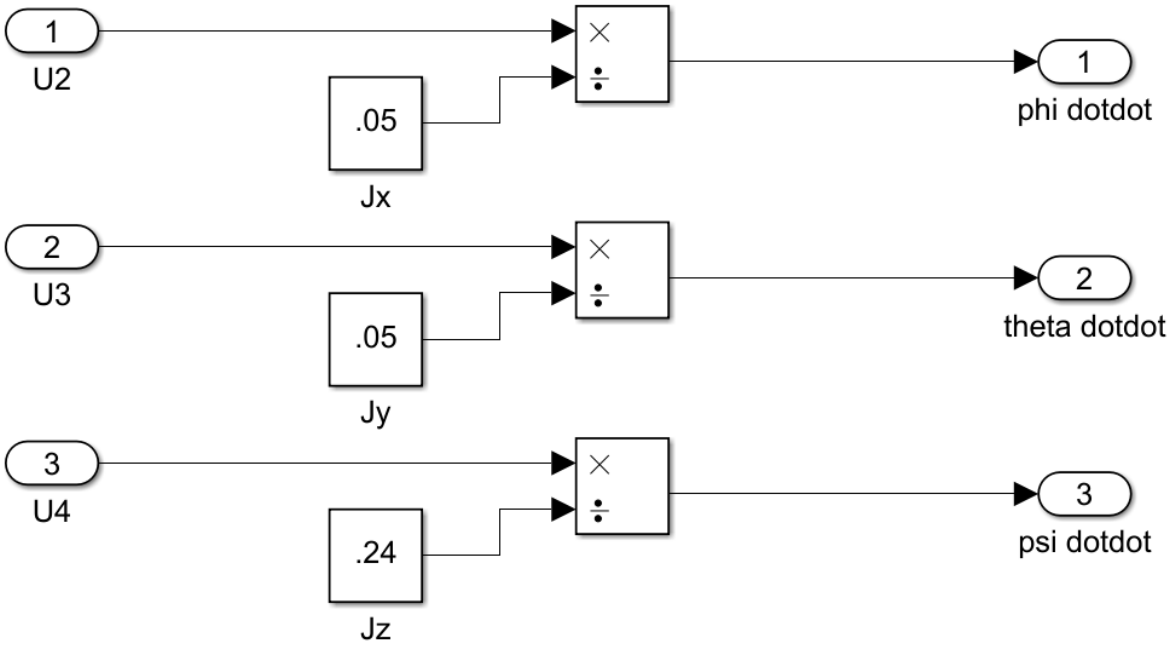


Figure 18: Implementazione della Dinamica Rotazionale lineare

Analizzando dapprima il blocco *Cinematica Rotazionale*, si può osservare l'implementazione della matrice $\boldsymbol{S}$. Gli output del blocco forniscono le velocità angolari aggiornate che, oltre a finire nel bus che le porterà al loop di controllo più interno, saranno anche utili al calcolo

