Università Politecnica delle Marche

FACOLTÀ DI INGEGNERIA
Corso di Laurea Triennale in Ingegneria Informatica e dell'Automazione

Tesi di laurea triennale

Studio e sviluppo di tecniche di controllo per l'assetto di droni

Study and development of attitude control techniques for drones

Candidato:
Roberto Bramucci
Matricola 1087030

Relatore:
Gianluca Ippoliti

Correlatore:
Giuseppe Orlando

Anno Accademico 2020-2021

Sommario

Il presente elaborato si pone l'obiettivo di progettare controllori per l'assetto, in particolare per il rollio, del mini drone Parrot Mambo.
In primo luogo si sviluppa il modello che descrive il comportamento fisico del drone, il quale viene poi analizzato nel progetto Simulink *asbQuadcopter*. Sulla base del modello fornito da Simulink si procede all'implementazione di tre controllori, usando le tecniche di sintesi tramite luogo delle radici e in frequenza, che possono sostituire il sistema di controllo predefinito, presente nel modello originale. Infine vengono confrontate le prestazioni dei suddetti controllori, facendo riferimento ai dati ottenuti dalle simulazioni sul drone reale, rispetto a diversi tipi di volo.

Indice

Elenco delle figure

1 Quadricotteri

I quadricotteri sono un'invenzione innovativa nel campo degli aeromobili a pilotaggio remoto, noti con la sigla *UAV*. Grazie alle ridotte dimensioni, al basso costo e alla semplice manutenzione, i quadricotteri sono ampiamente utilizzati per scopi molto diversi, come l'intrattenimento, le attività di ricerca e soccorso, le spedizioni commerciali e le attività militari, e in generale in tutti i campi in cui sono richiesti compiti standardizzati, dispendiosi in termini di tempo o pericolosi, e richiedono funzionalità particolari per ogni settore in cui sono impiegati.
Un quadricottero è un aerogiro sollevato e spinto da quattro rotori, ognuno dotato di un'elica, di cui due ruotano in senso orario e i restanti due in senso antiorario, consentendo al velivolo di librarsi e mantenere stabilità. I quadricotteri si svilupparono durante gli anni '20 del Novecento, grazie all'espansione degli studi sul controllo simultaneo di quattro motori entro una larghezza di banda limitata. Successivamente, nel 1939, all'inizio della Seconda guerra mondiale, venne prodotto negli Stati Uniti il primo velivolo a controllo remoto, noto come *Radioplane OQ-2*. Il progresso di questa tecnologia è continuato durante la Guerra Fredda, consentendo di ridurre progressivamente le dimensioni dei dispositivi. Negli ultimi vent'anni sono state svolte numerose ricerche sulla progettazione di quadricotteri, sul controllo dei rotori e sulla stabilità di volo. Le dimensioni sempre più compatte e la possibilità di manovrare facilmente questi dispositivi hanno permesso una sempre maggiore disponibilità sul mercato.
I moderni quadricotteri presentano diverse caratteristiche, come telecamere di bordo, visione artificiale per tracciare un oggetto in movimento, controllo tramite Wi-Fi o Bluetooth e localizzazione tramite GPS.

1.1 Parrot Mambo

In questa tesi si è deciso di prendere in considerazione il mini drone *Mambo*, prodotto dall'azienda francese Parrot. Questa scelta è dovuta al fatto che il software Simulink, un ambiente a blocchi che consente di modellare, simulare e analizzare sistemi dinamici, integrato in Matlab, fornisce un toolbox denominato *Simulink Support Package for Parrot Minidrones*, che, insieme ad *Aerospace Blockset*, consente di utilizzare un modello già implementato, relativo a tale mini drone. A partire da questo modello è possibile creare un progetto in cui si inseriscono i propri controllori, come verrà esposto di seguito per quanto riguarda il controllo del *rollio*.

Figura 1: Parrot Mambo.

Prima di analizzare in dettaglio il modello sopraccitato, si presentano le caratteristiche tecniche del drone Mambo.

- Il mini drone presenta un'unità di misura inerziale IMU per valutare la velocità, l'inclinazione e il contatto con gli ostacoli, caratterizzata da un accelerometro a 3 assi e un giroscopio a 3 assi. Inoltre, nella parte inferiore del drone, è inserito un sensore a ultrasuoni che ha lo scopo di calcolare la distanza tra le superfici circostanti e il dispositivo, mentre per quanto riguarda la stabilizzazione orizzontale utilizza un sensore telecamera, posto accanto al sensore a ultrasuoni. Infine il drone possiede anche un sensore di pressione, utilizzato per misurare l'altitudine.
  La scheda madre montata è di tipo SIP6 con processore ARM A9 - 800 MHz e utilizza Linux come sistema operativo.

- I quattro motori sono brushed coreless: la caratteristica che contraddistingue i motori coreless rispetto a quelli di tipo tradizionale, che prevedono un avvolgimento di filo di rame intorno a un nucleo di ferro, è l'assenza del nucleo in ferro nel rotore: l'avvolgimento di rame è realizzato in una configurazione a cilindro e ruota esternamente a un magnete anch'esso cilindrico. Ciò offre molteplici vantaggi, tra i quali quello di avere un motore più leggero, caratterizzato da un minore momento d'inerzia, consentendo una più rapida accelerazione e decelerazione e un'elevata costante di coppia, quindi una risposta rapida. I motori coreless di tipo brushed utilizzano spazzole per realizzare il contatto e sono efficaci in applicazioni che richiedono alta precisione, rapida capacità di risposta sia a bassa sia ad alta velocità, nonché compattezza ed elevata efficienza.

Figura 2: Parrot Mambo, vista verticale.

- Dal punto di vista dell'alimentazione utilizza una batteria ai polimeri di litio (LiPo) da 550 mAh, che garantisce circa 10 minuti di autonomia senza accessori collegati o paraurti, che scende a 8 minuti montando le carene o altri accessori, mentre la ricarica richiede circa 30 minuti con un caricatore da 2.1 A.
- Nella parte inferiore è alloggiata una piccola fotocamera verticale da 60 FPS e 0.3 megapixel.
- Le dimensioni del drone sono di 18×18 centimetri con i paraeliche inseriti, che si riducono a circa 13×13 centimetri rimuovendo i paraurti. In quest'ultimo caso il peso risulta essere di 63 grammi circa.

Sensore a ultrasuoni Il sensore a ultrasuoni è uno strumento che consente di misurare la distanza da un oggetto generando onde sonore. In particolare, esso calcola la distanza tra un oggetto e il sensore tramite il ritardo di tempo fra l'onda generata dal trasduttore e l'onda riflessa dall'oggetto. Il valore fornito dal sensore dipende dal periodo di tempo dell'eco e dalla frequenza dell'onda generata. Maggiore è il periodo di tempo dell'eco dell'onda generata, maggior è la distanza dall'oggetto. In figura 3 è mostrata la parte inferiore del drone ed il sensore a ultrasuoni è indicato con una freccia.

Figura 3: Parrot Mambo, sensore a ultrasuoni.

Il segnale fornito dal sensore a ultrasuoni mostra la distanza fra il sensore e la superficie più vicina, quindi è possibile utilizzarlo per misurare la distanza del drone dal suolo. Pertanto consente di determinare l'altitudine del drone, ipotizzando che non ci siano ostacoli fra il dispositivo e il suolo. Tale sensore può anche essere usato in applicazioni che richiedono di evitare ostacoli.
Il funzionamento del sensore è spiegato in maniera dettagliata in [7].

2 Modellazione matematica del drone

Il quadricottero presenta sei gradi di libertà, indicati con la sigla *6DOF*, dall'inglese *Degree of Freedom*, di cui tre rotazionali e tre traslazionali, controllati modificando la velocità dei quattro rotori, i quali generano quattro forze di spinta indipendenti. La complessità del controllo di un quadricottero dipende dalla non linearità della sua dinamica e dal numero di parametri di volo coinvolti.
In definitiva, si hanno sei gradi di libertà ma solo quattro ingressi di controllo, perciò si può parlare di sistema sottoattuato o *underactuated system*.
Per ottenere un modello matematico del drone si devono definire due sistemi di coordinate, come visibile nella figura 4:

- Il sistema di riferimento inerziale, fisso rispetto al suolo, chiamato *Earth Frame*, con gli assi N (Nord), E (Est) e D (rivolta verso il basso), la cui origine è posizionata al livello del suolo;

- Il sistema di riferimento solidale al drone, denominato *Body Frame*, costituito dagli assi x, y e z. L'origine del sistema *Body Frame* è situato nel centro di massa del quadricottero (centro del corpo del drone) e i suoi assi sono paralleli a quelli del sistema inerziale. La direzione positiva dell'asse x è rivolta verso il motore 1, quella dell'asse y verso il motore 2 mentre l'asse z è diretta verso il basso.

La posizione del drone è descritta rispetto al sistema di riferimento *inerziale* mentre le velocità traslazionale e angolare sono definite nel sistema *Body Frame*.

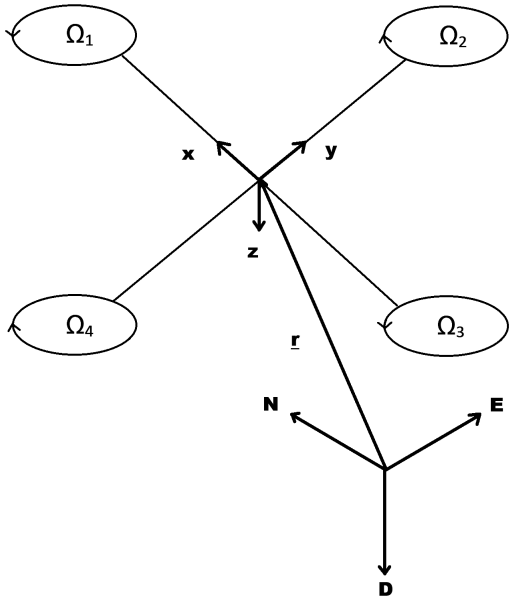
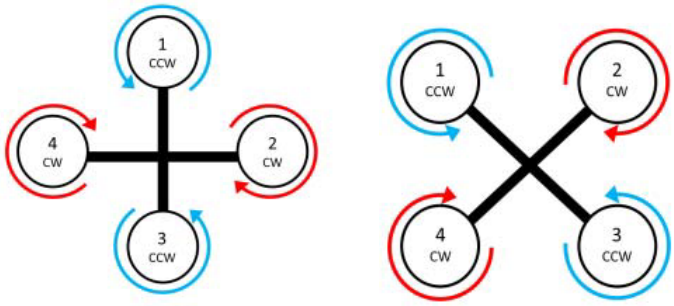


Figura 4: Sistemi di riferimento *Body Frame* e *Earth Frame*.

I motori, per bilanciare l'imbardata prodotta a causa della resistenza, ruotano nella stessa direzione a coppie: i motori 1 e 3 ruotano in senso antiorario, mentre quelli indicati con 2 e 4 si muovono in senso orario. I movimenti del drone si ottengono variando la velocità angolare di ciascun rotore.

2.1 Configurazione a *più* e a *croce*

A seconda dell'orientazione, si può considerare una struttura a '+' oppure a 'x' dei rotori, come osservabile in figura 5.



(a) Struttura a più. (b) Struttura a croce.

Figura 5: Struttura a più e a croce del drone.

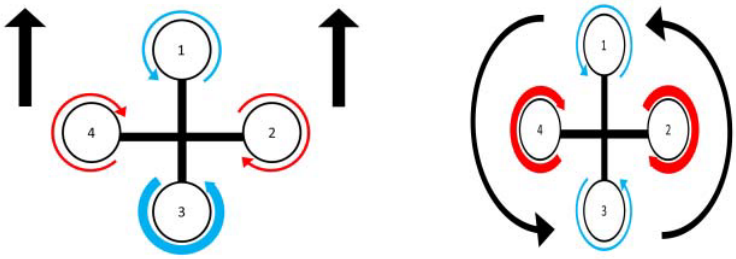
Prendendo in considerazione la struttura a *più*, il drone può effettuare i seguenti tipi di movimento:

- La **spinta verticale** del quadricottero si ottiene quando tutti i quattro motori ruotano alla stessa velocità (figura 6a).
- Il **roll** o **rollio** è la rotazione del drone attorno all'asse x e si verifica variando la velocità dei motori 2 e 4 situati lungo l'asse y. Quando il rotore 4 aumenta la sua velocità di rotazione mentre quella del rotore 2 diminuisce (figura 6b) il drone si muove verso destra, che corrisponde alla direzione positiva dell'asse y; viceversa, per spostarsi verso sinistra. La somma delle due velocità è mantenuta costante in modo da conservare l'equilibrio di coppia complessivo.

(a) Spinta verticale. (b) Roll.

Figura 6: Spinta verticale e Roll.

- Il **pitch** o **beccheggio** è la rotazione del drone attorno all'asse y e si verifica variando la velocità dei motori 1 e 3 situati lungo l'asse x. Quando il rotore 3 aumenta la sua velocità di rotazione mentre quella del rotore 1 diminuisce (figura 7a) il drone si muove in avanti, cioè nella direzione positiva dell'asse x; viceversa, per muoversi nel verso opposto. Anche in questo caso la somma delle velocità rimane costante.
- Lo **yaw** o **imbardata** è la rotazione del drone attorno all'asse z e si ottiene variando la velocità della coppia di motori lungo l'asse x rispetto alla velocità della coppia di motori lungo l'asse y. Per ottenere una rotazione in senso antiorario (figura 7b) si decrementa la velocità dei rotori 1 e 3, che ruotano in senso antiorario, mentre viene aumentata la velocità dei motori 2 e 4, che ruotano in senso orario. Si procede nel modo opposto per avere una rotazione in senso orario.



(a) Pitch. (b) Yaw.

Figura 7: Pitch e Yaw.

In figura 8 vengono raffigurati i movimenti appena descritti, questa volta nel caso di struttura a *croce*.

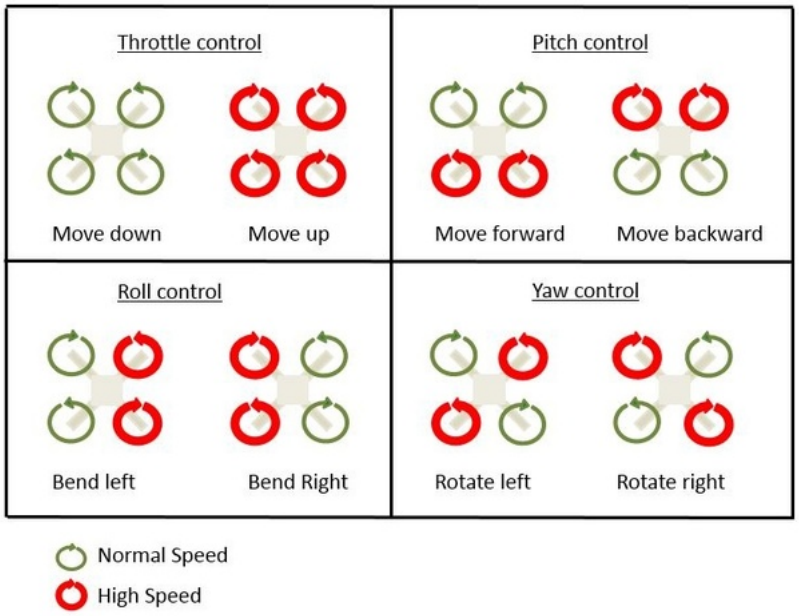


Figura 8: Movimenti con configurazione a x.

In questo caso, i movimenti di **spinta verticale** e di **yaw** si ottengono nello stesso modo spiegato in precedenza.
Per quanto riguarda il **roll**, per far sì che il drone si muova verso destra, è necessario aumentare la velocità dei motori 1 e 4 e conseguentemente diminuire la velocità dei motori 2 e 3. Invece per il **pitch**, affinché il drone si muova in avanti, i rotori 3 e 4 devono aumentare la velocità di rotazione, mentre i motori 1 e 2 devono diminuirla.
Nelle successive analisi si considerano i sistemi di riferimento mostrati in figura 4, quindi con configurazione a *più*.

2.2 Equazioni cinematiche

Le variabili di stato della velocità sono espresse nel *Body Frame*, da qui in poi indicato con B, mentre le variabili di stato per la posizione sono indicate rispetto all'*Earth Frame*, denominato E, dal vettore $r = \begin{bmatrix} x & y & z \end{bmatrix}^T$.
Per effettuare la trasformazione di coordinate tra i due sistemi di riferimento, cioè dal sistema E al sistema B, si definisce una matrice di rotazione, definita come R_E^B come di seguito:

$$R_E^B = R(\phi)R(\theta)R(\psi)$$

$$R(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{bmatrix} \tag{1}$$

$$R(\theta) = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \tag{2}$$

$$R(\psi) = \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3}$$

Moltiplicando le tre matrici sopra definite si ricava una matrice ortogonale che rappresenta appunto la trasformazione $E \rightarrow B$. Di seguito si usa la notazione $s = \sin$ e $c = \cos$.

$$R_E^B = \begin{bmatrix} c(\psi)c(\theta) & s(\psi)c(\theta) & -s(\theta) \\ c(\psi)s(\phi)s(\theta) - c(\phi)s(\psi) & s(\phi)s(\psi)s(\theta) + c(\phi)c(\psi) & c(\theta)s(\phi) \\ c(\phi)c(\psi)s(\theta) + s(\phi)s(\psi) & c(\phi)s(\psi)s(\theta) - c(\psi)s(\phi) & c(\phi)c(\theta) \end{bmatrix} \tag{4}$$

Dove ϕ rappresenta l'angolo di *roll*, θ è l'angolo di *pitch* e ψ quello di *yaw*.
Analogamente, per trasformare le variabili dal sistema B al sistema E, viene definita la matrice di trasformazione R_B^E:

$$R_B^E = R(\psi)^T R(\theta)^T R(\phi)^T$$

$$R_B^E = \begin{bmatrix} c(\psi)c(\theta) & c(\psi)s(\phi)s(\theta) - c(\phi)s(\psi) & c(\phi)c(\psi)s(\theta) + s(\phi)s(\psi) \\ s(\psi)c(\theta) & s(\phi)s(\psi)s(\theta) + c(\phi)c(\psi) & c(\phi)s(\psi)s(\theta) - c(\psi)s(\phi) \\ -s(\theta) & c(\theta)s(\phi) & c(\phi)c(\theta) \end{bmatrix} \tag{5}$$

L'equazione cinematica traslazionale può essere scritta nel seguente modo:

$$\dot{r} = R_B^E \cdot v \tag{6}$$

in cui $\dot{r}$, derivata prima rispetto al tempo del vettore posizione, e v rappresentano la velocità lineare del centro di massa del quadricottero rispettivamente nel sistema *inerziale* e nel *Body Frame*.
Anche per quanto riguarda le velocità angolari sono necessarie matrici di rotazione per passare da un sistema di riferimento all'altro, le quali vengono ricavate a partire dalle matrici (1), (2) e (3). Se si indica con $\omega = \begin{bmatrix} p & q & r \end{bmatrix}^T$ la velocità angolare rispetto al sistema B solidale con il drone e con $\dot{\eta} = \begin{bmatrix} \dot{\phi} & \dot{\theta} & \dot{\psi} \end{bmatrix}^T$ la velocità angolare rispetto al sistema inerziale, cioè la derivata prima rispetto al tempo del vettore $\eta = \begin{bmatrix} \phi & \theta & \psi \end{bmatrix}^T$, che rappresenta gli angoli di Eulero del quadricottero, quindi la sua orientazione, rispetto al sistema inerziale, si ottiene la seguente matrice di rotazione per ottenere ω a partire da $\dot{\eta}$, quindi per passare da E a B:

$$\omega = S_E^B \cdot \dot{\eta} \tag{7}$$

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

$$\omega = \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & -s(\theta) \\ 0 & c(\phi) & s(\phi)c(\theta) \\ 0 & -s(\phi) & c(\phi)c(\theta) \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = S_E^B \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{8}$$

$$S_E^B = \begin{bmatrix} 1 & 0 & -s(\theta) \\ 0 & c(\phi) & s(\phi)c(\theta) \\ 0 & -s(\phi) & c(\phi)c(\theta) \end{bmatrix} \tag{9}$$

La trasformazione inversa, dal *Body Frame B* all'*Earth Frame E*, si ricava di conseguenza:

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

L'equazione cinematica rotazionale è quindi:

$$\dot{\eta} = S_B^E \cdot \omega \tag{11}$$

2.3 Modello della dinamica

Il moto del quadricottero può essere distinto in due componenti: il movimento traslazionale e il movimento rotazionale. Prima di proseguire con

lo sviluppo del modello matematico del drone usando il formalismo di *Newton - Eulero* (come spiegato in [8] e [11]), sono necessarie, senza perdità di generalità, le seguenti ipotesi semplificative:

- Il quadricottero è un corpo rigido simmetrico, da cui segue che la matrice di inerzia è supposta essere diagonale.
- La massa del drone è concentrata nel centro di gravità, che coincide con l'origine del *Body Frame*.
- Le eliche sono considerate anch'esse corpi rigidi, in modo tale da poter trascurare l'effetto della deformazione durante la rotazione (flapping).
- Spinta e resistenza aerodinamica sono proporzionali al quadrato della velocità delle eliche.

Equazioni del moto rotatorio Usando, come anticipato, il metodo di *Newton - Eulero*, il moto rotatorio può essere descritto usando le seguenti equazioni:

$$J\dot{\omega} + \omega \times J\omega + M_G = M_B \tag{12}$$

in cui

- $J = \begin{bmatrix} I_X & 0 & 0 \\ 0 & I_Y & 0 \\ 0 & 0 & I_Z \end{bmatrix}$: matrice di inerzia;
- ω: velocità angolare nel *Body Frame*;
- M_G: effetto giroscopico prodotto dai rotori;
- M_B: momento torcente applicato al drone dai rotori.

Il primo termine rappresenta il momento angolare nel sistema *Body fixed*, il termine $\omega \times J\omega$ rappresenta l'effetto giroscopico prodotto dal corpo del drone mentre M_G deriva dalla relazione

$$M_G = \omega \times \begin{bmatrix} 0 \\ 0 \\ J_r\Omega_r \end{bmatrix} \tag{13}$$

dove

- J_r è l'inerzia dei rotori;
- $\Omega_r = -\Omega_1 + \Omega_2 - \Omega_3 + \Omega_4$ è la velocità relativa dei rotori;

Sostituendo la (13) nella (12), si ricava:

$$J\dot{\omega} + \omega \times J\omega + \omega \times \begin{bmatrix} 0 \\ 0 \\ J_r\Omega_r \end{bmatrix} = M_B \tag{14}$$

M_B deriva dalla forza aerodinamica F_i e dal momento aerodinamico M_i generati dai rotori e definiti di seguito:

$$F_i = \frac{1}{2}\rho A C_T d^2 \Omega_i^2 \tag{15}$$

$$M_i = \frac{1}{2}\rho A C_D d^2 \Omega_i^2 \tag{16}$$

con

- ρ è la densità dell'aria;
- A è l'area della sezione dell'elica;
- C_T, C_D sono i coefficienti aerodinamici;
- d è il diametro dell'elica;
- Ω_i è la velocità angolare del rotore i-esimo.

Visto che A, C_T, C_D e d possono essere considerati costanti nelle equazioni (15) e (16), l'altezza di volo del quadricottero è limitata e la densità dell'aria può essere considerata costante, le equazioni sopra menzionate possono essere scritte nella seguente forma:

$$F_i = K_F \Omega_i^2 \tag{17}$$

$$M_i = K_M \Omega_i^2 \tag{18}$$

Complessivamente vengono applicate al corpo rigido tre coppie, vale a dire una intorno a ciascun asse, determinate usando la regola della mano destra: l'equazione (19) descrive la coppia intorno all'asse x, la coppia relativa all'asse y è data dalla (20), infine la coppia attorno all'asse z è fornita dall'equazione (21). La direzione di rotazione di ciascun motore può essere vista in figura 9.

$$M_x = -F_2 l + F_4 l = -(K_F\Omega_2^2)l + (K_F\Omega_4^2)l = lK_F(-\Omega_2^2 + \Omega_4^2) \tag{19}$$

$$M_y = F_1 l - F_3 l = (K_F\Omega_1^2)l - (K_F\Omega_3^2)l = lK_F(\Omega_1^2 - \Omega_3^2) \tag{20}$$

$$\begin{aligned} M_z &= M_1 - M_2 + M_3 - M_4 = \\ &= (K_M\Omega_1^2) - (K_M\Omega_2^2) + (K_M\Omega_3^2) - (K_M\Omega_4^2) = \\ &= K_M(\Omega_1^2 - \Omega_2^2 + \Omega_3^2 - \Omega_4^2) \end{aligned} \tag{21}$$

Utilizzando le equazioni (19), (20) e (21), M_B si può scrivere nella seguente forma matriciale:

$$M_B = \begin{bmatrix} lK_F(-\Omega_2^2 + \Omega_4^2) \\ lK_F(\Omega_1^2 - \Omega_3^2) \\ K_M(\Omega_1^2 - \Omega_2^2 + \Omega_3^2 - \Omega_4^2) \end{bmatrix} \tag{22}$$

Il parametro l rappresenta la distanza fra il centro di ciascun rotore e l'origine del sistema di coordinate B.

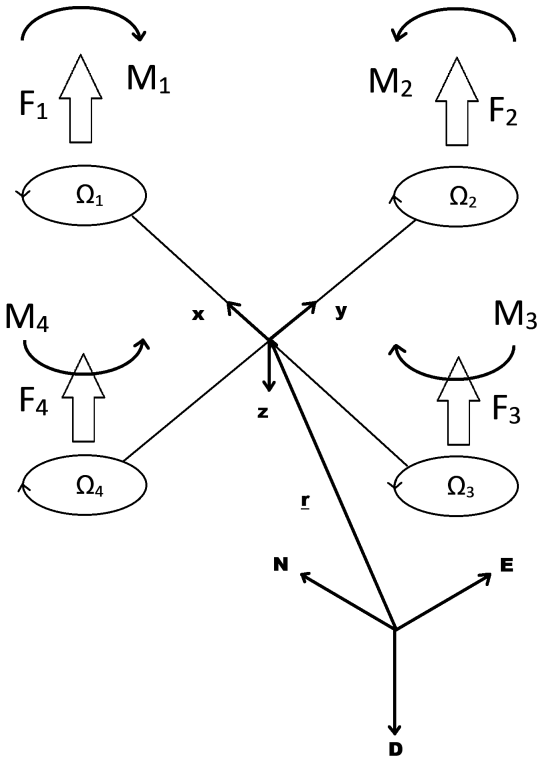


Figura 9: Forze e momenti agenti sul drone.

Equazioni del moto traslatorio L'equazione che descrive il movimento di traslazione è basata sulla seconda legge di Newton ed è relativa al sistema di coordinate inerziale *Earth fixed*:

$$m\ddot{r} = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} + R_B^E F_B + \begin{bmatrix} -K_{dx} & 0 & 0 \\ 0 & -K_{dy} & 0 \\ 0 & 0 & -K_{dz} \end{bmatrix} \dot{r} \tag{23}$$

in cui:

- $r = \begin{bmatrix} x & y & z \end{bmatrix}^T$ è la posizione del quadricottero rispetto al sistema E;
- m è la massa del drone;
- g è l'accelerazione di gravità;
- R_B^E è la matrice di rotazione dal *Body Frame* al sistema *inerziale*;
- F_B è la forza non gravitazionale applicata al drone;
- K_{dx}, K_{dy}, K_{dz} sono i coefficienti di attrito traslazionale.

Nello specifico, la forza F_B è composta dalla somma delle forze generate dai rotori, che dipendono dal coefficiente di portanza K_F e dal quadrato della velocità angolare dei rotori:

$$F_B = \begin{bmatrix} 0 \\ 0 \\ -K_F(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \end{bmatrix} \tag{24}$$

L'ultimo termine dell'equazione (23) è la risultante delle forze di attrito lungo gli assi x, y e z.

2.4 Modello in spazio di stato

Sulla base del modello cinematico e dinamico, può essere derivata la descrizione in spazio di stato del sistema. Il vettore che descrive gli stati del quadricottero è il seguente:

$$X = \begin{bmatrix} \phi & \dot{\phi} & \theta & \dot{\theta} & \psi & \dot{\psi} & x & \dot{x} & y & \dot{y} & z & \dot{z} \end{bmatrix}^T \tag{25}$$

Il sistema presenta quattro ingressi:

$$U = \begin{bmatrix} U_1 & U_2 & U_3 & U_4 \end{bmatrix} \tag{26}$$

dove

$$U_1 = K_F(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \tag{27}$$
$$U_2 = K_F(-\Omega_2^2 + \Omega_4^2) \tag{28}$$
$$U_3 = K_F(\Omega_1^2 - \Omega_3^2) \tag{29}$$
$$U_4 = K_M(\Omega_1^2 - \Omega_2^2 + \Omega_3^2 - \Omega_4^2) \tag{30}$$

U_1 è responsabile della salita verticale (z, $\dot{z}$), U_2 rappresenta la rotazione di roll (ϕ, $\dot{\phi}$), U_3 rappresenta la rotazione di pitch (θ, $\dot{\theta}$) mentre U_4 la rotazione

di yaw intorno all'asse z (ψ, $\dot{\psi}$). La coppia totale applicata al quadricottero M_B si deriva sostituendo le equazioni (28), (29), (30) in (22) e può essere scritta nella forma seguente:

$$M_B = \begin{bmatrix} lU_2 \\ lU_3 \\ U_4 \end{bmatrix} \tag{31}$$

A partire dall'equazione (8), che mette in relazione la velocità angolare ω del *Body Frame* con il vettore $\dot{\eta}$ relativo all'*Earth Frame*, si ricava facilmente:

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} \dot{\phi} - \dot{\psi}\sin(\theta) \\ \dot{\theta}\cos(\phi) + \dot{\psi}\sin(\phi)\cos(\theta) \\ -\dot{\theta}\sin(\phi) + \dot{\psi}\cos(\phi)\cos(\theta) \end{bmatrix} \tag{32}$$

Dopo semplici passaggi si ottiene:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} (p\cos(\theta) + q\sin(\phi)\sin(\theta) + r\cos(\phi)\sin(\theta))/\cos(\theta) \\ q\cos(\phi) - r\sin(\phi) \\ (q\sin(\phi) + r\cos(\phi))/\cos(\theta) \end{bmatrix} \tag{33}$$

Pertanto le equazioni che descrivono le componenti del vettore $\dot{\eta}$, cioè la derivata rispetto al tempo degli angoli di Eulero, sono:

$$\begin{cases} \dot{\phi} = \dfrac{p\cos(\theta) + q\sin(\phi)\sin(\theta) + r\cos(\phi)\sin(\theta)}{\cos(\theta)} \\ \dot{\theta} = q\cos(\phi) - r\sin(\phi) \\ \dot{\psi} = \dfrac{q\sin(\phi) + r\cos(\phi)}{\cos(\theta)} \end{cases} \tag{34}$$

Come visto in precedenza, usando il metodo di *Newton - Eulero*, il moto rotatorio è descritto dall'equazione (14), qui sotto scritta in funzione degli ingressi U e ulteriormente sviluppata per determinare le componenti del vettore $\dot{\omega}$:

$$J\dot{\omega} + \omega \times J\omega + \omega \times \begin{bmatrix} 0 \\ 0 \\ J_r\Omega_r \end{bmatrix} = \begin{bmatrix} lU_2 \\ lU_3 \\ U_4 \end{bmatrix} \tag{35}$$

$$\begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix} \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} + \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} + \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ J_r\Omega_r \end{bmatrix} = \begin{bmatrix} lU_2 \\ lU_3 \\ U_4 \end{bmatrix} \tag{36}$$

$$\begin{bmatrix} I_x\dot{p} \\ I_y\dot{q} \\ I_z\dot{r} \end{bmatrix} + \begin{bmatrix} (I_z - I_y)qr \\ (I_x - I_z)pr \\ (I_y - I_x)pq \end{bmatrix} + \begin{bmatrix} J_r\Omega_r q \\ -J_r\Omega_r p \\ 0 \end{bmatrix} = \begin{bmatrix} lU_2 \\ lU_3 \\ U_4 \end{bmatrix} \tag{37}$$

Le equazioni che descrivono le componenti dell'accelerazione angolare $\dot{\omega}$ nel *Body Frame* sono quindi le seguenti:

$$\begin{cases} \dot{p} = \dfrac{I_y - I_z}{I_x}qr - \dfrac{J_r\Omega_r}{I_x}q + \dfrac{lU_2}{I_x} \\ \dot{q} = \dfrac{I_z - I_x}{I_y}pr + \dfrac{J_r\Omega_r}{I_y}p + \dfrac{lU_3}{I_y} \\ \dot{r} = \dfrac{I_x - I_y}{I_z}pq + \dfrac{U_4}{I_z} \end{cases} \tag{38}$$

Per quanto riguarda l'equazione della dinamica (23) che esprime il moto traslatorio, essa si può sviluppare come segue:

$$m\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} + R_B^E \begin{bmatrix} 0 \\ 0 \\ -U_1 \end{bmatrix} + \begin{bmatrix} -K_{dx} & 0 & 0 \\ 0 & -K_{dy} & 0 \\ 0 & 0 & -K_{dz} \end{bmatrix} \dot{r} \tag{39}$$

Si ricavano pertanto le seguenti accelerazioni traslazionali:

$$\begin{cases} \ddot{x} = -\dfrac{U_1}{m}(\cos(\phi)\sin(\theta)\cos(\psi) + \sin(\phi)\sin(\psi) + K_{dx}\dot{x}) \\ \ddot{y} = -\dfrac{U_1}{m}(\cos(\phi)\sin(\theta)\sin(\psi) - \sin(\phi)\cos(\psi) + K_{dy}\dot{y}) \\ \ddot{z} = g - \dfrac{U_1}{m}(\cos(\phi)\cos(\theta) + K_{dz}\dot{z}) \end{cases} \tag{40}$$

Complessivamente, le equazioni non lineari del moto del quadricottero si ottengono combinando le equazioni del moto rotazionale e del moto trasla-

zionale:

$$\begin{cases} \ddot{x} = -\dfrac{U_1}{m}(\cos(\phi)\sin(\theta)\cos(\psi) + \sin(\phi)\sin(\psi) + K_{dx}\dot{x}) \\ \ddot{y} = -\dfrac{U_1}{m}(\cos(\phi)\sin(\theta)\sin(\psi) - \sin(\phi)\cos(\psi) + K_{dy}\dot{y}) \\ \ddot{z} = g - \dfrac{U_1}{m}(\cos(\phi)\cos(\theta) + K_{dz}\dot{z}) \\ \dot{p} = \dfrac{I_y - I_z}{I_x}qr - \dfrac{J_r\Omega_r}{I_x}q + \dfrac{lU_2}{I_x} \\ \dot{q} = \dfrac{I_z - I_x}{I_y}pr + \dfrac{J_r\Omega_r}{I_y}p + \dfrac{lU_3}{I_y} \\ \dot{r} = \dfrac{I_x - I_y}{I_z}pq + \dfrac{U_4}{I_z} \\ \dot{\phi} = \dfrac{p\cos(\theta) + q\sin(\phi)\sin(\theta) + r\cos(\phi)\sin(\theta)}{\cos(\theta)} \\ \dot{\theta} = q\cos(\phi) - r\sin(\phi) \\ \dot{\psi} = \dfrac{q\sin(\phi) + r\cos(\phi)}{\cos(\theta)} \end{cases} \tag{41}$$

2.4.1 Linearizzazione

Si possono introdurre alcune ipotesi che consentano di semplificare il precedente sistema di equazioni, che descrive la dinamica del moto del quadricottero, per renderlo più adatto per la descrizione del suo comportamento e per il controllo.
In assenza di correnti d'aria significative, come avviene in un ambiente chiuso, i coefficienti di attrito possono essere trascurati mentre, ipotizzando che gli angoli di Eulero siano sufficientemente piccoli, condizione verificata nella situazione di volo stazionario, la matrice S_E^B, riportata in (9), viene approssimata con la matrice identità.
Applicando quindi una linearizzazione in un punto di volo stazionario, gli angoli di **roll** e **pitch** sono considerati pressocché nulli e pertanto i relativi coseni valgono 1 mentre i seni si annullano. Gli elementi che diventano significativi ad alte velocità vengono trascurati e il modello è semplificato sulla base delle ipotesi per cui le velocità e le accelerazioni rotazionali e traslazionali sono considerate sufficientemente piccole.
In definitiva vale la seguente relazione:

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} \approx \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{42}$$

Il sistema complessivo si può riscrivere quindi nella forma seguente:

$$\begin{cases} \ddot{x} = -\dfrac{U_1}{m}(\cos(\phi)\sin(\theta)\cos(\psi) + \sin(\phi)\sin(\psi)) \\ \ddot{y} = -\dfrac{U_1}{m}(\cos(\phi)\sin(\theta)\sin(\psi) - \sin(\phi)\cos(\psi)) \\ \ddot{z} = g - \dfrac{U_1}{m}(\cos(\phi)\cos(\theta)) \\ \ddot{\phi} = \dfrac{I_y - I_z}{I_x}\dot{\theta}\dot{\psi} - \dfrac{J_r\Omega_r}{I_x}\dot{\theta} + \dfrac{lU_2}{I_x} \\ \ddot{\theta} = \dfrac{I_z - I_x}{I_y}\dot{\phi}\dot{\psi} + \dfrac{J_r\Omega_r}{I_y}\dot{\phi} + \dfrac{lU_3}{I_y} \\ \ddot{\psi} = \dfrac{I_x - I_y}{I_z}\dot{\phi}\dot{\theta} + \dfrac{U_4}{I_z} \end{cases} \tag{43}$$

2.4.2 Equazioni nella configurazione a *croce*

Per quanto riguarda la configurazione a *croce*, maggiormente rappresentativa del drone reale, le equazioni ricavate rimangono valide, come si vede in [15] e in [16], opportunamente modificate considerando la diversa orientazione degli assi x, y e z, mostrata in figura 10.

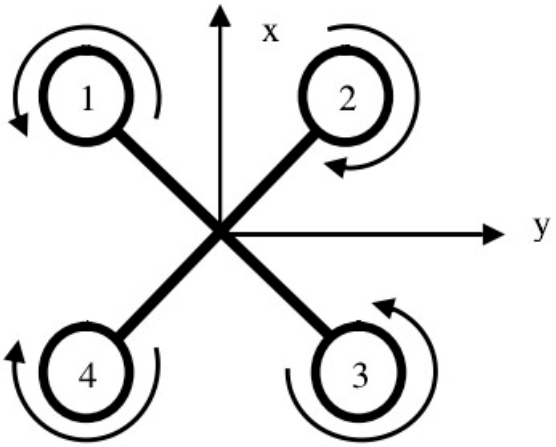


Figura 10: Sistema di riferimento *Body fixed* per la configurazione a croce.

In particolare, l'equazione che descrive il moto rotatorio si differenzia dalla precedente per quanto riguarda le coppie applicate al drone dai quattro motori, ovvero le componenti del vettore M_B, mostrato in (44), mentre l'equazione del moto traslatorio rimane invariata.

$$M_B = \begin{bmatrix} lK_F(\Omega_1^2 - \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \\ lK_F(\Omega_1^2 + \Omega_2^2 - \Omega_3^2 - \Omega_4^2) \\ K_M(\Omega_1^2 - \Omega_2^2 + \Omega_3^2 - \Omega_4^2) \end{bmatrix} \tag{44}$$

Si definisce il nuovo vettore degli ingressi U in questo modo:

$$U_1 = K_F(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \tag{45}$$
$$U_2 = K_F(\Omega_1^2 - \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \tag{46}$$
$$U_3 = K_F(\Omega_1^2 + \Omega_2^2 - \Omega_3^2 - \Omega_4^2) \tag{47}$$
$$U_4 = K_M(\Omega_1^2 - \Omega_2^2 + \Omega_3^2 - \Omega_4^2) \tag{48}$$

La componente U_1, relativo alla **spinta verticale**, e la componente U_4, relativa allo **yaw**, sono le stesse del caso precedente, coerentemente con quanto affermato nel paragrafo 2.1 relativo alle configurazioni, mentre le componenti

U_2 e U_3, che fanno riferimento rispettivamente a **roll** e **pitch**, tengono conto del fatto che nella configurazione a *croce* in questi movimenti entrano in gioco tutti e 4 i motori, differentemente dal caso di configurazione a *più*.
Di conseguenza il vettore M_B, scritto rispetto al vettore degli ingressi definito dalle equazioni (45), (46), (47), (48) risulta essere:

$$M_B = \begin{bmatrix} lU_2 \\ lU_3 \\ U_4 \end{bmatrix} \tag{49}$$

Ci si è pertanto ricondotti alle equazioni mostrate in precedenza.

3 Modello Simulink

Il modello Simulink in figura 11, fornito dall'*Aerospace Blockset*, raffigura lo schema in controreazione relativo al drone Parrot Mambo, in cui il blocco *Airframe* rappresenta il processo, cioè il modello del quadricottero, mentre *FCS* è il blocco relativo al controllo.

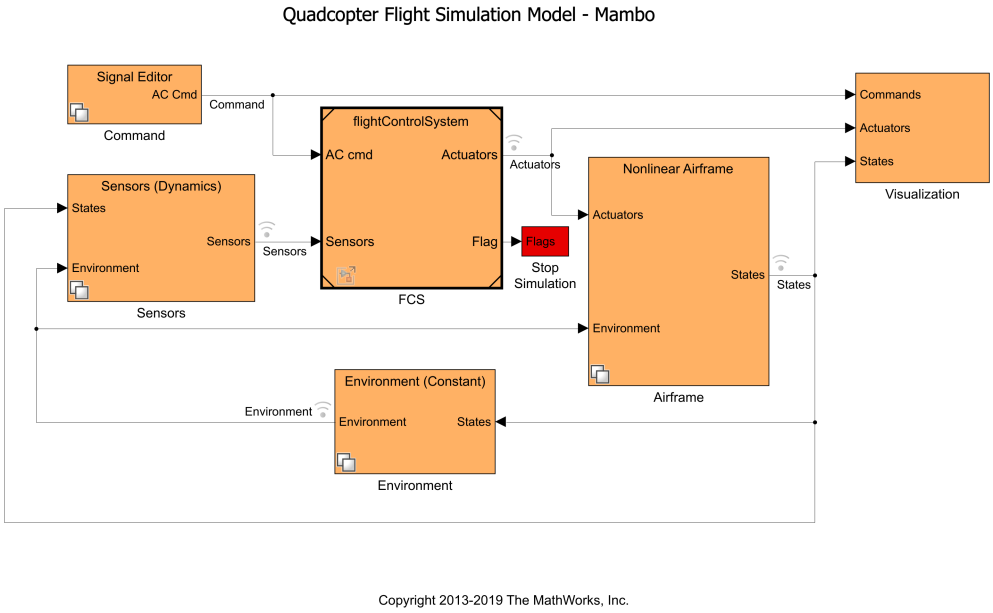


Figura 11: Progetto *asbQuadcopter*.

3.1 Airframe

All'interno del blocco *Airframe* si può scegliere fra il modello lineare e non lineare, impostando opportunamente la variabile d'ambiente VSS_VEHICLE: il valore 0 corrisponde al modello lineare, viceversa il valore 1 consente di selezionare il modello non lineare.

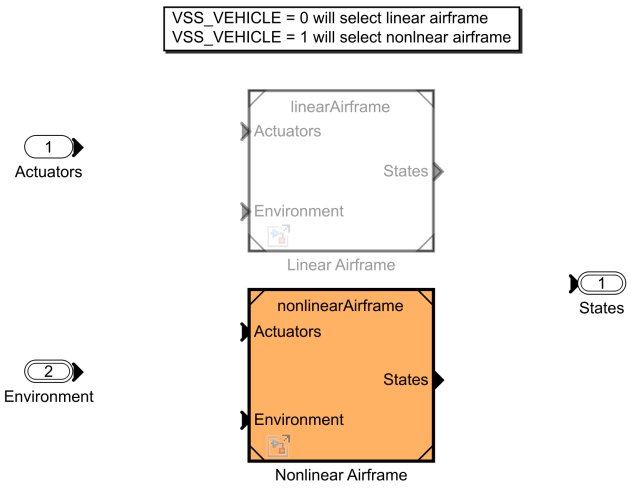


Figura 12: Scelta tra modello *lineare* (in alto) e *non lineare* (in basso).