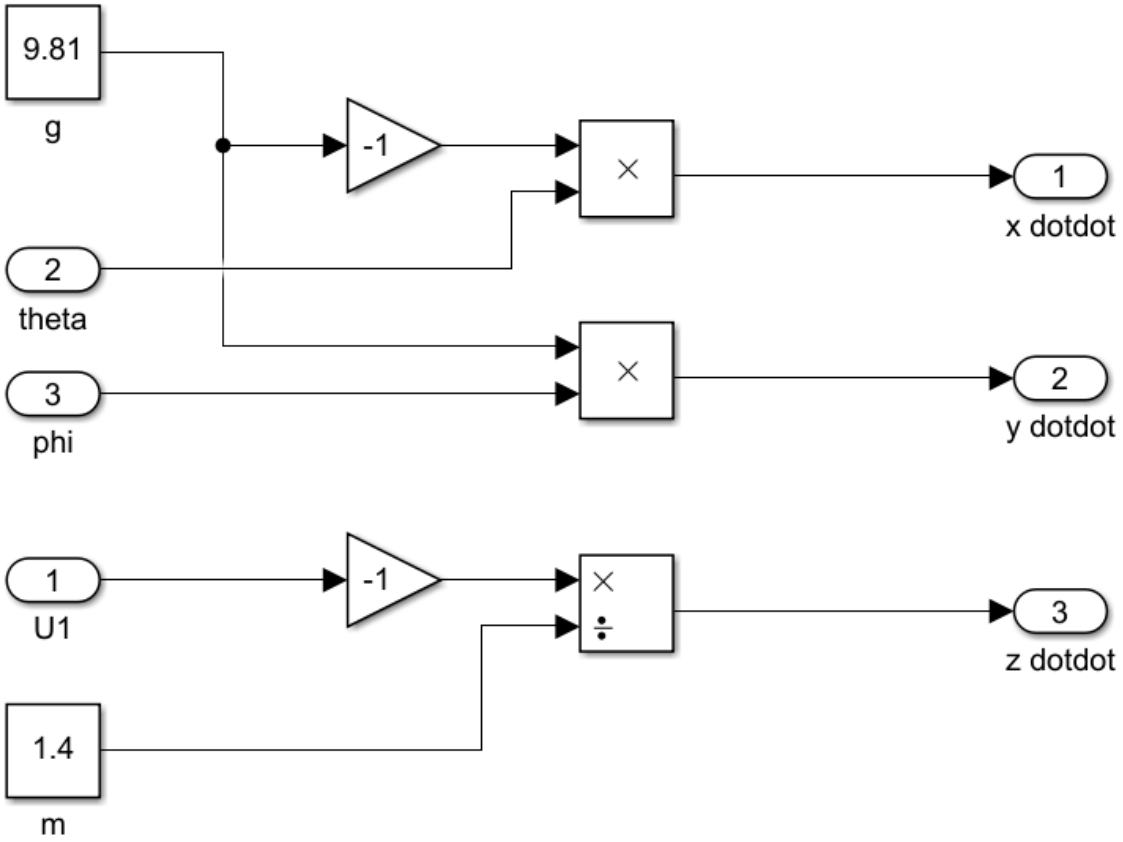


Figure 19: Implementazione della Dinamica Traslazionale lineare

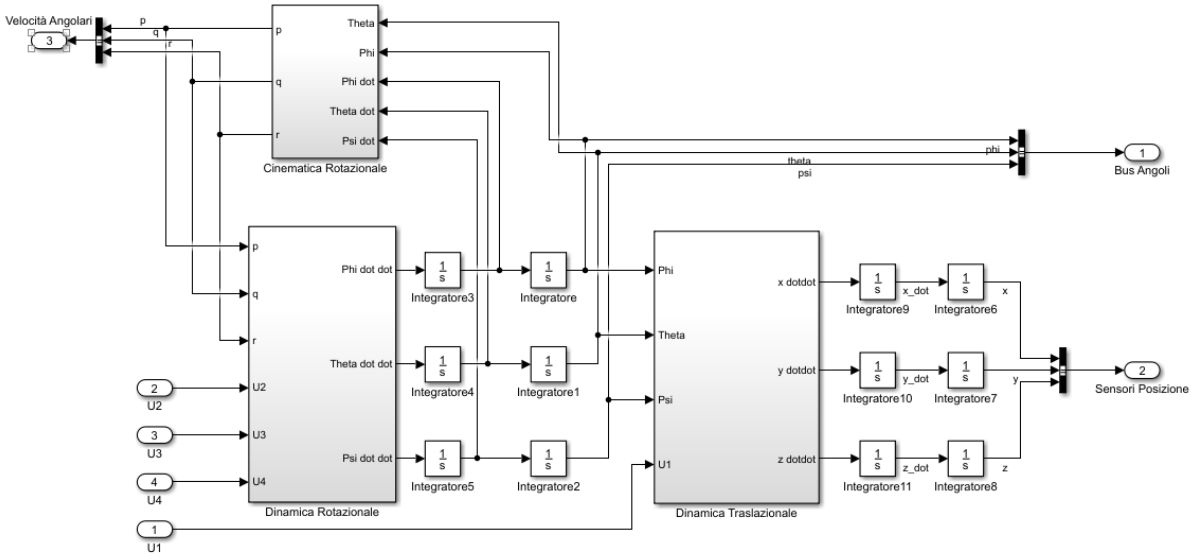


Figure 20: Implementazione del modello non lineare

dei valori della dinamica Rotazionale.
Gli input del blocco *Dinamica Traslazionale* sono costituiti dagli output degli integratori di cui si è accennato sopra e dalla spinta verticale $U_1$. Gli input vengono combinati secondo l'equazione (25) come illustrato nella Fig. (22). Gli output, invece, dopo essere stati doppiamente integrati, vengono raccolti nel bus che trasporta le informazioni sulla posizione inerziale.