Il modello non lineare consta di due blocchi principali, come si vede in figura 13.

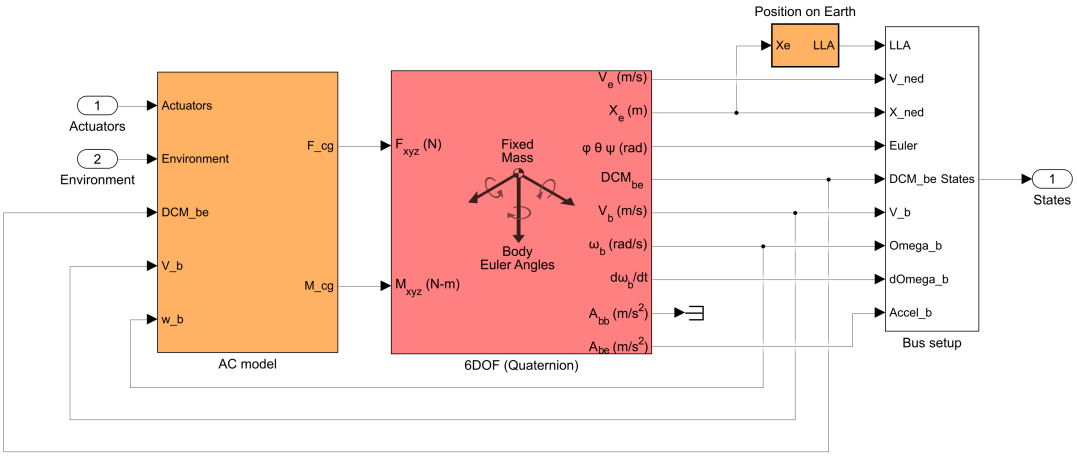


Figura 13: Modello non lineare.

Il primo è *AC model*, sulla sinistra in figura 13, che contiene i modelli degli attuatori e consente di determinare come i disturbi dell'ambiente circostante influenzino il sistema. In questo blocco vengono considerate le forze e le coppie agenti sul mini drone. Queste vengono poi fornite al modello *6DOF*.

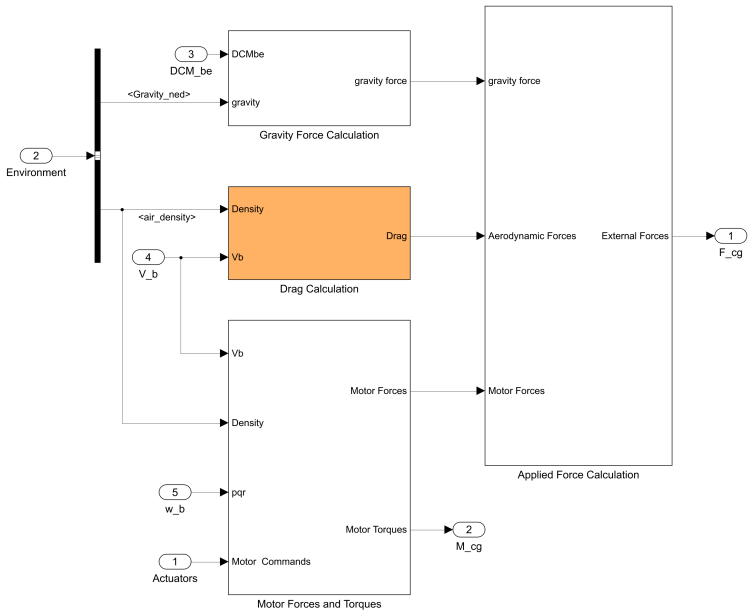


Figura 14: Blocco *AC model*.

AC model Questo sottosistema è composto da quattro blocchi, spiegati singolarmente di seguito.

Gravity Force Calculation In figura 15, consente di calcolare la componente relativa alla forza di gravità dell'equazione (23) e di moltiplicarla per la matrice DCM_{be} per convertirla nel sistema di riferimento *Body Frame*.

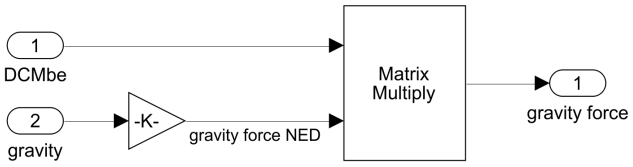


Figura 15: Blocco *Gravity Force Calculation*.

Drag Calculation Preposto al calcolo della forza di resistenza dell'aria che si oppone al moto del drone, che si ottiene con la formula:

$$F_D = \frac{1}{2}\rho v^2 A C_D \tag{50}$$

dove A è la sezione dell'elica, indicata con S da Matlab, mentre gli altri simboli hanno lo stesso significato spiegato in precedenza. Quanto detto è implementato in figura 16.

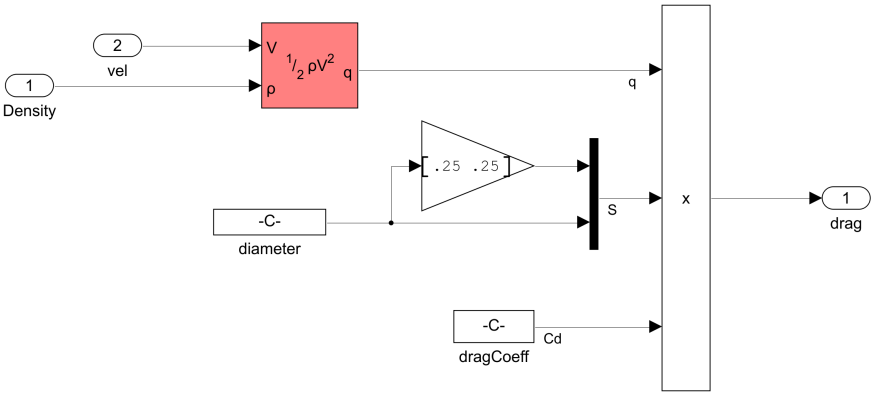


Figura 16: Blocco *Drag Calc*.

Visto che la forza di attrito deve avere direzione opposta rispetto al moto, al vettore uscente dal blocco in 16 si applica il segno opposto rispetto a quello della velocità V_b, come visibile in 17.

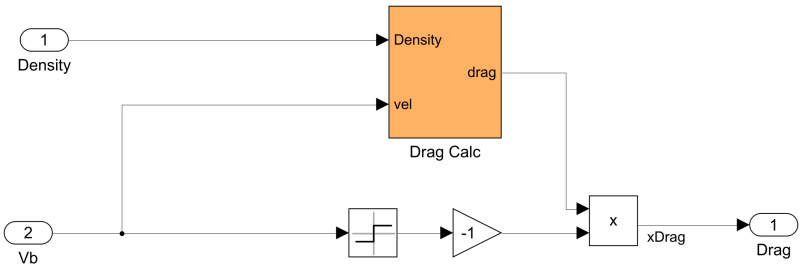


Figura 17: Blocco *Drag Calculation*.

Motor Forces and Torques Qui si calcolano le forze e i momenti generati da ciascun motore, che sono uguali e indipendenti, e pertanto si possono analizzare singolarmente senza perdita di generalità. A differenza del modello matematico descritto in precedenza, il progetto Simulink presenta un modello più dettagliato, derivato dall'articolo [17], che considera alcuni elementi precedentemente trascurati, come la coppia dovuta alla resistenza dell'aria e il fenomeno del *blade flapping*, ossia la deformazione delle eliche dovuta alle alte velocità e alla flessibilità del materiale con cui sono realizzate. Data l'equazione (50) per il calcolo della forza di attrito dell'aria, si può ottenere il corrispondente momento torcente mediante:

$$Q_i = \frac{1}{2} R\rho C_D A v^2 = \frac{1}{2} R\rho C_D A(\Omega_i R)^2 = \frac{1}{2} C_D \rho A R^3 \Omega_i |\Omega_i| e_3 \qquad (51)$$

dove R è il raggio dell'elica. La velocità dell'i-esimo rotore Ω_i viene moltiplicata per il suo modulo in modo da preservare il verso della rotazione per i

rotori che si muovono in senso opposto. Si assume che la forza sia applicata sull'estremità dell'elica. Si nota, inoltre, che il momento causato dalle forze di resistenza aerodinamica agisce solo lungo l'asse z (moltiplicazione per il versore e_3).
Mentre in (44) si considerava il parametro l come la distanza fra il centro di ciascun rotore e l'origine del sistema di coordinate B, in questa analisi si considera anche l'altezza dei rotori al di sopra del centro di massa. Pertanto il posizionamento dei rotori è il seguente:

$$D_1 = \begin{bmatrix} l_{xy} & -l_{xy} & h \end{bmatrix} \tag{52}$$

$$D_2 = \begin{bmatrix} l_{xy} & l_{xy} & h \end{bmatrix} \tag{53}$$

$$D_3 = \begin{bmatrix} -l_{xy} & l_{xy} & h \end{bmatrix} \tag{54}$$

$$D_4 = \begin{bmatrix} -l_{xy} & -l_{xy} & h \end{bmatrix} \tag{55}$$

in cui l_{xy} è memorizzata nella variabile *Vehicle.Airframe.xy* e vale 4.41 cm mentre h è contenuta in *Vehicle.Airframe.h* e vale -1.59 cm (è negativa perché l'asse z è rivolta verso il basso e l'origine del sistema è posizionata sul CdM).
Il quadricottero presenta necessariamente una distanza orizzontale fra gli alberi dei rotori e il centro di gravità. Quando il drone effettua movimenti di *pitch* e *roll*, i rotori producono una velocità verticale. Il coefficiente C_T può essere relazionato con la velocità verticale V_c, cioè la terza componente di v_{ri} rispetto a r (equazione (59)), in questo modo:

$$\frac{C_T}{\sigma} = \frac{a(\alpha)}{4}\left[\theta_{tip} - \frac{v_i + V_c}{\omega R}\right] \tag{56}$$

in cui $a(\alpha)$ è la pendenza della curva di portanza del profilo alare (che dipende dall'angolo di attacco del rotore α), θ_{tip} è l'angolo geometrico dell'elica sulla punta del rotore, v_i è la velocità indotta attraverso il rotore e σ è la rigidità del rotore, definita come il rapporto fra l'area della superficie delle eliche e la sezione orizzontale del rotore.
Nelle ipotesi fatte in 2.3 si era trascurato il fenomeno del blade flapping: nel volo traslazionale, la lama più avanzata dell'elica di un rotore presenta una velocità relativa, rispetto all'aria, maggiore rispetto all'altra lama, che ha una velocità relativa minore. Questo comporta una differenza di portanza fra le due lame, facendo sì che le lame del rotore si alzino e si abbassino una volta per giro. Questo fenomeno inclina il piano del rotore indietro rispetto alla direzione del movimento, causando diversi effetti sulla dinamica del velivolo, influendo in particolare sulla stabilità dell'assetto. Pertanto, come si vede nelle figure 18 e 19, si considerano gli angoli $a_{1_s i}$ e $b_{1_s i}$ come gli angoli di

inclinazione longitudinale e laterale dell'i-esimo rotore, dovuti al fenomeno del blade flapping.

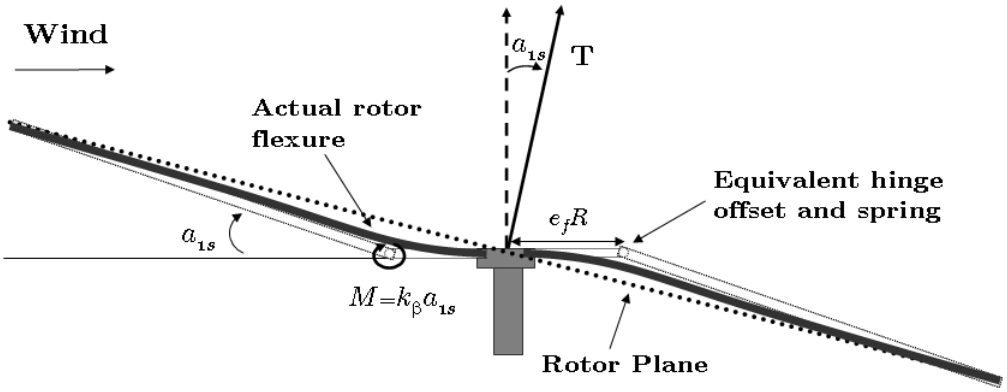


Figura 18: Effetto del blade flapping.

La spinta del rotore i-esimo diventa quindi

$$T_i = C_T \rho A R^2 \Omega_i^2 \begin{bmatrix} -\sin(a_{1_s i}) \\ \cos(a_{1_s i})\sin(b_{1_s i}) \\ -\cos(b_{1_s i})\cos(a_{1_s i}) \end{bmatrix} \tag{57}$$

mentre il relativo momento vale:

$$M_i = T_i \times D_i \tag{58}$$

Gli angoli di flapping dell'i-esimo rotore vengono determinati calcolando l'ampiezza e la direzione della traslazione del rotore e definendo un sistema di riferimento locale, B_i, allineato in tale direzione. Gli angoli di flapping longitudinale e laterale sono calcolati prima nel sistema di riferimento B_i, denominati $u_{1_s i}$ e $v_{1_s i}$, e poi espressi nel *Body Frame* come $a_{1_s i}$ e $b_{1_s i}$ utilizzando una matrice di rotazione. Per prima cosa si determina l'advance ratio e la direzione azimutale del rotore, tramite le seguenti equazioni:

$$v_{ri} = v + \omega \times D_i \tag{59}$$

$$\mu_{ri} = \frac{\|v_{r(1,2)i}\|}{\Omega_i R} \tag{60}$$

$$\psi_{ri} = \arctan\left(\frac{v_{r(2)i}}{v_{r(1)i}}\right) \tag{61}$$

L'equazione (59) calcola la velocità v_{ri} dell'i-esimo rotore rispetto al *Body Frame*. Il primo pedice indica la coordinata, $r = 1, 2, 3$ ($\equiv x, y, z$), mentre

il secondo pedice indica il motore, $i = 1, 2, 3, 4$. La (60) serve per determinare l'advance ratio dell'i-esimo motore, in cui il termine al numeratore è la norma calcolata sulle prime due componenti, longitudinale e latitudinale, del vettore velocità calcolato in (59), mentre il termine al denominatore è il prodotto tra la velocità angolare Ω_i dell'i-esimo rotore e il raggio R dell'elica. Infine la (61) definisce la direzione azimutale del movimento, cioè l'angolo di inclinazione, sul piano orizzontale, del movimento di avanzamento, che è pari all'arcotangente del rapporto fra la componente latitudinale e la componente longitudinale del vettore velocità v_{ri}. Gli angoli di flapping dell'i-esimo rotore nel sistema di coordinate locale B_i valgono:

$$u_{1_s i} = \frac{1}{1 - \dfrac{\mu_{ri}^2}{2}} \mu_{ri}(4\theta_{tip} - 2\lambda_{hi}^2) \tag{62}$$

$$v_{1_s i} = \frac{1}{1 + \dfrac{\mu_{ri}^2}{2}} \left(\frac{8 C_T \mu_{ri} \gamma}{9\sigma a} + \frac{C_T}{2\mu_{ri}} \right) \tag{63}$$

in cui $\lambda_{hi} \simeq \sqrt{C_T/2}$ è una costante nota come *inflow ratio* e rappresenta il flusso d'aria in ingresso alle eliche dell'i-esimo rotore, $\gamma = \rho a_0 c R^4 / I_b$ (c: lunghezza della corda dell'elica) è il cosiddetto *Lock Number*, il quale rappresenta il rapporto fra le forze aerodinamiche, che agiscono per sollevare l'elica, e le forze inerziali, che tendono a mantenere l'elica nel piano di rotazione.

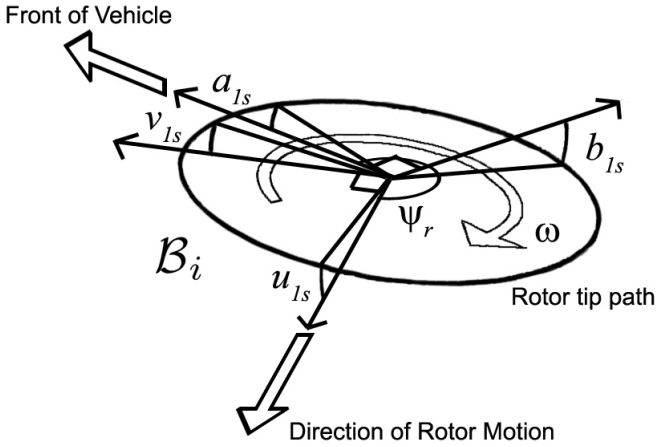


Figura 19: Angoli di flapping nei due sistemi di riferimento B_i e B.

Gli angoli $u_{1_s i}$ e $v_{1_s i}$ in (62) e (63), come detto in precedenza, sono espressi rispetto al sistema di riferimento locale B_i, pertanto si definisce la matrice

$J^B_{B_i}$ che consente di effettuare la trasformazione di coordinate $B_i \to B$:

$$J^B_{B_i} = \begin{bmatrix} \cos(\psi_{ri}) & -\sin(\psi_{ri}) \\ \sin(\psi_{ri}) & \cos(\psi_{ri}) \end{bmatrix} \tag{64}$$

$$\begin{bmatrix} a_{1_s i} \\ b_{1_s i} \end{bmatrix} = J^B_{B_i} \begin{bmatrix} u_{1_s i} \\ v_{1_s i} \end{bmatrix} \tag{65}$$

A questo punto si aggiungono le componenti degli angoli di flapping prodotte dalle velocità di *pitch* e *roll* del drone:

$$a_{1_s i} = ... + \frac{-\frac{16}{\gamma}\left(\frac{q}{\Omega_i}\right) + \left(\frac{p}{\Omega_i}\right)}{1 - \frac{\mu^2_{ri}}{2}} \tag{66}$$

$$b_{1_s i} = ... + \frac{-\frac{16}{\gamma}\left(\frac{p}{\Omega_i}\right) + \left(\frac{p}{\Omega_i}\right)}{1 + \frac{\mu^2_{ri}}{2}} \tag{67}$$

Quanto detto finora viene implementato all'interno del sottosistema *Motor Forces and Torques*, come si può osservare in figura 20, che presenta come ingressi:

- V_b, la velocità traslazionale rispetto al *Body Frame*, che coincide con v definita in (6) e che compare in (59);
- La densità dell'aria ρ;
- ω, la velocità angolare rispetto al *Body Frame*, definita in (11) e presente in (59)
- w, la velocità angolare dell'i-esimo rotore Ω_i, ottenuta dallo sforzo di controllo generato dal blocco *FCS*, processato da *MotorsToW* per ottenere un segnale interpretabile dai motori;
- D, la posizione dei rotori rispetto al CdM, definita dalle equazioni (52), (53), (54), (55).

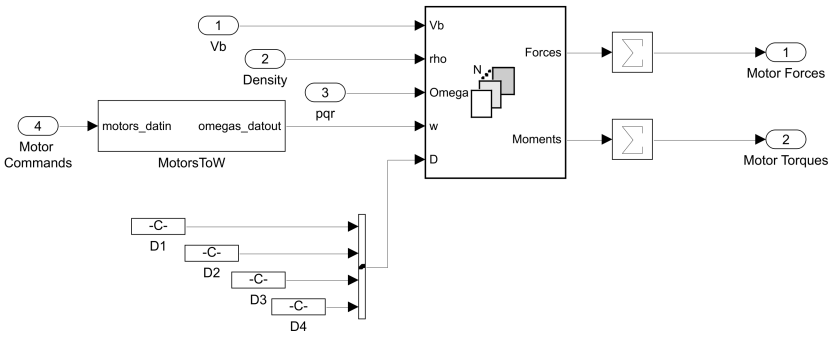


Figura 20: Sottosistema *Motor Forces and Torques*.

In uscita vengono fornite le forze e i momenti relativi a ciascun rotore che, come detto in precedenza, si assume abbiano la stessa dinamica e siano indipendenti fra loro, il che giustifica l'uso del blocco *For Each* in 20. All'interno di questo blocco si trova l'implementazione della dinamica di un rotore.

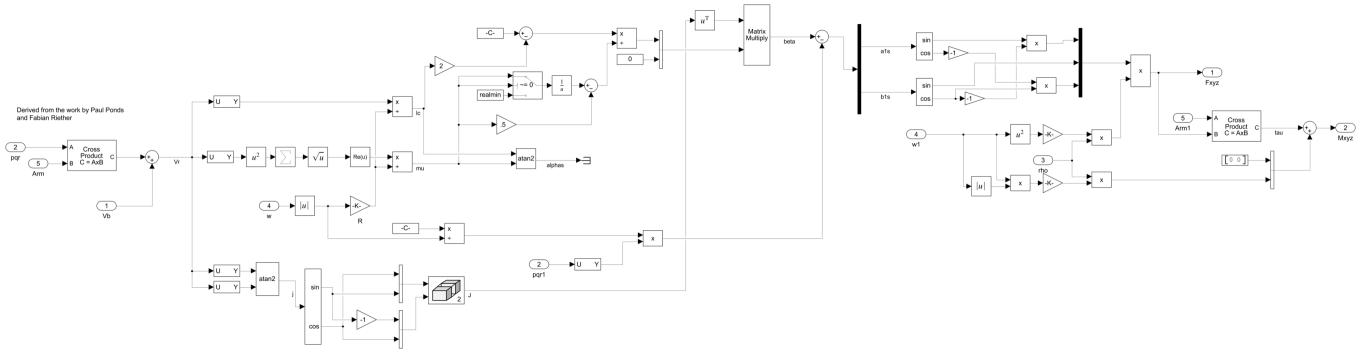


Figura 21: Dinamica di un rotore.

Innanzitutto si definisce la velocità v_{ri} come nell'equazione (59) (figura 22)

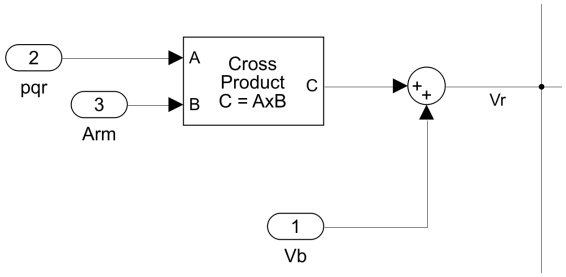


Figura 22: Implementazione eq.(59).