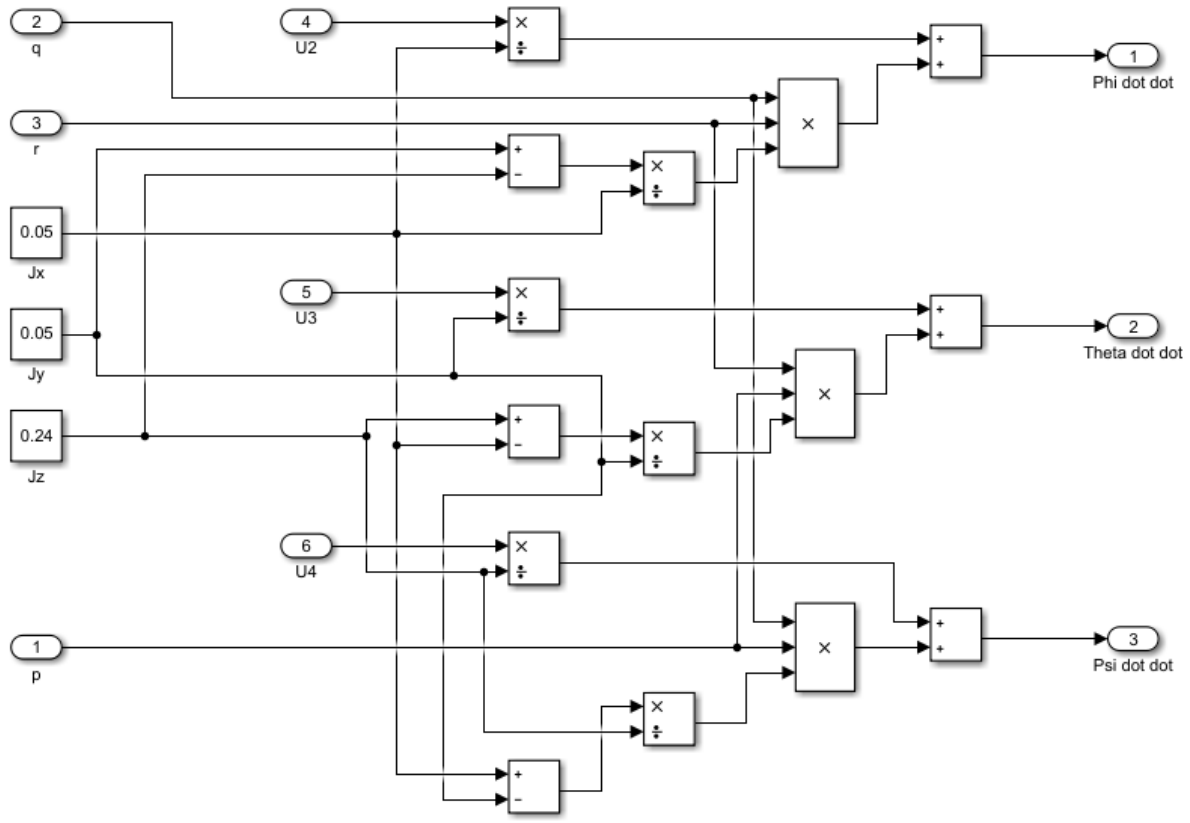


Figure 21: Implementazione della Dinamica Rotazionale non lineare

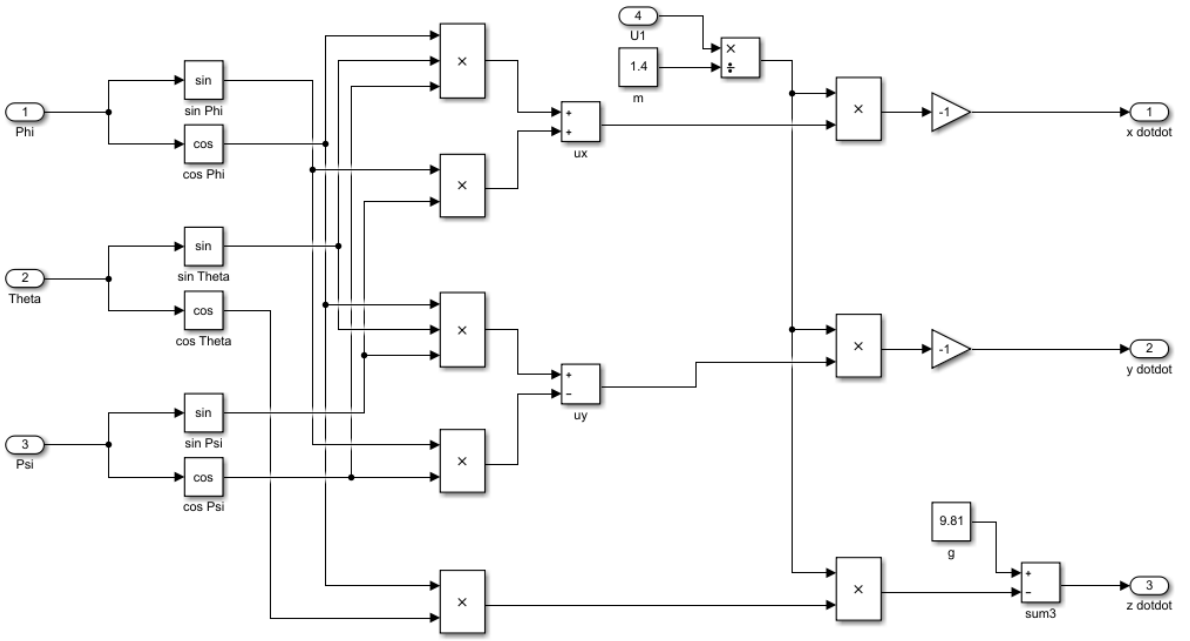


Figure 22: Implementazione della Dinamica Traslazionale non lineare

## 4.8 Risultati simulazione

Il drone, arrivati a questo punto, è pronto per essere simulato. I valori delle variabili desiderate in input, utilizzate durante i test, sono i seguenti: $x_d = 0.5$; $y_d = 1$; $z_d = 2$; $\psi_d = \frac{\pi}{6}$. Questi valori possono essere variati a piacimento, ma è preferibile, per evitare

sforzi di controllo eccessivi che potrebbero far divergere numericamente la simulazione, dare in input valori vicini al punto di partenza. Difatti, se si volesse far seguire al drone una traiettoria in maniera automatica, basterebbe dare in input sequenze di punti non eccessivamente distanti tra di loro a intervalli di tempo regolari.
Nella Table 2, si possono osservare in maniera accurata gli andamenti del drone, sia in modo bidimensionale che tridimensionale. La linea blu indica la traiettoria ideale da seguire (gradino filtrato), mentre quella rossa indica la traiettoria lineare e quella gialla la traiettoria non lineare. Per rendere in maniera più efficiente il grafico 3D sono state dedicate due immagini da due punti di vista differenti.