Successivamente sono posizionati diversi blocchi di tipo *Selector*, utilizzati per estrarre le componenti del vettore velocità relative alle diverse coordinate. Proseguendo centralmente, il blocco *Selector* preleva le prime due componenti, lungo x e y, della velocità v_{ri} e le utilizza per calcolare l'*advance ratio* μ_{ri}, come in (60):

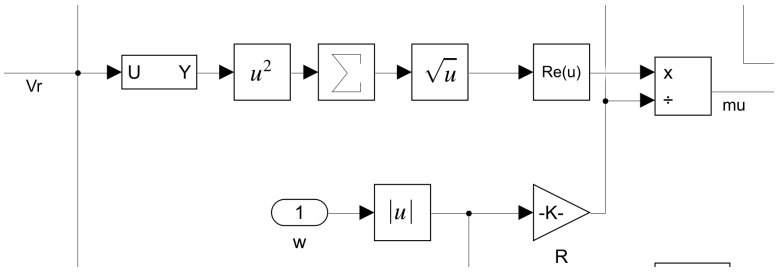


Figura 23: Implementazione eq.(60).

Nella parte inferiore vengono utilizzati due *Selector* per estrarre la prima e la seconda componente della velocità, che vengono usate per calcolare la direzione azimutale del movimento ψ_{ri} (equazione (61)), la quale a sua volta serve per costruire la matrice di trasformazione $J^B_{B_i}$ (equazione (64)):

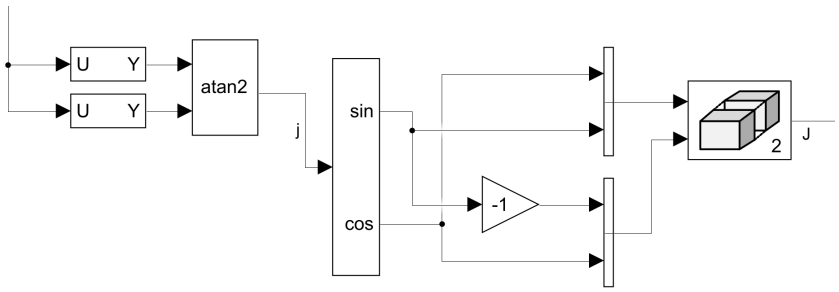


Figura 24: Implementazione delle equazioni (61) e (64).

Nella parte superiore si procede al calcolo dell'angolo di flapping $u_{1_s i}$, seguendo la formula (62): il ramo uscente dal nodo sommatore in alto in figura 25 rappresenta il termine all'interno della parentesi tonda, il quale viene poi moltiplicato per il coefficiente $\dfrac{1}{1-\dfrac{\mu_{ri}^2}{2}}\mu_{ri}$:

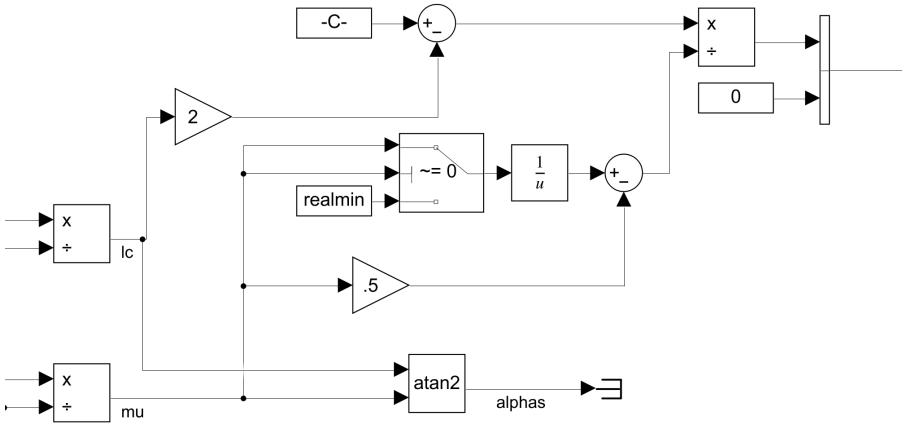


Figura 25: Implementazione eq.(62).

Come si può notare in 25, la componente v_{1_si} (63), che rappresenta l'angolo di flapping laterale, viene trascurata e impostata a zero. A questo punto si moltiplica il vettore ottenuto per la matrice $J^B_{B_i}$ e si sottrae il termine correttivo espresso nell'equazione (66) per ottenere gli angoli a_{1_si} e b_{1_si}, come in (65)

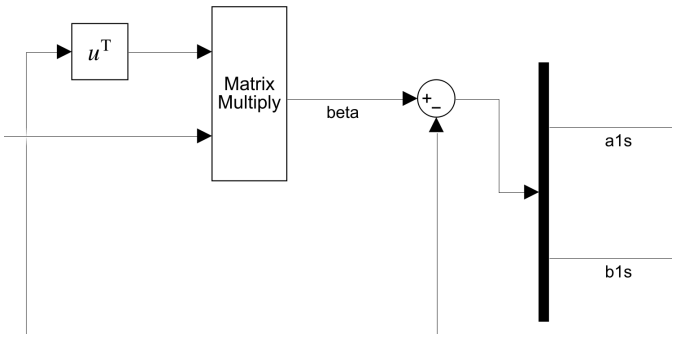


Figura 26: Implementazione eq.(65).

Infine si calcolano le forze F_{xyz} secondo la formula (57) e i momenti M_{xyz} come la differenza fra M_i in (58) e Q_i in (51).

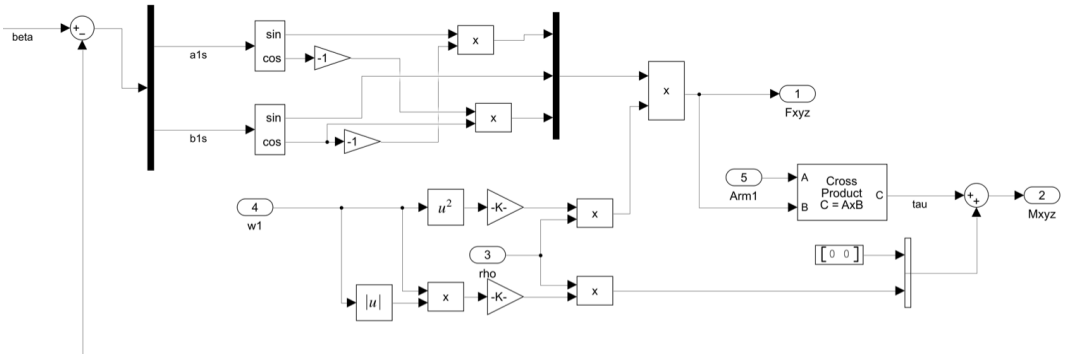


Figura 27: Implementazione delle equazioni (57) e (58).

Applied Force Calculation Somma la forza di gravità, la forza di resistenza dell'aria e la spinta dei motori, che sono fornite rispettivamente dai blocchi *Gravity Force Calculation*, *Drag Calculation* e *Motor Forces and Torques*.

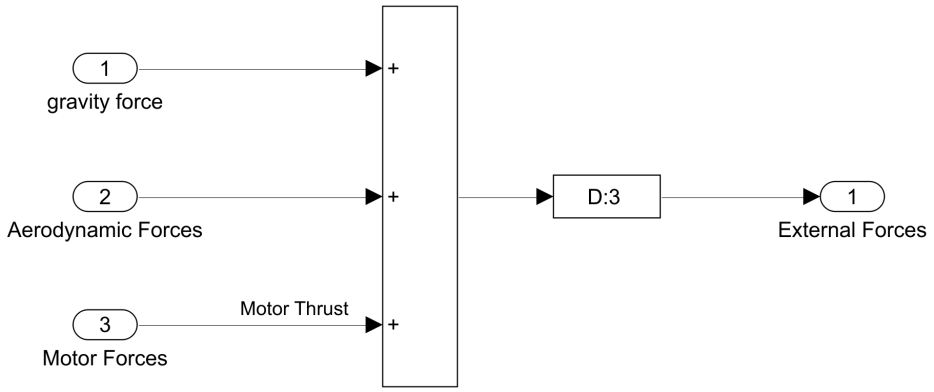


Figura 28: Sottosistema *Applied Force Calculation*.

6DOF L'altro blocco del modello non lineare, che può essere descritto sia con gli angoli di Eulero che con i Quaternioni. Considerando la rappresentazione con gli angoli di Eulero, il blocco *6DOF* implementa le equazioni del moto a sei gradi di libertà, considerando la rotazione del sistema *Body Frame* rispetto al sistema *Earth Frame*. Si ipotizza, come fatto in precedenza, che le forze applicate agiscano nel centro di gravità del drone e che la massa e l'inerzia siano costanti.

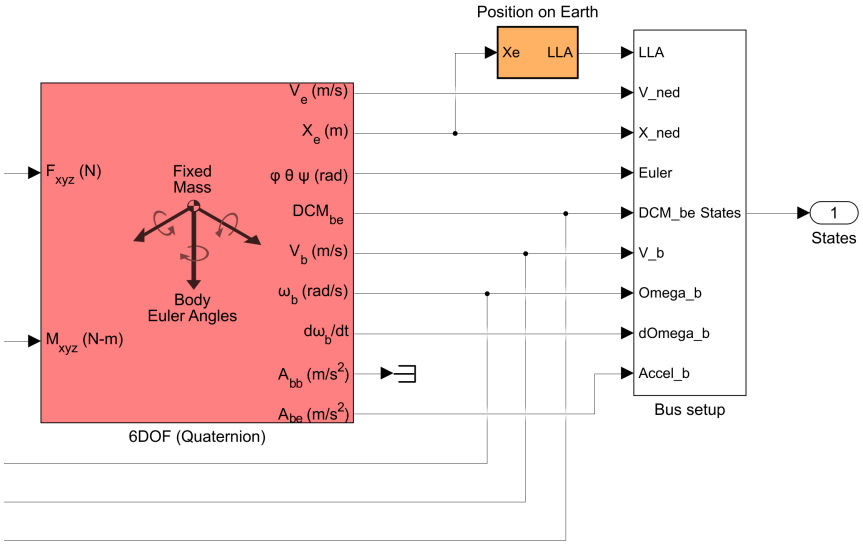


Figura 29: Blocco *6DOF*.

Gli ingressi F_{xyz} e M_{xyz} costituiscono le forze e i momenti applicati. Per quanto riguarda le uscite del blocco, esse rappresentano:

- V_e (m/s): velocità traslazionale rispetto all'*Earth Frame*;
- X_e (m): posizione rispetto all'*Earth Frame*;
- φ θ ψ (rad): angoli di Eulero di *roll*, *pitch* e *yaw*;
- DCM_{be}: matrice di rotazione *Earth Frame* - *Body Frame*;
- V_b (m/s): velocità traslazionale nel *Body Frame*;
- ω_b (rad/s): velocità angolare nel *Body Frame*;
- $d\omega_b/dt$: accelerazione angolare nel *Body Frame*, in rad/s^2;
- A_{be} (m/s^2): accelerazione traslazionale rispetto all'*Earth Frame*.

Il blocco *Position on Earth* consente di convertire la posizione del drone rispetto all'*Earth Frame* in coordinate geodetiche, ossia latitudine, longitudine e altitudine. La sigla LLA sta infatti per Latitude, Longitude, Altitude. Come si vede in figura 30, gli ingressi del blocco *Flat Earth to LLA* sono X_e, menzionato in precedenza, e h_{ref}, che è l'altezza di riferimento dalla superficie terrestre, rispetto all'*Earth Frame*, mentre le sue uscite sono μ e l, che

sono la latitudine e la longitudine geodetiche espresse in gradi, e h, che rappresenta la differenza di altitudine rispetto a quella di riferimento data in ingresso, espressa in metri.

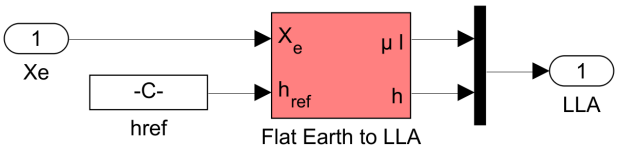


Figura 30: Blocco *Position to Earth*.

Il modello lineare è strutturato come il tipico modello in spazio di stato e sarà analizzato nel dettaglio in seguito.

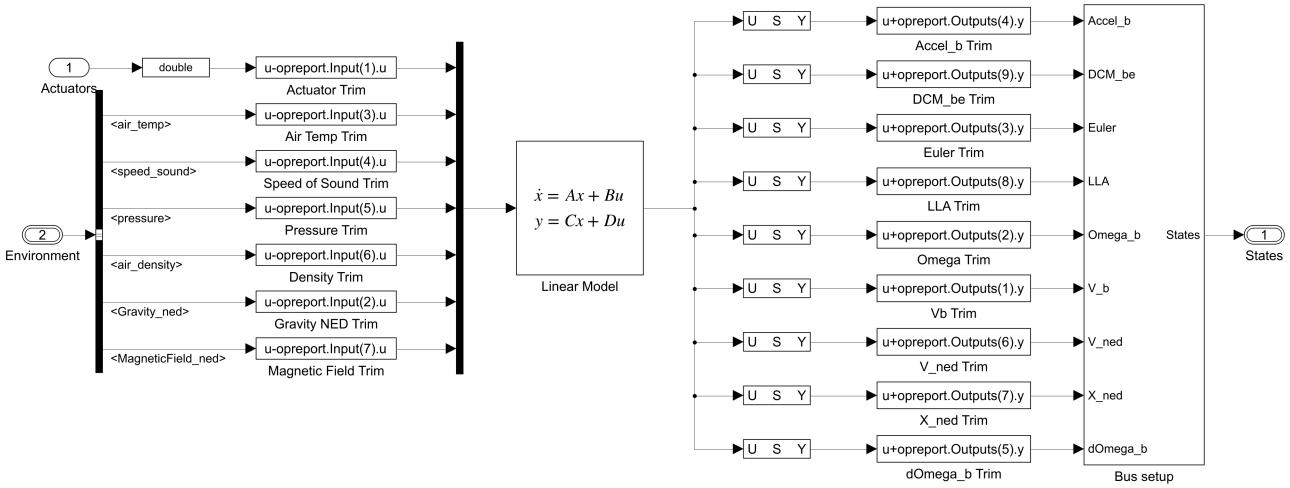


Figura 31: Modello lineare.

3.2 Environment

Nel blocco *Environment* in 32 è possibile scegliere, attraverso il valore della variabile VSS_ENVIRONMENT, se usare variabili di ambiente costanti oppure variabili che dipendono dalla posizione. In questo caso si considerano le variabili costanti, visto che la gravità e la pressione dell'aria non variano tra il suolo e l'altezza di volo considerata. In altre circostanze, come nel caso in cui si desideri determinare l'altezza di volo massima del drone, è invece opportuno scegliere le variabili dipendenti dalla posizione.

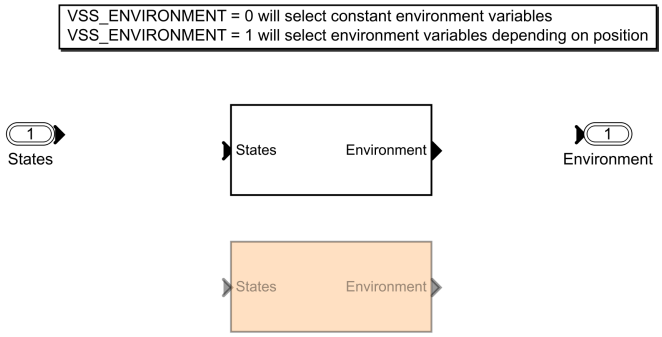


Figura 32: Blocco *Environment*.

Tra i parametri considerati in figura 33 si notano l'accelerazione di gravità rispetto al sistema inerziale e la densità dell'aria ρ.

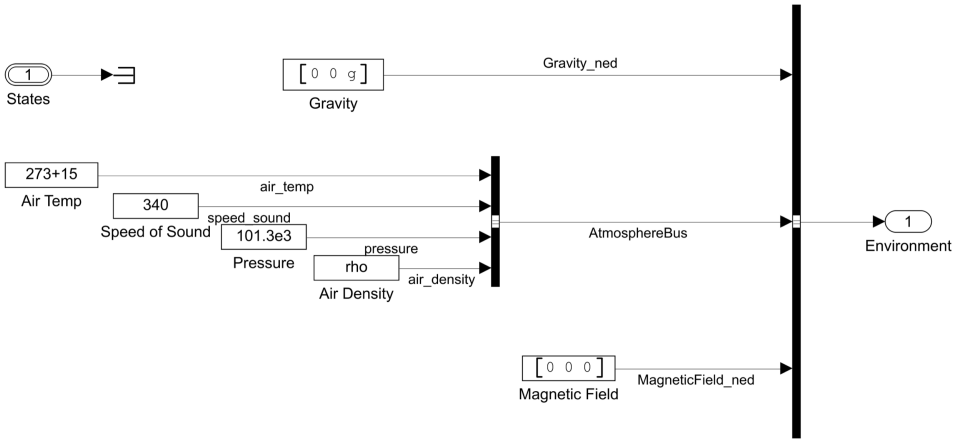


Figura 33: Blocco *Environment (Constant)*.

3.3 Sensors

Le due alternative tra cui scegliere all'interno del blocco *Sensors* in 34 sono *feedthrough sensors* e sensori dinamici con rumore. Per gli scopi della simulazione i sensori devono comportarsi il più possibile come nel caso reale, pertanto si sceglie la seconda opzione, impostando la variabile VSS_SENSORS a 1.

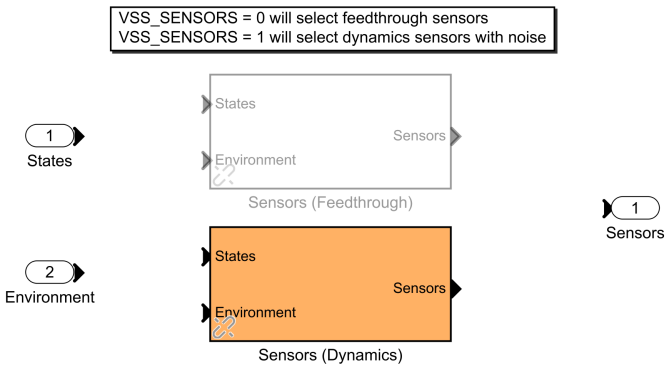


Figura 34: Blocco *Sensors*.

In questo sottosistema sono presenti i dati di calibrazione del sensore e un blocco chiamato *Sensor System* che contiene i modelli della telecamera, dell'unità IMU e dei sensori a ultrasuoni e di pressione:

- L'unità di misura inerziale (IMU) misura le accelerazioni traslazionali e le velocità angolari;
- La telecamera, rivolta verso il basso, viene utilizzata per la stima del flusso ottico;
- Il sensore a ultrasuoni è necessario per la misura dell'altitudine.

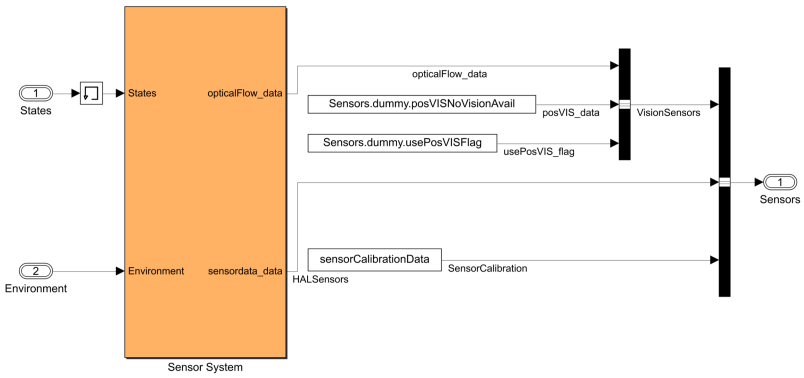


Figura 35: Sensori dinamici con rumore.

Il blocco *Camera* è una rappresentazione approssimata della funzione "optical flow(vx, vy, vz)" immessa nella funzione in C *rsedu_optical_flow()* nel mini drone dal codice interno e inaccessibile del firmware Parrot.

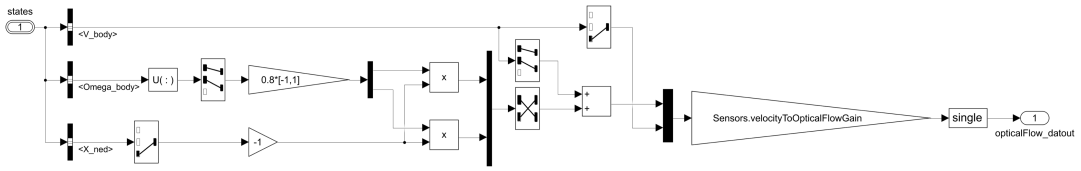


Figura 36: Blocco *Camera*.

Il blocco *IMU pressure* si presenta il questo modo

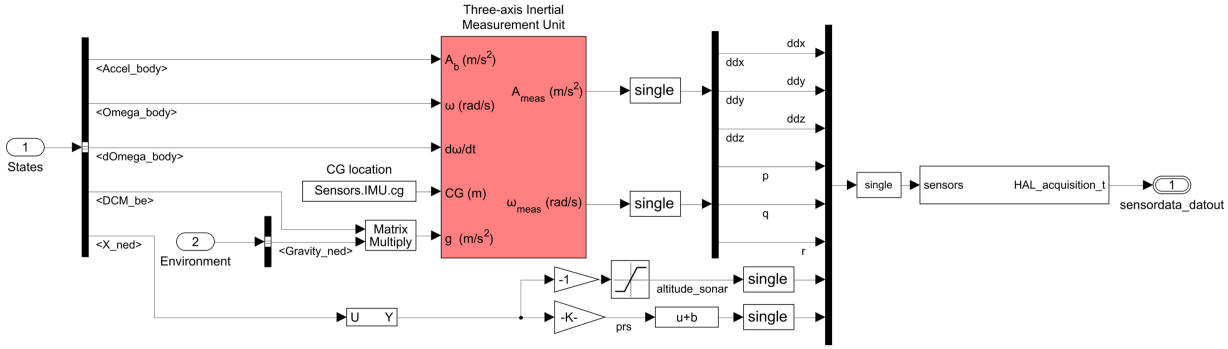


Figura 37: Blocco *IMU pressure*.

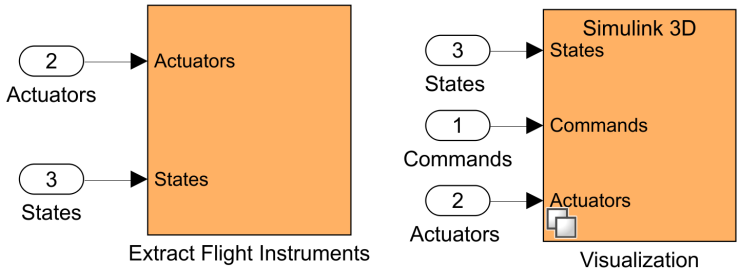
Il blocco *Three-Axis Inertial Measurement Unit* implementa l'unità di misura inerziale, contenente un accelerometro e un giroscopio a tre assi. Esso presenta i seguenti ingressi:

- A_b (m/s^2): accelerazione effettiva rispetto al *Body Frame*;
- ω (rad/s): velocità angolare nel *Body Frame*;
- $d\omega/dt$ (rad/s^2): accelerazione angolare nel *Body Frame*;
- CG (m): posizione del centro di gravità;
- g (m/s^2): accelerazione di gravità nel *Body Frame*.

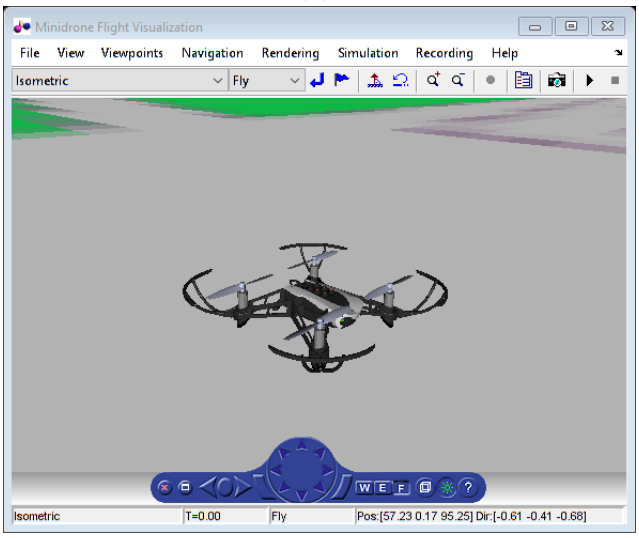
mentre le due uscite sono l'accelerazione lineare misurata dall'accelerometro A_{meas} in m/s^2 e la velocità angolare misurata dal giroscopio ω_{meas} in rad/s. Entrambe si riferiscono al sistema di coordinate inerziale.

3.4 Visualization

Questo sottosistema ha il compito di rappresentare visivamente il comportamento del drone, in quanto consente di generare una visualizzazione grafica del drone e dell'ambiente circostante durante la simulazione.



(a)



(b)

Figura 38: Sottosistema *Visualization* (in alto) e simulazione grafica (in basso).

3.5 Command

Questo sottosistema permette di generare il riferimento di posizione, relativo agli assi x, y, z, e il riferimento di orientazione, riguardante gli angoli di *roll*, *pitch* e *yaw*, in modo tale da imporre al drone il movimento lungo una particolare traiettoria.

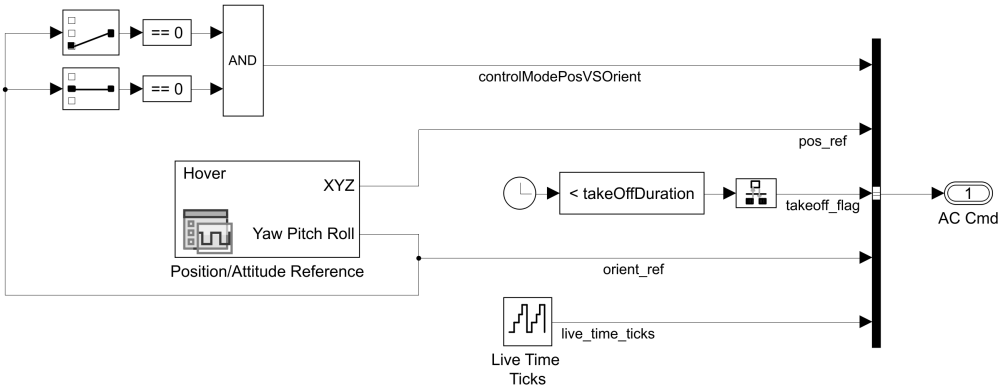


Figura 39: Sottosistema *Command*.

Il blocco *Position/Attitude Reference* consente di creare i riferimenti desiderati, i quali sono poi inseriti nel bus come *pos_ref* e *orient_ref*, che contengono rispettivamente i segnali di riferimento per x, y, z e per *roll*, *pitch* e *yaw*. La variabile *takeoff_flag* assume valore 1 per tutta la durata del decollo del drone. Quest'ultima di default è impostata a un secondo ed è memorizzata nella variabile *takeOffDuration*. Infine la variabile logica *controlModePosVSOrient* determina se utilizzare i riferimenti di posizione o quelli di orientazione. Nel caso in cui i valori assegnati a *pitch* e *roll* in *Position/Attitude Reference* siano entrambi nulli, la variabile *controlModePosVSOrient* assume valore 1.

3.6 Flight Control System

Il blocco *Flight Control System*, abbreviato con *FCS*, è responsabile del controllo dei sei gradi di libertà caratteristici del quadricottero, di cui solo quattro di questi, ossia *roll*, *pitch*, *yaw* e z, si possono controllare direttamente, essendo il drone un sistema sottoattuato. Questo sottosistema contiene quindi tutta la logica di controllo del drone: sulla base dei segnali di riferimento e dei dati raccolti dai sensori, calcola lo sforzo di controllo da fornire al drone.

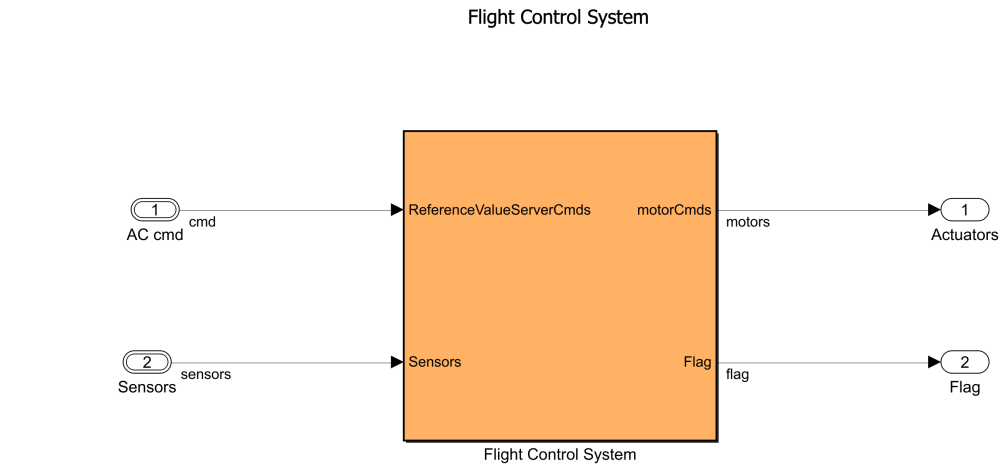


Figura 40: Blocco *FCS*.

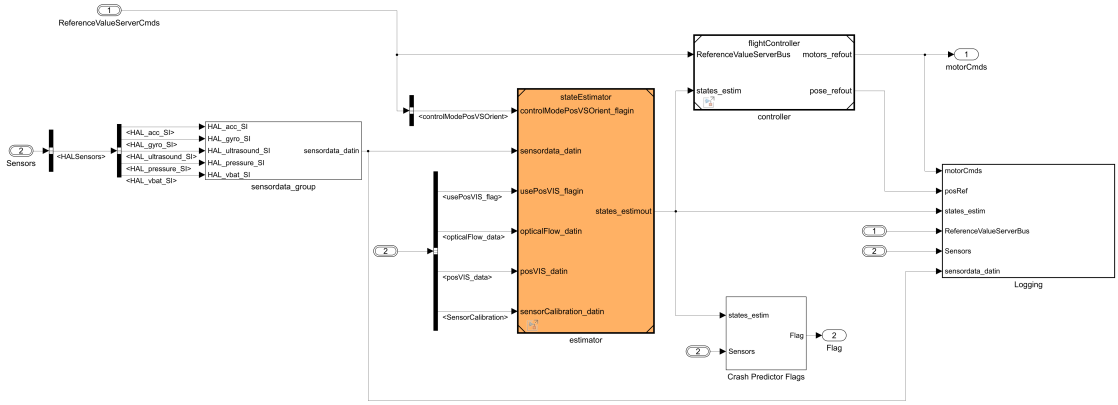


Figura 41: Blocco *Flight Control System*.

sensordata_group Il blocco in questione, mostrato in 42, raggruppa i dati provenienti dall'accelerometro, dal giroscopio e dai sensori di pressione e a ultrasuoni. In particolare:

- HAL_acc_SI contiene l'accelerazione lineare del drone nelle coordinate x, y, z;
- HAL_gyro_SI contiene la velocità angolare del drone attorno agli assi x, y, z;

- HAL_pressure_SI contiene i valori del sensore pressione del drone;
- HAL_ultrasound_SI contiene i valori di altitudine del drone;
- HAL_vbat_SI contiene i dati relativi al voltaggio e alla percentuale di carica della batteria al litio.

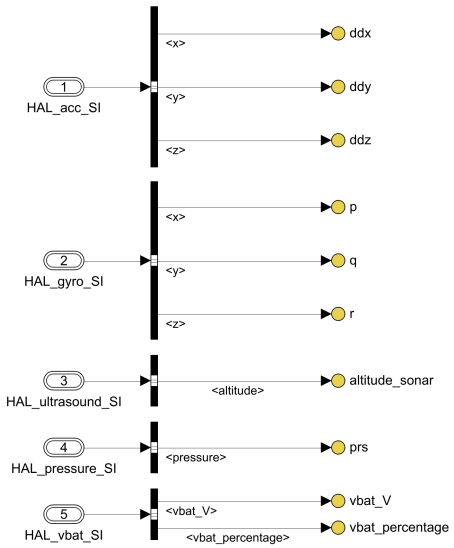


Figura 42: Blocco *sensordata_group*.

Logging E' il blocco che consente di memorizzare e visualizzare i dati relativi alla simulazione di volo. I dati relativi allo sforzo di controllo dei motori, ai segnali di riferimento, alla stima del valore delle variabili di stato in uscita da *estimator* e ai sensori possono essere visualizzati al termine della simulazione Matlab tramite blocchi *Scope* oppure possono essere salvati nei file di log al termine dalla simulazione di volo sul drone reale. Tramite questi file è possibile analizzare il valore di tutte le variabili di stato durante la simulazione di volo e determinare la causa di un eventuale malfunzionamento.

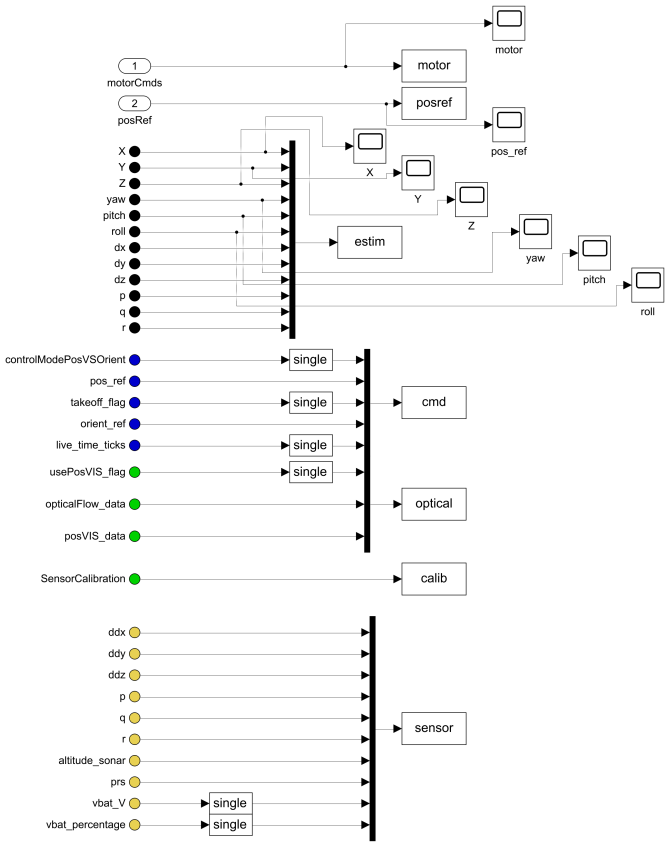


Figura 43: Blocco *Logging*.

Crash Predictor Flag Il sottosistema in questione serve per calcolare una variabile di flag che indica se il volo del drone procede in modo opportuno. Infatti se il drone effettua manovre pericolose e indesiderate, tale flag termina forzatamente la simulazione ed evita danni alla struttura hardware durante il volo.

estimator Fornisce al controllore una stima del valore delle variabili di stato, ossia le componenti del vettore in (25), a partire dai dati dei sensori. Questo sottosistema è costituito dai seguenti blocchi:

- **Sensor Preprocessing**: effettua il preprocessamento dei dati provenienti dall'unità di misura inerziale e dei dati relativi allo yaw dell'unità ottica.

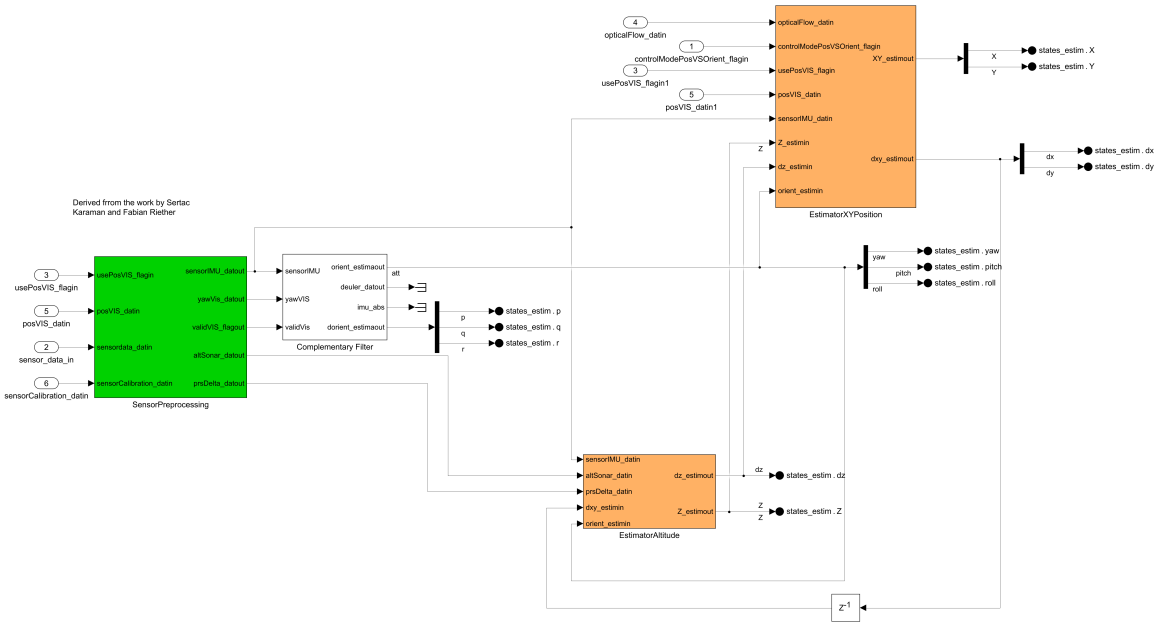


Figura 44: Blocco *estimator*.

- **Complementary Filter**: utilizza i dati provenienti dal blocco *Sensor Preprocessing* per ottenere la posizione angolare del drone, cioè gli angoli di *roll*, *pitch*, *yaw* e le relative variazioni nel tempo p, q, r. Il filtro complementare utilizza i dati dell'accelerometro e del giroscopio: per ottenere la posizione angolare con il giroscopio è necessario integrare la velocità angolare nel tempo. A causa dell'integrazione nel tempo, la misura tende a deviare, non tornando a zero quando il sistema è tornato nella posizione originale, pertanto i dati del giroscopio sono affidabili nel breve periodo. D'altra parte, per ottenere la posizione angolare con l'accelerometro, si determina la posizione del vettore forza di gravità, sempre visibile sull'accelerometro, attraverso la funzione arcotangente. Visto che l'accelerometro misura tutte le forze agenti sul dispositivo, si deve verificare che il valore misurato sia compatibile con il valore della forza di gravità. Se tale valore è troppo grande o troppo piccolo, si tratta di un disturbo di cui non si deve tenere conto. Il filtro complementare utilizza i dati del giroscopio nel breve termine, poiché è molto preciso e non è suscettibile alle forze esterne, mentre a lungo termine vengono considerati i dati dell'accelerometro.

- **Estimator Altitude**: elabora i dati provenienti dal blocco *Sensor Preprocessing* e dal sonar e, confrontandoli con i valori dell'orientamento del drone, determina la quota e la velocità lungo l'asse z del veicolo.

- **Estimator XY Position**: elabora i dati provenienti dal sensore ottico e dai tre blocchi precedentemente descritti per determinare la posizione e la velocità lungo gli assi x e y del drone.

controller Il sottosistema *controller* in 45 sviluppa la vera e propria azione di controllo, prendendo in ingresso i valori di riferimento e la stima dei valori delle variabili di stato, e fornendo in uscita lo sforzo di controllo relativo a ciascun motore.

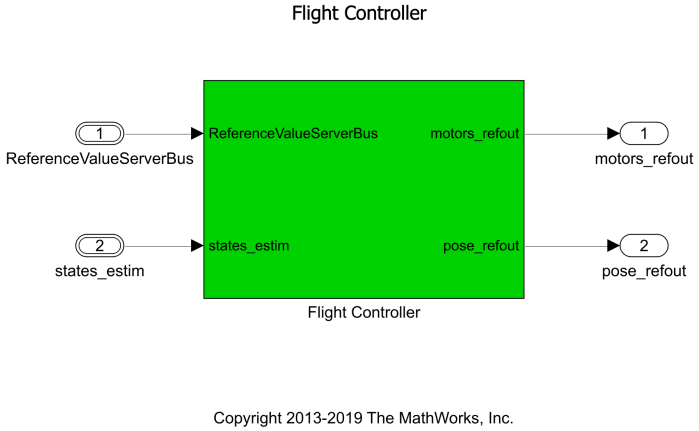


Figura 45: Blocco *controller*.

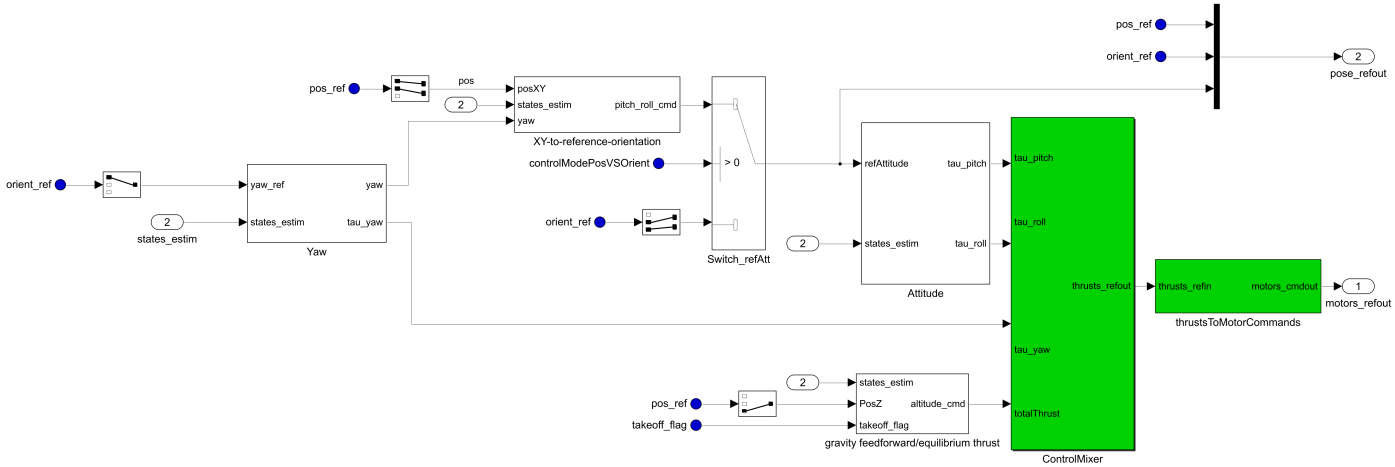


Figura 46: Blocco *Flight Controller*.

Yaw Piccole variazione dello yaw potrebbero determinare elevati errori di posizionamento sugli assi x e y, per questo lo yaw necessita di un controllo

preciso. *yaw_ref* è il riferimento di yaw richiesto per la traiettoria desiderata, mentre <*yaw*> e <*r*> provengono dal blocco *State Estimator* e sono la misura stimata, rispettivamente, dello yaw e della velocità angolare relativa allo yaw. Il controllore PD riduce l'errore di yaw e fornisce l'uscita *tau_yaw*, ossia lo sforzo di controllo relativo a tale grado di libertà.

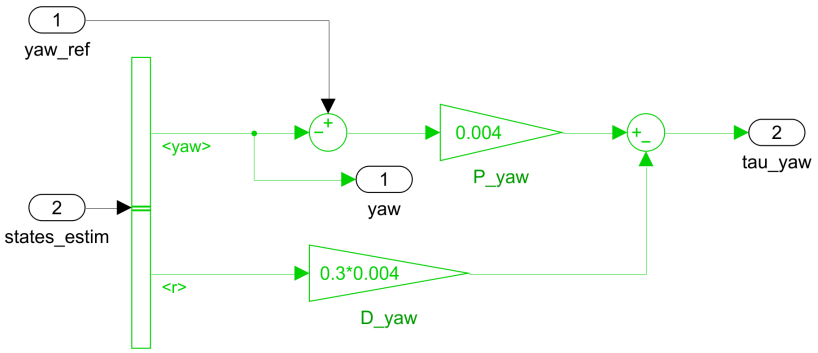


Figura 47: Blocco *Yaw*.