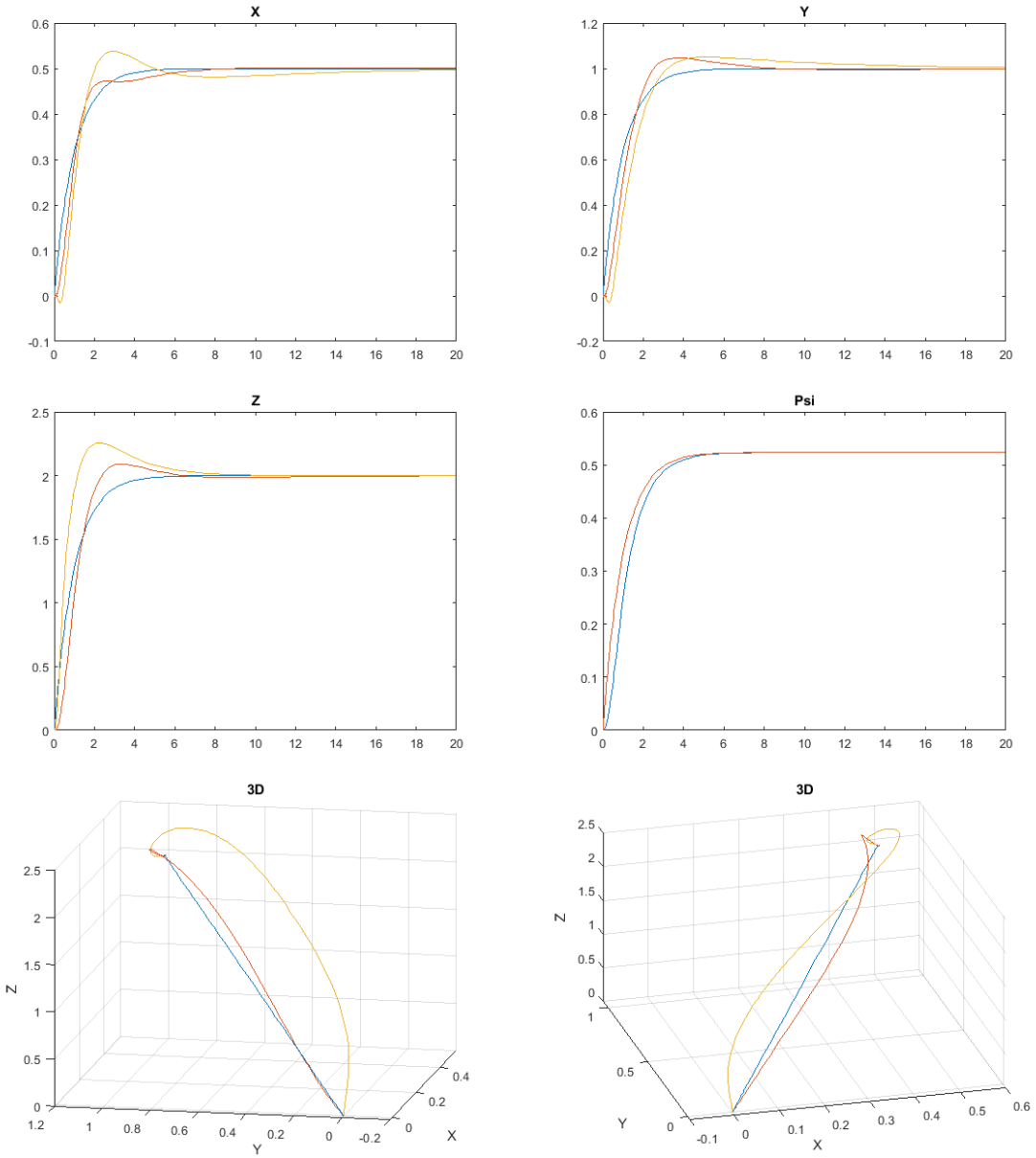


Table 2: Grafici dell'andamento a confronto

# 5 Conclusioni

Come si può osservare dai grafici della Table 2, il controllore sintetizzato sul modello lineare funziona e svolge un buon lavoro, anche applicandolo al modello non lineare. Il comportamento del modello non linearizzato del drone, qualora non si fossero trascurati i vari attriti e gli effetti aerodinamici, utilizzando lo stesso controllore, sarebbe stato leggermente meno performante. Per quanto riguarda il controllore applicato al modello non lineare, l'unico ritocco è stato fatto sul PID della $z$: poiché l'input $U_1$ che regola la spinta verticale, a differenza del modello linearizzato, compare anche nelle equazioni che descrivono l'accelerazione lineare lungo gli assi $\mathbf{x}$ e $\mathbf{y}$, è stato necessario modificare i parametri del PID affinché attuasse un'azione di controllo che tenesse conto di questo fattore, riducendo l'aggressività dell'azione di controllo nel transitorio, permettendo di evitare eccessive sovraelongazioni. In generale, però, è stato possibile regolare i PID in modo tale da ottenere delle performance decisamente soddisfacenti.
Per quanto riguarda i futuri sviluppi, il modello potrà essere ampliato aggiungendo le varie forze ed attriti che sono stati volutamente trascurati nella modellazione, con relative modifiche ai parametri dei controllori per permettere un volo performante. Per quanto riguarda l'aspetto della simulazione, uno spunto interessante consisterebbe nel programmare una sequenza di punti da far seguire al drone, tracciando in questo modo una traiettoria più complessa volta ad evitare eventuali ostacoli. Sarebbe interessante, inoltre, applicare il controllore ad un drone reale, osservarne le prestazioni e modificarlo di conseguenza, magari limitando le velocità raggiungibili in base ai motori montati.

# References

Bouabdallah, S. (2007). *Design and control of quadrotors with application to autonomous flying* (Tech. Rep.). Epfl.

CH-Robotics. (n.d.). *Understanding euler angles.* Retrieved from `http://www.chrobotics.com/library/understanding-euler-angles`

Gibiansky, A. (2012). Quadcopter dynamics, simulation, and control. *Andrew. gibiansky. com*.

Sabatino, F. (2015). *Quadrotor control: modeling, nonlinearcontrol design, and simulation.*

Selby, W. (n.d.). *System modeling.* Retrieved from `https://www.wilselby.com/research/arducopter/modeling/`

Tyler, C. (n.d.). *Modeling vehicle dynamics – euler angles.* Retrieved from `https://charlestytler.com/modeling-vehicle-dynamics-euler-angles/`