XY-to-reference-orientation Questo sottosistema è responsabile del controllo della posizione del drone lungo gli assi x e y. Per far muovere il quadricottero nel piano x-y come desiderato, la posizione relativa a tali assi deve essere controllata in modo da garantire un errore minimo. Anche in questo caso si usa un controllore PD, il quale considera anche il valore misurato dello *yaw* che, come detto in precedenza, influenza l'errore di posizione. Visto che il drone è un sistema sottoattuato, la posizione lungo x e y non può essere controllata direttamente. Perciò questo sottosistema calcola il riferimento di *pitch* e *roll* da utilizzare per il controllo della posizione e lo fornisce al blocco *Switch_refAtt*, come visibile in figura 46. Quest'ultimo verifica il valore della variabile logica *controlModePosVSOrient*, inizializzata in *Command* (3.5): se tale variabile vale 1, viene scelto il controllo della posizione e pertanto il sottosistema *Attitude* riceve in ingresso i riferimenti di *pitch* e *roll* necessari per tale scopo e calcolati da *XY-to-reference-orientation*. Se invece vale 0, significa che in *Command* sono stati forniti riferimenti di orientamento non nulli e che quindi vengono usati tali valori per calcolare lo sforzo di controllo relativo ai gradi di libertà di *pitch* e *roll*, rinunciando al controllo della posizione.

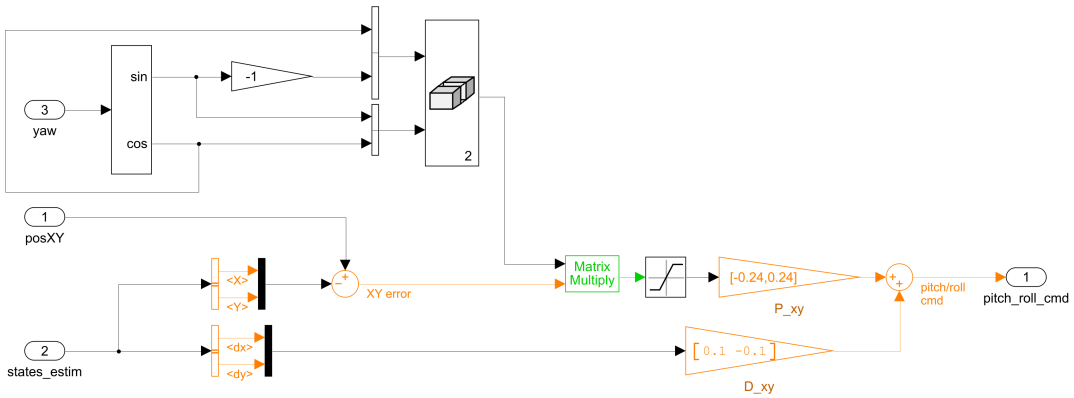


Figura 48: Blocco *XY-to-reference-orientation*.

Attitude Un semplice controllore PID viene usato per controllare gli errori relativi agli angoli di *pitch* e *roll*. Tale controllore migliora sia la risposta transitoria che quella a regime permanente, senza compromettere la stabilità del quadricottero. In 49, il segnale *refAttitude* contiene i riferimenti desiderati per *pitch* e *roll* mentre i valori stimati sono indicati con <*pitch*> e <*roll*>. L'errore ottenuto viene moltiplicato al guadagno *P_pr* e simultaneamente fornito ad un anello contenente un integratore, in modo da ottenere *tau_pitch* e *tau_roll*, ossia i comandi di controllo relativi alle coppie di *pitch* e di *roll*, i quali vengono forniti poi al *Control Mixer*.

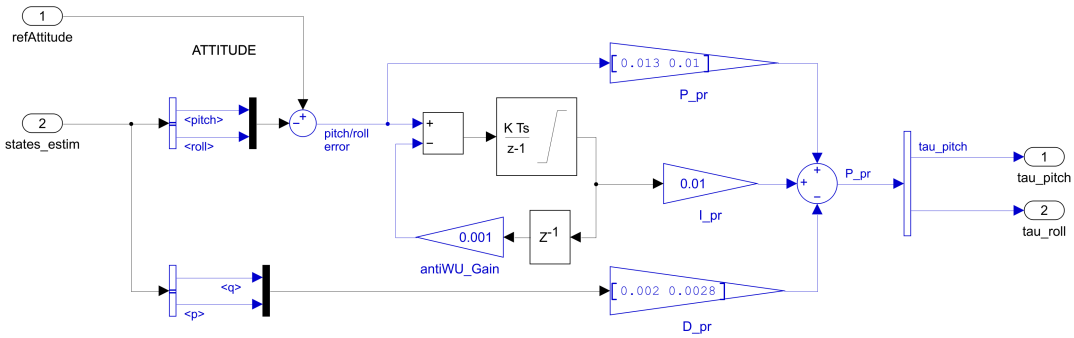


Figura 49: Blocco *Attitude*.

gravity feedforward/equilibrium thrust La figura 50 mostra il blocco *gravity feedforward/equilibrium thrust*, responsabile del controllo dell'altitudine del drone. Visto che il quadricottero possiede una massa non nulla, è necessario considerare la forza gravitazionale che si oppone alla spinta verticale e sottrarre tale forza dalla spinta. Quando la variabile *takeoff_flag* risulta essere 1, lo switch seleziona l'ingresso relativo al *takeoff_gain*. Ciò

avviene per facilitare il decollo del drone. Dopo un secondo, che è la durata del decollo impostata, lo switch passa al controllo tramite PD, il quale serve per ridurre l'errore di posizione lungo l'asse z. Al valore di riferimento, viene sottratto il valore stimato di z ottenuto dallo *State Estimator*, ottenendo così l'errore, che viene amplificato da P_z e a cui viene sottratta la velocità stimata dz, amplificata da D_z. L'uscita del controllore PD è il comando di spinta proporzionale all'errore di posizione.

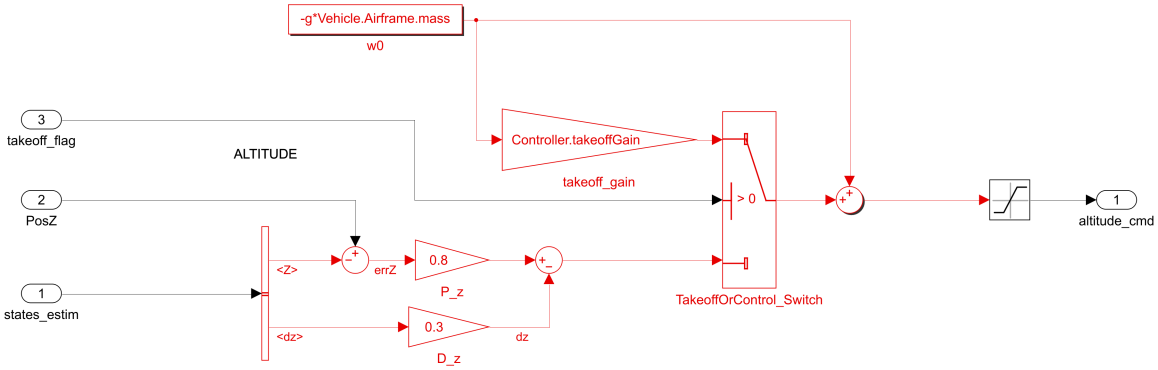


Figura 50: Blocco *gravity feedforward/equilibrium thrust*.

Control Mixer Calcola le velocità angolari di ciascun rotore, richieste per generare la forza di spinta e i momenti necessari per modificare l'assetto del drone in volo. Come visto in precedenza (equazioni (57) e (58)), la spinta e i momenti del quadricottero sono direttamente proporzionali al quadrato della velocità angolare dei rotori. Moltiplicando gli ingressi *totalThrust*, *tau_yaw*, *tau_pitch*, *tau_roll* per la matrice *Controller.Q2Ts* si ottiene appunto la velocità angolare Ω_i relativa a ciascun motore, come si vede in 51.

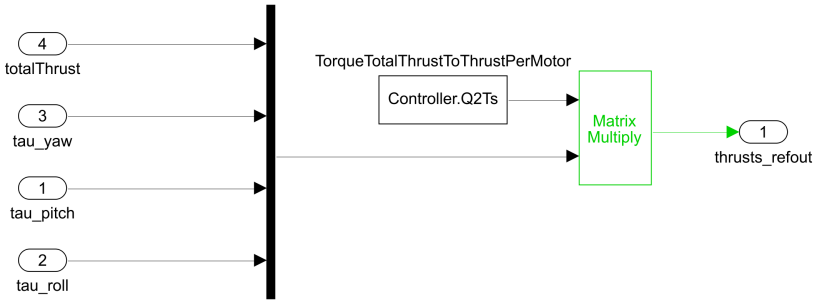


Figura 51: Blocco *ControlMixer*.

Thrusts To Motor Commands I valori di uscita del blocco *Control Mixer* sono a loro volta gli ingressi di *Thrusts To Motor Commands*, dove sono

presenti un gain e un saturatore per proteggere i motori da eventuali sforzi mal proporzionati. Inoltre il secondo blocco gain in figura 52 regola la direzione di rotazione dei motori, in quanto due di essi ruotano in senso orario mentre gli altri due in senso antiorario.

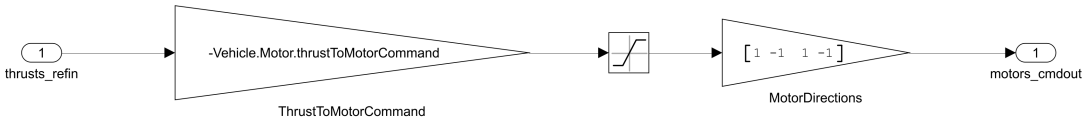


Figura 52: Blocco *thrustsToMotorCommands*.

4 Linearizzazione

Il modello non lineare analizzato nei capitoli precedenti rappresenta in maniera sufficientemente adeguata il comportamento del drone reale. Tuttavia è molto complicato implementare un controllore per un determinato grado di libertà basandosi su tale modello. Pertanto è necessario effettuare una linearizzazione del modello non lineare per ottenere una funzione di trasferimento $P(s)$, che rappresenta il processo da controllare. Tale linearizzazione si effettua rispetto ad un punto di equilibrio, supponendo che il drone effettui movimenti limitati intorno al punto di volo stazionario. Si presenta ora la procedura di linearizzazione generale per un sistema non lineare e stazionario, per poi considerare nel paragrafo 4.1 il modello lineare fornito dal progetto Simulink.
Dato il seguente sistema non lineare e stazionario, in generale MIMO:

$$\begin{cases} \dot{x}(t) = f(x(t), u(t)) \\ y(t) = g(x(t), u(t)) \end{cases} \tag{68}$$

in cui il vettore $x(t)$ contiene le variabili di stato, $u(t)$ le variabili di ingresso e $y(t)$ quelle di uscita, il procedimento di linearizzazione consiste nel descrivere il comportamento di un sistema non lineare attorno all'equilibrio nominale, mediante un particolare sistema lineare, il quale è un'approssimazione del sistema originario. Dopo avere definito un punto di lavoro $(\bar{x}, \bar{y})$, ossia il punto di volo stazionario, che soddisfa le condizioni di equilibrio nominale rispetto ad un ingresso costante $u(t) = \bar{u}$:

$$\begin{cases} 0 = f(\bar{x}, \bar{u}) \\ \bar{y} = g(\bar{x}, \bar{u}) \end{cases} \tag{69}$$

si descrivono le variazioni delle variabili di ingresso, stato, uscita e dello stato iniziale, rispetto ai valori di equilibrio. Si ottiene quindi:

$$\begin{cases} u(t) = \bar{u} + \delta u(t) \\ x(t) = \bar{x} + \delta x(t) \\ y(t) = \bar{y} + \delta y(t) \\ x_{t_0} = \bar{x} + \delta x_{t_0} \end{cases} \tag{70}$$

Sostituendo le equazioni di (70) in (68), e considerando che la derivata dello stato di equilibrio $\bar{x}$ è nulla, si ricava:

$$\begin{cases} \delta\dot{x}(t) = f(\bar{x} + \delta x(t), \bar{u} + \delta u(t)) \\ \bar{y} + \delta y(t) = g(\bar{x} + \delta x(t), \bar{u} + \delta u(t)) \end{cases} \tag{71}$$

mentre, per quanto riguarda la condizione iniziale:

$$\bar{x} + \delta x(t_0) = \bar{x} + \delta x_{t_0} \Rightarrow \delta x(t_0) = \delta x_{t_0} \tag{72}$$

Supponendo che le funzioni f e g siano sufficientemente regolari, esse possono essere sviluppate in serie di Taylor, troncate al primo ordine, rispetto a x ed u, nell'intorno di $x = \bar{x}$ e $u = \bar{u}$.

$$\delta\dot{x}(t) = f(\bar{x}, \bar{u}) + \left.\frac{\partial f(x,u)}{\partial x}\right|_{x=\bar{x},u=\bar{u}} \delta x(t) + \left.\frac{\partial f(x,u)}{\partial u}\right|_{x=\bar{x},u=\bar{u}} \delta u(t) \tag{73}$$

$$\bar{y} + \delta y(t) = g(\bar{x}, \bar{u}) + \left.\frac{\partial g(x,u)}{\partial x}\right|_{x=\bar{x},u=\bar{u}} \delta x(t) + \left.\frac{\partial g(x,u)}{\partial u}\right|_{x=\bar{x},u=\bar{u}} \delta u(t) \tag{74}$$

Tenendo conto delle equazioni (69):

$$\begin{cases} \delta\dot{x}(t) = A\delta x(t) + B\delta u(t) \\ \delta y(t) = C\delta x(t) + D\delta u(t) \end{cases} \tag{75}$$

in cui:

$$A = \left.\frac{\partial f(x,u)}{\partial x}\right|_{x=\bar{x},u=\bar{u}} \quad B = \left.\frac{\partial f(x,u)}{\partial u}\right|_{x=\bar{x},u=\bar{u}} \quad C = \left.\frac{\partial g(x,u)}{\partial x}\right|_{x=\bar{x},u=\bar{u}} \quad D = \left.\frac{\partial g(x,u)}{\partial u}\right|_{x=\bar{x},u=\bar{u}} \tag{76}$$

Il sistema ottenuto in (75) è il sistema lineare e stazionario che lega le variazioni prime delle variabili in gioco. Le variabili del sistema lineare indicano le differenze tra le coordinate del sistema non lineare e quelle del punto di equilibrio. Il sistema lineare descrive in maniera approssimata il comportamento del sistema non lineare attorno al particolare punto di equilibrio considerato, nel caso in cui le variazioni delle funzioni di ingresso $\delta u(t)$ e dello stato iniziale δx_{t_0}, nonché le variazioni dello stato $\delta x(t)$ e dell'uscita $\delta y(t)$ da esse provocate, siano sufficientemente piccole in norma. Infatti, sotto questa ipotesi, è valida l'approssimazione derivante dall'aver trascurato i termini di ordine superiore al primo nello sviluppo in serie di Taylor.

4.1 Modello lineare

Assegnando il valore 0 alla variabile d'ambiente VSS_VEHICLE è possibile selezionare il modello lineare fornito di default dal progetto Simulink, che consente di determinare la funzione di trasferimento del sottosistema relativo all'angolo di *roll*, che è il grado di libertà che si è deciso di controllare. Il modello lineare viene ottenuto tramite la funzione *trimLinearizeOpPoint*, fornita da *Simulink Control Design*. Questa funzione consente di creare un sistema lineare del velivolo rispetto a un punto di lavoro. Tale sistema viene salvato nella variabile chiamata *linsys*, che può essere importata nel Workspace manualmente, attraverso il file .mat denominato *linearizedAirframe*, all'interno della cartella *linearAirframe*, oppure scrivendo il comando *load('linearizedAirframe.mat')* nella Command Window. La variabile *linsys* contiene le matrici A, B, C e D del modello lineare in spazio di stato e i nomi delle variabili di ingresso u, di stato x e di uscita y. Il sistema presenta 14 variabili di ingresso, 12 variabili di stato e 33 variabili di uscita. Le matrici del modello lineare sono:

- la matrice A è la matrice della dinamica del sistema e ha dimensione 12×12;
- la matrice B è la matrice degli ingressi del sistema, di dimensione 12×14;
- la matrice C è la matrice delle uscite del sistema, di dimensione 33×12;
- infine la matrice D è la matrice che mette in relazione le uscite con gli ingressi e ha dimensione 33×14.

0	3.6611E-17	0	0	0	0	0	0	0	1	3.0292E-23	-5.2296E-12	phi
-3.6611E-17	0	0	0	0	0	0	0	0	0	1	5.2295E-12	theta
-6.8347E-13	-2.0323E-28	0	0	0	0	0	0	0	0	-5.2295E-12	1	psi
0	-9.8100	0	-3.4402E-09	-3.4402E-09	6.8532E-13	0	0	0	-5.4616E-11	0.1380	-3.2848E-13	uv
9.8100	-2.8478E-22	0	-3.6595E-09	-3.6595E-09	-6.8520E-13	0	0	0	-0.1380	5.8098E-11	-3.5159E-13	vb
5.4456E-11	5.4456E-11	0	-6.8347E-13	6.8347E-13	0	0	0	0	3.2920E-13	3.5087E-13	0	wb
1.7216E-24	1.3478E-19	3.2920E-13	1	-1.4969E-18	-5.2296E-12	0	0	0	0	0	0	xe
-1.3478E-19	2.0176E-37	3.5087E-13	1.4969E-18	1	5.2295E-12	0	0	0	0	0	0	ye
-3.2920E-13	-3.5087E-13	0	5.2296E-12	-5.2295E-12	1	0	0	0	0	0	0	ze
0	0	0	-5.7590E-08	-5.7590E-08	-2.7287E-14	0	0	0	-2.17153	9.1429E-10	4.3301E-13	p
0	0	0	4.0369E-08	4.0369E-08	-2.1644E-14	0	0	0	6.4089E-10	-1.6192	-5.1363E-13	q
0	0	0	-1.4434E-19	-1.2831E-19	3.6185E-19	0	0	0	1.2157E-13	1.1242E-13	-6.8807E-09	r

Figura 53: Matrice A.

0	0	0	0	0	0	0	0	0	0	0	0	0	0	Actuators(1)
0	0	0	0	0	0	0	0	0	0	0	0	0	0	Actuators(2)
0	0	0	0	0	0	0	0	0	0	0	0	0	0	Actuators(3)
-6.8995E-17	6.900E-17	-6.900E-17	6.8995E-17	-9.9331E-14	0	0	0	1.0000	1.4969E-18	5.2296E-12	0	0	0	Actuators(4)
6.7628E-17	-6.763E-17	6.763E-17	-6.7628E-17	9.8151E-14	0	0	0	-1.4969E-18	1.0000	-5.2295E-12	0	0	0	AtmosphereBus.air_density
-0.0096	0.0096	-0.0096	0.0096	-8.2855	0	0	0	0	0	1.0000	0	0	0	AtmosphereBus.air_temp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	AtmosphereBus.pressure
0	0	0	0	0	0	0	0	0	0	0	0	0	0	AtmosphereBus.speed_sound
0	0	0	0	0	0	0	0	0	0	0	0	0	0	Gravity_ned(1)
0.420192	0.4202	-0.4202	-0.4202	2.5288E-14	0	0	0	0	0	0	0	0	0	Gravity_ned(2)
0.3133	-0.3133	-0.3133	0.3133	1.8856E-14	0	0	0	0	0	0	0	0	0	Gravity_ned(3)
-0.0115	-0.0115	-0.0115	-0.0115	-7.9371E-16	0	0	0	0	0	0	0	0	0	MagneticField_ned(1)

Figura 54: Matrice B.

Accel_body(1)	0	-9.8100	0	-3.4402E-09	-3.4402E-09	1.8445E-15	0	0	0	-5.4616E-11	0.1380	7.2109E-16	phi
Accel_body(2)	9.8100	-2.8478E-22	0	-3.6595E-09	-3.6595E-09	-1.7339E-15	0	0	0	-0.1380	5.8098E-11	-7.2109E-16	theta
Accel_body(3)	5.4456E-11	5.4456E-11	0	0	0	0	0	0	0	0	0	0	psi
DCM_be(1)	0	5.5511E-12	0	0	0	0	0	0	0	0	0	0	uv
DCM_be(2)	-5.2296E-12	-5.2295E-12	-1	0	0	0	0	0	0	0	0	0	vb
DCM_be(3)	1.4969E-18	1	-5.2295E-12	0	0	0	0	0	0	0	0	0	wb
DCM_be(4)	0	8.3096E-30	1	0	0	0	0	0	0	0	0	0	xe
DCM_be(5)	5.5511E-12	0	0	0	0	0	0	0	0	0	0	0	ye
DCM_be(6)	-1	1.4969E-18	-5.2296E-12	0	0	0	0	0	0	0	0	0	ze
DCM_be(7)	0	-1	0	0	0	0	0	0	0	0	0	0	p
DCM_be(8)	1	-2.9029E-23	0	0	0	0	0	0	0	0	0	0	q
DCM_be(9)	5.5511E-12	5.5511E-12	0	0	0	0	0	0	0	0	0	0	r
Euler(1)	1	0	0	0	0	0	0	0	0	0	0	0	
Euler(2)	0	1	0	0	0	0	0	0	0	0	0	0	
Euler(3)	0	0	1	0	0	0	0	0	0	0	0	0	
LLA(1)	0	0	0	0	0	0	9.0026E-06	0	0	0	0	0	
LLA(2)	0	0	0	0	0	0	0	1.2127E-05	0	0	0	0	
LLA(3)	0	0	0	0	0	0	0	0	-1	0	0	0	
Omega_body(1)	0	0	0	0	0	0	0	0	0	1	0	0	
Omega_body(2)	0	0	0	0	0	0	0	0	0	0	1	0	
Omega_body(3)	0	0	0	0	0	0	0	0	0	0	0	1	
V_body(1)	0	0	0	1	0	0	0	0	0	0	0	0	
V_body(2)	0	0	0	0	1	0	0	0	0	0	0	0	
V_body(3)	0	0	0	0	0	1	0	0	0	0	0	0	
V_ned(1)	1.7216E-24	1.3478E-19	3.2920E-13	1	-1.4969E-18	-5.2296E-12	0	0	0	0	0	0	
V_ned(2)	-1.3478E-19	2.0176E-37	3.5087E-13	1.4969E-18	1	5.2295E-12	0	0	0	0	0	0	
V_ned(3)	-3.2920E-13	-3.5087E-13	0	5.2296E-12	-5.2295E-12	1	0	0	0	0	0	0	
X_ned(1)	0	0	0	0	0	0	1	0	0	0	0	0	
X_ned(2)	0	0	0	0	0	0	0	1	0	0	0	0	
X_ned(3)	0	0	0	0	0	0	0	0	1	0	0	0	
dOmega_body(1)	0	0	0	-5.7590E-08	-5.7590E-08	-2.7287E-14	0	0	0	-2.17152838	9.1429E-10	4.3301E-13	
dOmega_body(2)	0	0	0	4.0369E-08	4.0369E-08	-2.1644E-14	0	0	0	6.4089E-10	-1.6192	-5.1363E-13	
dOmega_body(3)	0	0	0	-1.4434E-19	-1.2831E-19	3.6185E-19	0	0	0	1.2157E-13	1.1242E-13	-6.8807E-09	

Figura 55: Matrice C.

Accel_body(1)	-6.8995E-17	6.8995E-17	-6.8995E-17	6.8995E-17	-9.9331E-14	0	0	0	1	1.4969E-18	5.2296E-12	0	0	0	Actuators(1)
Accel_body(2)	6.7628E-17	-6.7628E-17	6.7628E-17	-6.7628E-17	9.8151E-14	0	0	0	-1.4969E-18	1	-5.2295E-12	0	0	0	Actuators(2)
Accel_body(3)	-0.0096	0.0096	-0.0096	0.0096	-8.2855	0	0	0	0	0	1	0	0	0	Actuators(3)
DCM_be(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Actuators(4)
DCM_be(2)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	AtmosphereBus.air_density
DCM_be(3)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	AtmosphereBus.air_temp
DCM_be(4)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	AtmosphereBus.pressure
DCM_be(5)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	AtmosphereBus.speed_sound
DCM_be(6)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Gravity_ned(1)
DCM_be(7)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Gravity_ned(2)
DCM_be(8)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Gravity_ned(3)
DCM_be(9)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	MagneticField_ned(1)
Euler(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Euler(2)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Euler(3)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
LLA(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
LLA(2)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
LLA(3)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Omega_body(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Omega_body(2)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Omega_body(3)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
V_body(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
V_body(2)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
V_body(3)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
V_ned(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
V_ned(2)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
V_ned(3)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
X_ned(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
X_ned(2)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
X_ned(3)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
dOmega_body(1)	0.4202	0.4202	-0.4202	-0.4202	0	0	0	0	0	0	0	0	0	0	
dOmega_body(2)	0.3133	-0.3133	-0.3133	0.3133	0	0	0	0	0	0	0	0	0	0	
dOmega_body(3)	-0.0115	-0.0115	-0.0115	-0.0115	-1.3323E-15	0	0	0	0	0	0	0	0	0	

Figura 56: Matrice D.

A partire da queste matrici si può estrarre il sottosistema desiderato del modello lineare, in questo caso è quello relativo all'angolo di *roll*. Per fare ciò si considerano:

- in A, i coefficienti relativi alle variabili di stato φ e p;
- in B, i coefficienti relativi alle variabili di stato φ e p, rispetto agli ingressi $Actuators(1), (2), (3), (4)$;
- in C, i coefficienti dell'uscita $Euler(1)$, corrispondente proprio all'angolo di *roll*, relativi alle variabili di stato φ e p;
- in D, i coefficienti relativi all'uscita $Euler(1)$, rispetto agli ingressi $Actuators(1), (2), (3), (4)$.

Si ottiene pertanto il seguente sottosistema, considerando, senza perdita di generalità, un singolo attuatore. I coefficienti sono evidenziati in rosso nelle matrici 53, 54, 55 e 56.

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ 0 & -2.17153 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 0 \\ 0.420192 \end{bmatrix} \qquad \mathbf{C} = \begin{bmatrix} 1 & 0 \end{bmatrix} \qquad \mathbf{D} = \begin{bmatrix} 0 \end{bmatrix} \tag{77}$$

Per ricavare la funzione di trasferimento relativa alla variabile di stato considerata per l'implementazione dei controllori è necessario tenere conto di due costanti moltiplicative, K_1 e K_2: la prima determina la partizione del segnale di controllo tra i diversi motori; la seconda rappresenta il coefficiente di spinta dei motori. La costante K_1 si ricava dal blocco *Control Mixer*, in particolare dalla matrice *Controller.Q2Ts*, riportata di seguito:

```
>> Controller.Q2Ts

ans =

    0.2500  103.5736   -5.6659   -5.6659
    0.2500 -103.5736   -5.6659    5.6659
    0.2500  103.5736    5.6659    5.6659
    0.2500 -103.5736    5.6659   -5.6659
```

Figura 57: Matrice *Controller.Q2Ts*.

La prima colonna fornisce i coefficienti relativi alla spinta verticale *totalThrust*, la seconda quelli riguardanti il momento di yaw *tau_yaw*, la terza i coefficienti che si riferiscono al momento di pitch *tau_pitch*, infine la quarta

colonna quelli relativi al momento di roll *tau_roll*. K_1 è pertanto uguale al coefficiente della quarta colonna, considerando anche in questo caso la dinamica di un singolo motore.

```
>> k1 = 5.6659

k1 =

    5.6659
```

Figura 58: Calcolo della costante K_1.

La costante K_2 è pari al coefficiente di spinta *Vehicle.Motor.thrustToMotorCommand*, contenuto nel primo blocco gain di *thrustToMotorCommand* in 52.

```
>> Vehicle.Motor.thrustToMotorCommand

ans =

   1.5307e+03

>> k2 = 1530.7

k2 =

   1.5307e+03
```

Figura 59: Calcolo della costante K_2.

In definitiva, la funzione di trasferimento del sottosistema relativo all'angolo di *roll* si ottiene come:

$$P(s) = K_1 \cdot K_2 \cdot \mathbf{C} \cdot (s\mathbf{I} - \mathbf{A})^{-1} \cdot \mathbf{B} \tag{78}$$

che in Matlab si implementa nel seguente modo:

```
>> A = [0 1; 0 -2.17153];
B = [0; 0.420192];
C = [1 0];
D = 0;
sys = ss(A,B,C,D);
P = tf(sys);
k1 = 5.6659;
k2 = 1530.7;
Ps = P*k1*k2

Ps =

      3644
  -------------
  s^2 + 2.172 s

Continuous-time transfer function.
```

Figura 60: Calcolo della funzione di trasferimento $P(s)$.

Nei capitoli successivi, dedicati all'implementazione dei controllori per il grado di libertà di *roll*, si utilizza pertanto, come funzione di trasferimento del processo:

$$P(s) = \frac{3644}{s^2 + 2.172s} = \frac{3644}{s(s + 2.172)} \tag{79}$$

5 Controllo Roll

In questa sezione viene presentato inizialmente il sistema di controllo del *rollio* predefinito nel progetto *asbQuadcopter*, basato sui regolatori PID, per poi proseguire con l'implementazione e il confronto delle prestazioni di tre controllori, due tramite sintesi con luogo delle radici e uno con sintesi in frequenza.

5.1 PID

I regolatori PID sono dei regolatori standard che permettono di regolare i parametri del sistema di controllo entro ampi limiti, così da poter essere adattati al particolare sistema di regolazione in cui vengono inseriti. Il loro nome deriva dalle tre azioni svolte da questi dispositivi, rispettivamente azione *Proporzionale*, *Integrale* e *Derivativa*.

$$u(t) = K_P \cdot e(t) + K_I \int_0^t e(\tau)\, d\tau + K_D \cdot \frac{de(t)}{dt} \tag{80}$$

Il controllore viene posto in serie al sistema da controllare e il circuito di retroazione riporta in ingresso l'uscita $y(t)$ che, attraverso il nodo sommatore, viene sottratta al riferimento $r(t)$, ottenendo l'errore di uscita $e(t)$, che costituisce il vero e proprio ingresso del controllore.

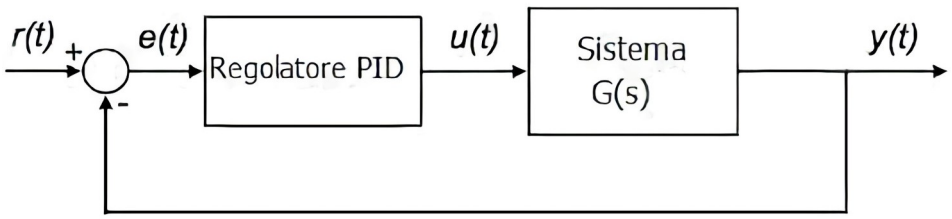


Figura 61: Schema di controllo con regolatore PID.

Azione proporzionale L'ingresso *e(t)* e l'uscita *u(t)* sono legati algebricamente da un coefficiente K_P, detto coefficiente dell'azione proporzionale:

$$u(t) = K_P \cdot e(t) \tag{81}$$

Il significato dell'azione proporzionale sta nel fatto che quanto maggiore è l'errore $e(t)$ all'ingresso del controllore tanto maggiore sarà l'azione di controllo svolta dallo stesso regolatore. L'azione proporzionale ha l'effetto di

diminuire il tempo di salita e di ridurre, ma non eliminare, l'errore a regime permanente. D'altra parte, però, incrementa le sovraelongazioni e, se il guadagno è elevato, diminuisce la stabilità, aumentando le oscillazioni.

Azione integrale Il contributo di questa azione è proporzionale all'integrale dell'errore $e(t)$ e il coefficiente dell'azione integrale K_I definisce la costante di tempo integrale T_I:

$$u(t) = K_I \cdot \int_0^t e(\tau)\, d\tau \qquad T_I = \frac{K_P}{K_I} \tag{82}$$

L'azione integrale assicura errore nullo a regime permanente per variazioni a gradino del riferimento $r(t)$. In generale, l'azione integrale è associata all'azione proporzionale, in modo tale da realizzare i cosiddetti controllori PI (Proporzionale - Integrale), i quali permettono, scegliendo opportunamente le costanti, di ottenere una maggiore precisione (incremento del tipo del sistema retroazionato) senza peggiorare il grado di stabilità del sistema, unita ad una maggiore velocità di risposta.
La taratura della costante di integrazione è legata alle sovraelongazioni ed alle oscillazioni che si possono innescare. Infatti cambiamenti improvvisi del carico possono portare il sistema verso l'instabilità, quando il coefficiente K_I dell'azione integrale non è scelto in modo opportuno. Il termine integrale introduce uno sfasamento di 90° in ritardo, corrispondente ad un polo in termini di trasformata di Laplace, che porta ad un peggioramento dei margini di stabilità del sistema ad anello chiuso, limitando il valore del guadagno integrale.

Azione derivativa Fornisce in uscita la derivata rispetto al tempo dell'errore $e(t)$ e il coefficiente dell'azione derivativa K_D definisce la costante di tempo derivativa T_D:

$$u(t) = K_D \cdot \frac{de(t)}{dt} \qquad T_D = \frac{K_D}{K_P} \tag{83}$$

L'uscita di un controllore derivativo non dipende dall'errore presente o passato, ma dalla velocità con cui varia l'errore. Inoltre, l'azione derivativa è complementare all'azione integrale perché fornisce un anticipo di fase di 90°, in quanto porta all'introduzione di uno zero nell'origine in termini di trasformata di Laplace. L'azione derivativa produce una riduzione delle oscillazioni che si verificano in occasione dei transitori, contribuendo a stabilizzare il sistema, ma introduce l'inconveniente di amplificare i segnali con contenuto

armonico a frequenze elevate, che in genere corrispondono al rumore elettromagnetico sovrapposto al segnale utile. In aggiunta, in presenza di una variazione a gradino del segnale di riferimento, il termine derivativo, e di conseguenza la variabile di controllo $u(t)$, assumerebbero un andamento di tipo impulsivo. Questa brusca variazione è in contrasto con il requisito di moderazione del controllo e può provocare la saturazione dell'attuatore e l'allontanamento del sistema dalla condizione di linearità, con riferimento alla quale normalmente si progetta il regolatore. Per queste ragioni l'azione derivativa è frequentemente esercitata sulla sola variabile di uscita $y(t)$, in quanto le sue variazioni istantanee sono in genere contenute e la presenza dell'azione derivativa non provoca il suddetto andamento impulsivo di $u(t)$.
Ulteriori approfondimenti sui regolatori PID si possono trovare in [20].

Controllori PID per il rollio Il progetto *asbQuadcopter*, come detto in 3.6, presenta di default controllori PID per i gradi di libertà di *pitch* e *roll*. Le prestazioni di tali controllori, di seguito riportate rispetto alla simulazione Matlab, non risultano essere soddisfacenti.

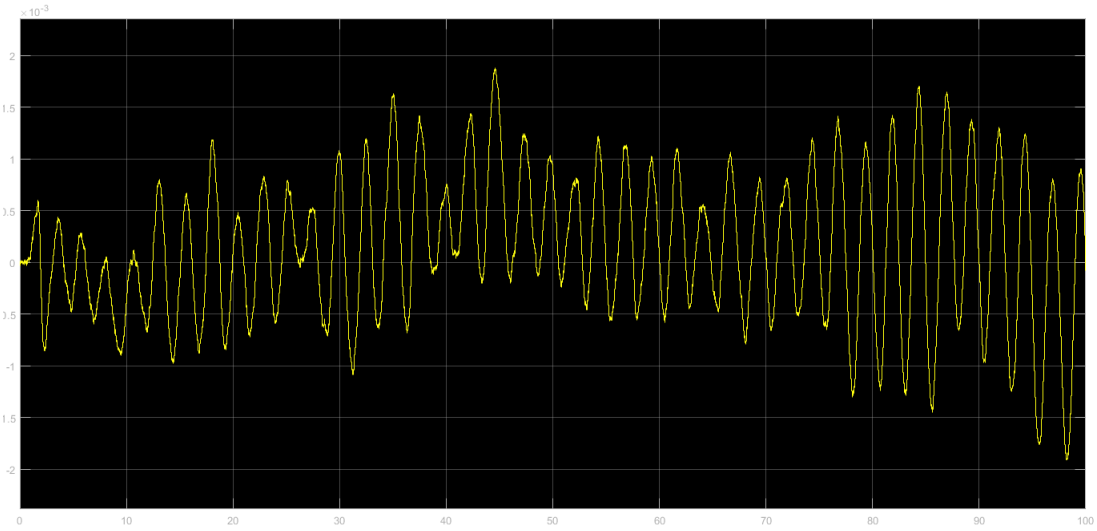


Figura 62: Grafico Roll con controllore PID.

Sebbene dalla 62 si può notare che le oscillazioni del *roll* siano dell'ordine di grandezza di 10^{-3}, nelle simulazioni sul drone reale il controllore PID non garantisce un comportamento adeguato sia per il volo stazionario, sia per il volo con movimenti lungo gli assi x e y.
Lo scopo dei successivi paragrafi è quello di implementare controllori che forniscano prestazioni soddisfacenti nei due tipi di volo menzionati sopra, utilizzando le tecniche di sintesi tramite luogo delle radici e in frequenza.

5.2 Sintesi tramite luogo delle radici

La prima tecnica di sintesi utilizzata per l'implementazione del controllore per il grado di libertà di *roll* è il luogo delle radici.
Come spiegato in [1], il luogo delle radici è un procedimento grafico mediante il quale è possibile stabilire correlazioni fra i valori dei poli p_i $(i = 1, ..., n)$, degli zeri z_i $(i = 1, ..., m)$, del coefficiente di guadagno K che caratterizzano la funzione razionale di variabile complessa:

$$F(s) = K \cdot \frac{\prod_{i=1}^{m}(s - z_i)}{\prod_{i=1}^{n}(s - p_i)} \tag{84}$$

e i valori delle radici dell'equazione algebrica:

$$\prod_{i=1}^{n}(s - p_i) + K \cdot \prod_{i=1}^{m}(s - z_i) = 0 \tag{85}$$

Poichè le radici dell'equazione (85) coincidono con i poli della funzione razionale:

$$W(s) = \frac{F(s)}{1 + F(s)} \tag{86}$$

il procedimento in questione può venire utilizzato, nello studio di un sistema di controllo in retroazione, per correlare alcune importanti caratteristiche della funzione di trasferimento del sistema reazionato, in particolare i valori dei poli, ai valori dei parametri che caratterizzano la funzione di trasferimento del sistema non reazionato. Il procedimento consiste essenzialmente nell'*individuazione del luogo geometrico* dei punti del piano complesso descritto dalle radici dell'equazione (85) al variare del parametro K. Il luogo geometrico delle radici di (85) corrispondente a valori positivi del parametro K viene detto luogo *positivo* delle radici. Per individuare il luogo delle radici corrispondente ad un'assegnata funzione del tipo (84), è conveniente riscrivere l'equazione (85) nella forma:

$$\prod_{i=1}^{n}(s - p_i) = -K \cdot \prod_{i=1}^{m}(s - z_i) \tag{87}$$

e, successivamente, sostituire a questa la coppia di uguaglianze tra grandezze reali:

$$\prod_{i=1}^{n}|s - p_i| = |K| \cdot \prod_{i=1}^{m}|s - z_i| \tag{88}$$

$$\sum_{i=1}^{n}\angle(s - p_i) = \pi + \angle K + \sum_{i=1}^{m}\angle(s - z_i) + 2\pi h \tag{89}$$

essendo h un qualsiasi numero intero, positivo o negativo. La (88) viene detta condizione di *modulo* o di *ampiezza* mentre la (89) condizione di *argomento* o di *fase*. Riferendosi all'interpretazione delle grandezze complesse del tipo $(s - p_i)$ come vettori aventi per estremi i punti rappresentativi di s e p_i, orientati da p_i verso s (analogamente per $(s - z_i)$), è immediato osservare che la prima di tali condizioni, riscritta come:

$$|K| = \frac{\prod_{i=1}^{n} |s - p_i|}{\prod_{i=1}^{m} |s - z_i|} \tag{90}$$

corrisponde alla proprietà che, in ogni punto del luogo delle radici associato alla funzione $F(s)$, il valore assoluto del parametro K è pari al prodotto delle lunghezze dei vettori che congiungono il punto stesso con i poli della $F(s)$, diviso il prodotto delle lunghezze dei vettori congiungenti il punto stesso con gli zeri della $F(s)$.
La seconda condizione assume due diverse forme, a seconda che il parametro K, che si suppone reale, assuma valore positivo o negativo:

$$\sum_{i=1}^{n} \angle(s - p_i) - \sum_{i=1}^{m} \angle(s - z_i) = \begin{cases} \pi + 2\pi h & \text{(luogo positivo)} \\ 2\pi h & \text{(luogo negativo)} \end{cases} \tag{91}$$

e corrisponde alla proprietà che la somma degli argomenti dei vettori orientati dai poli della funzione $F(s)$ verso un punto del luogo, meno la somma degli argomenti dei vettori orientati dagli zeri della funzione $F(s)$ verso lo stesso punto del luogo, è pari a π o a 0, a meno di un multiplo intero di 2π, a seconda che si tratti del luogo positivo o negativo. In definitiva, la seconda condizione può essere interpretata come l'equazione del luogo allo studio ed essere utilizzata per individuare i punti del piano complesso appartenenti al luogo (*tracciamento* del luogo), mentre la prima condizione può essere utilizzata per far corrispondere, a ciascuno dei punti del luogo così individuato, il valore del parametro K in corrispondenza al quale il punto stesso diventa radice della (85) (*graduazione* del luogo).
Si tralascia qui l'esposizione delle regole di tracciamento del luogo, che sono spiegate dettagliatamente in [1].

Implementazione primo controllore Le specifiche in catena chiusa prese in considerazione sono le seguenti:

- Sistema di tipo 1.
- Errore a regime permanente rispetto ad un ingresso a rampa $|\tilde{e}_1| \leq 0.025$.

- Tutti i poli in catena chiusa con parte reale minore di -0.45.

La specifica sull'errore fornisce un limite inferiore al guadagno statico del controllore, ottenuto come segue:

$$|\tilde{e}_1| = \frac{1}{K_F} \leq 0.025 \Rightarrow \begin{cases} K_F = K_G \cdot K_P \geq 40 \\ K_P = \dfrac{3644}{2.172} \end{cases} \Rightarrow K_G \geq 0.024 \tag{92}$$

La funzione di trasferimento del controllore ha una struttura del tipo:

$$G(s) = \frac{K \cdot (s - z_1)(s - z_2)}{(s - p_1)(s - p_2)} \tag{93}$$

Si posizionano i due zeri in $z_1 = -3$ e $z_2 = -0.38$:

$$G(s) = \frac{K \cdot (s + 3)(s + 0.38)}{(s - p_1)(s - p_2)} \tag{94}$$

$$F(s) = G(s) \cdot P(s) = \frac{3644 \cdot K \cdot (s + 3)(s + 0.38)}{s(s - p_1)(s - p_2)(s + 2.172)} \tag{95}$$

Il centro degli asintoti viene scelto in $s_0 = -22$, quindi si posizionano di conseguenza i poli:

$$s_0 = \frac{p_1 + p_2 - 2.172 + 3 + 0.38}{2} = -22 \Rightarrow \begin{cases} p_1 = -45 \\ p_2 = -0.02 \end{cases} \tag{96}$$

La funzione di trasferimento del controllore risulta essere la seguente:

$$G(s) = \frac{K \cdot (s + 3)(s + 0.38)}{(s + 45)(s + 0.02)} \tag{97}$$

Per scegliere il coefficiente di guadagno K si tiene conto della specifica sull'errore, che fornisce un limite inferiore:

$$K_G = \frac{K \cdot 3 \cdot 0.38}{45 \cdot 0.02} \geq 0.024 \Rightarrow K \geq 0.019 \tag{98}$$

e della specifica riguardante i poli a ciclo chiuso, che fornisce un limite superiore. Per determinare tale limite superiore si analizza il luogo delle radici, tramite il comando *rlocus* di Matlab. Per prima cosa si inseriscono nel Workspace le funzioni di trasferimento $P(s)$, come visto in 60, e $G(s)$, tramite il comando $Gs = zpk([-3 - 0.38], [-45 - 0.02], 1)$, in cui si pone $K = 1$. Quindi

si richiama il comando $rlocus(Gs * Ps)$. Dal luogo delle radici in figura 63 si osserva che deve valere $K \leq 0.0226$. Pertanto in definitiva:

$$0.0190 \leq K \leq 0.0226 \Rightarrow K = 0.02 \tag{99}$$

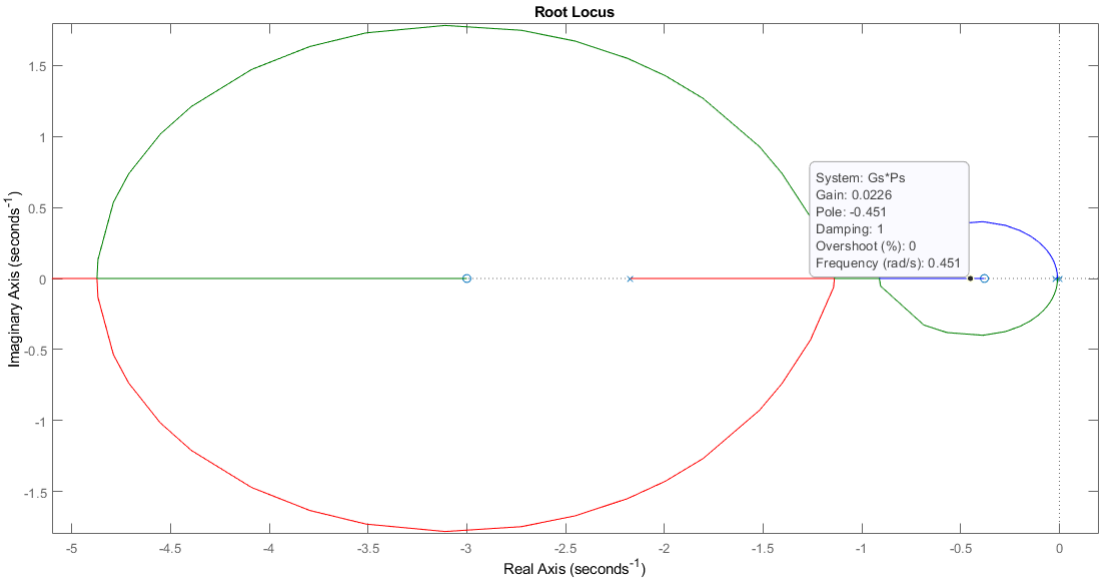


Figura 63: Luogo delle radici $G(s) \cdot P(s)$.

Si ottengono le seguenti funzioni di trasferimento del controllore e in catena aperta:

$$G(s) = \frac{0.02 \cdot (s+3)(s+0.38)}{(s+45)(s+0.02)} \tag{100}$$

$$F(s) = \frac{72.88 \cdot (s+3)(s+0.38)}{s(s+45)(s+0.02)(s+2.172)} \tag{101}$$

Si nota che:

- il sistema è di tipo 1;
- $|\tilde{e}_1| = \dfrac{1}{K_F} = \dfrac{45 \cdot 0.02 \cdot 2.172}{3644 \cdot 0.02 \cdot 3 \cdot 0.38} = 0.023528$;
- la specifica sui poli a ciclo chiuso è verificata.

Per determinare i poli della funzione di trasferimento a ciclo chiuso $W(s)$ si usano i seguenti comandi:

```
>> Ws = feedback(Gs*Ps, 1);
>> pole(Ws)

ans =
          -43.3668 + 0.0000i
          -1.6798 + 1.1389i
          -1.6798 - 1.1389i
          -0.4652 + 0.0000i
```

A questo punto si può analizzare la risposta al gradino del sistema, che si ottiene con il comando $step(Ws)$:

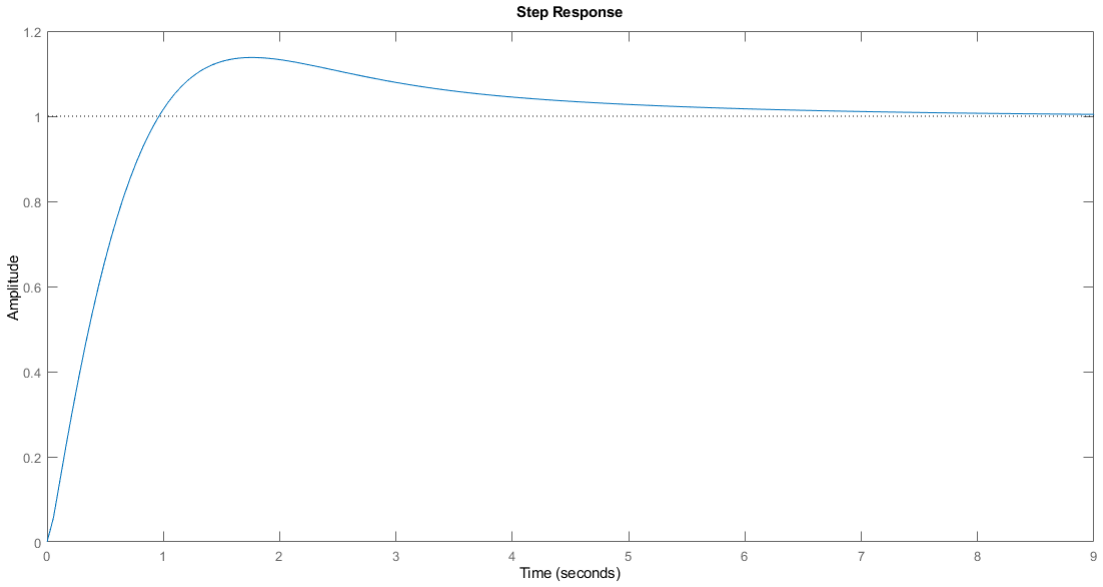


Figura 64: Risposta al gradino del sistema a ciclo chiuso.

Dal grafico 64 risulta:

- sovraelongazione $\hat{s} \simeq 13.8\%$;
- tempo di salita $t_s \simeq 0.96\ s$.

Per poter utilizzare tale controllore in una simulazione reale sul drone è necessario effettuarne la discretizzazione tramite il comando $Gz = c2d(Gs, Ts,' zoh')$, con cui si ottiene la funzione di trasferimento:

$$G(z) = \frac{0.02 \cdot (z - 0.9981)(z - 0.9868)}{(z - 1)(z - 0.7985)} \tag{102}$$

La $G(z)$ viene inserita all'interno del blocco *Attitude* del progetto in modo da poter eseguire il comando *Build, Deploy & Start* per generare l'eseguibile, caricarlo sul drone e avviare l'applicazione sull'hardware.

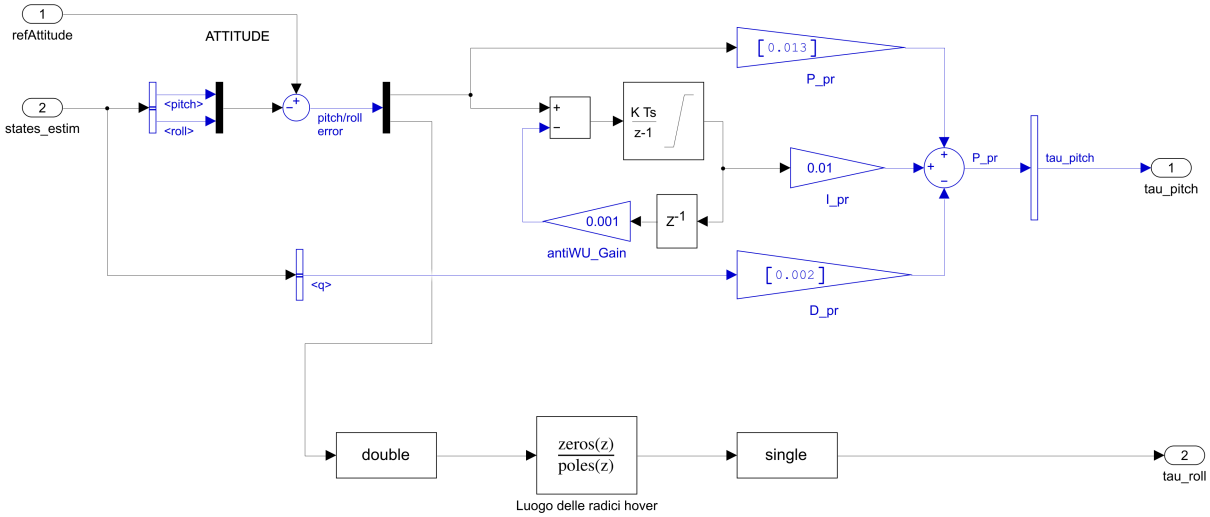


Figura 65: Controllo del *roll* tramite la $G(z)$ (blocco *Attitude*).

Questo controllore è stato provato sia per il volo stazionario che per il volo con una traiettoria nel piano $x - y$. Di seguito vengono riportati i grafici relativi alla simulazione reale del volo stazionario. Si fa notare che nei primi tre grafici, relativi ai gradi di libertà x, y e z, viene riportato in blu l'andamento dei dati, salvati al termine della simulazione, e in rosso il riferimento fornito, come spiegato nella legenda in alto a destra.

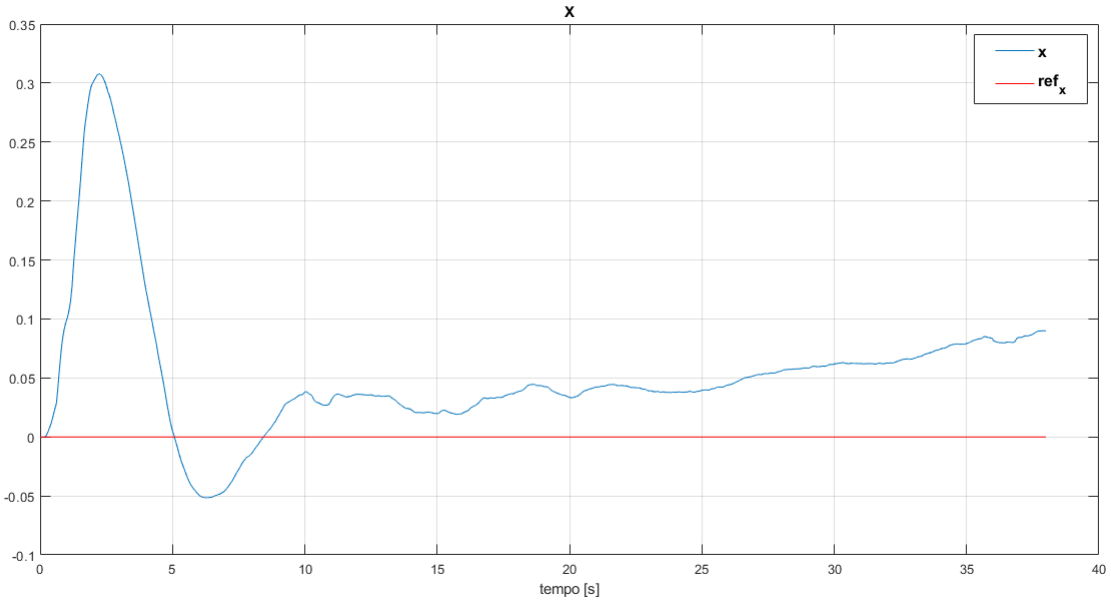


Figura 66: Grafico X.

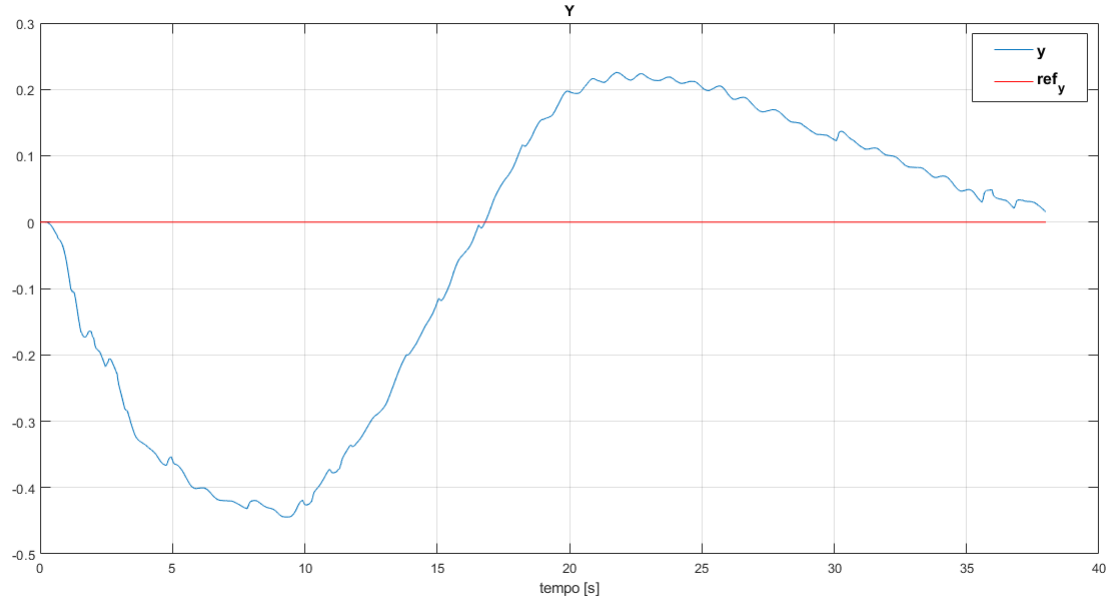


Figura 67: Grafico Y.

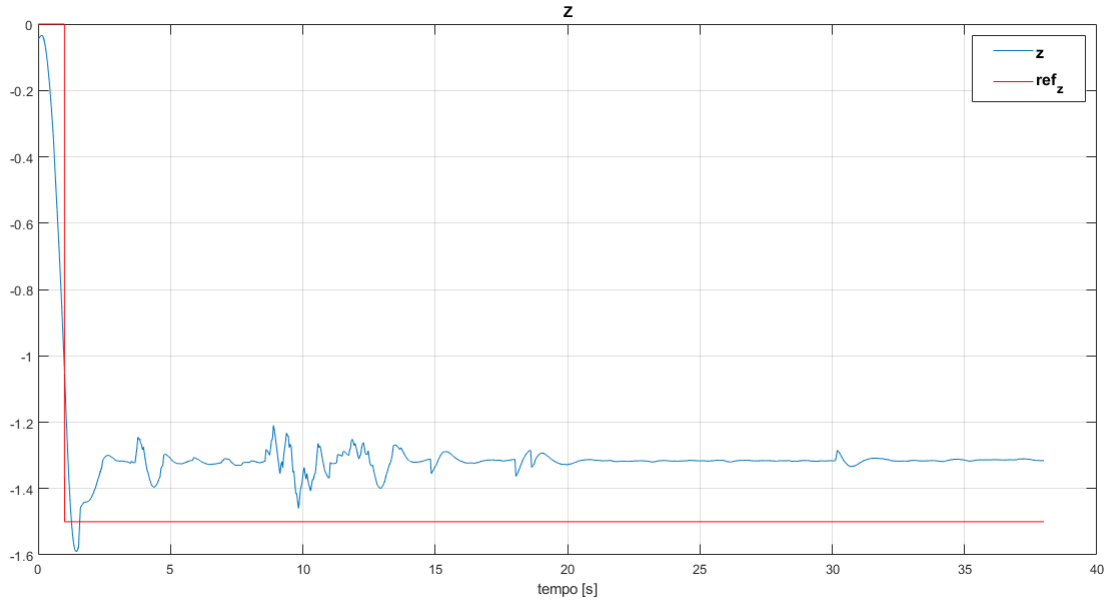


Figura 68: Grafico Z.

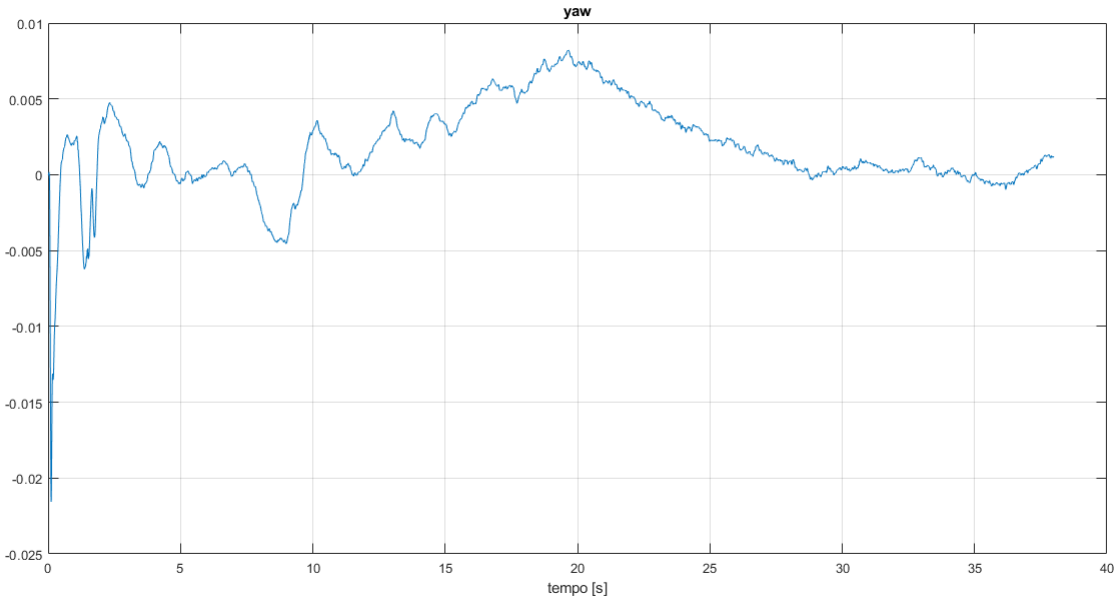


Figura 69: Grafico Yaw.

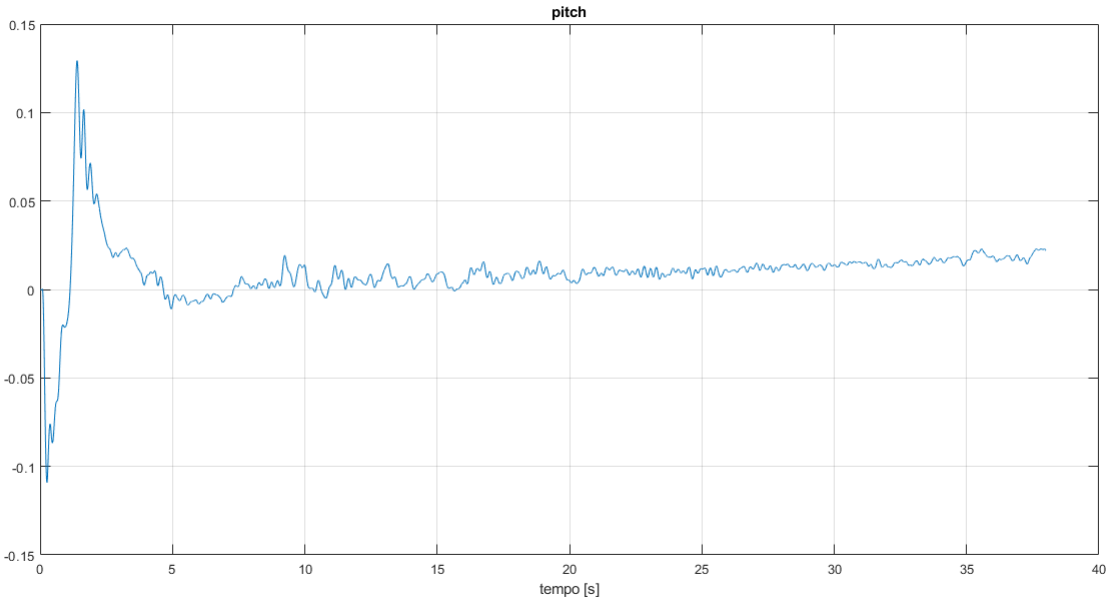


Figura 70: Grafico Pitch.

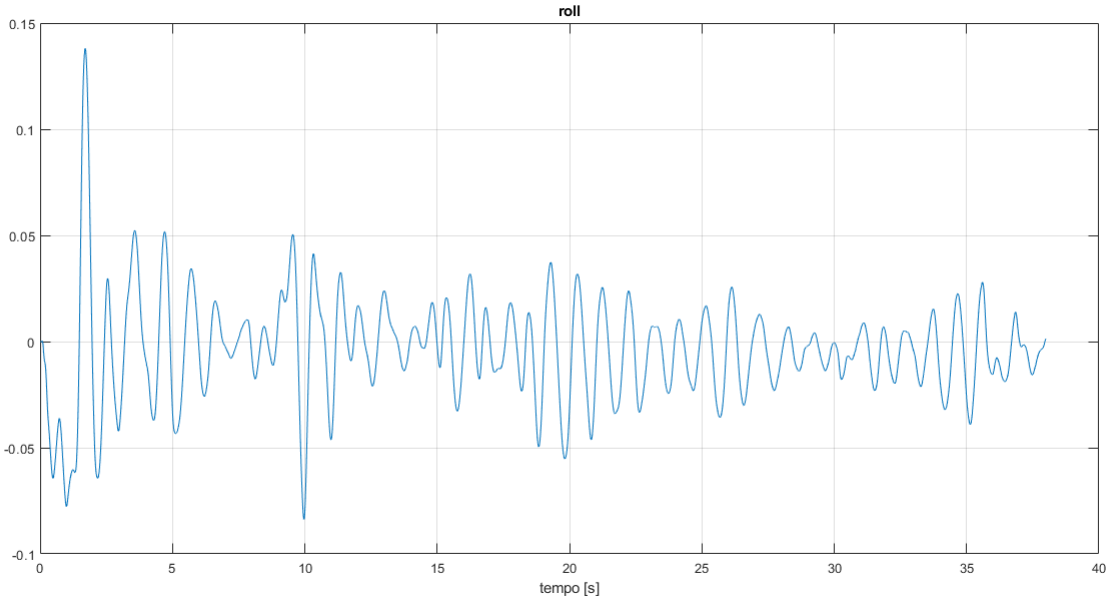


Figura 71: Grafico Roll.

Dal grafico relativo all'angolo di *roll* in 71 si osserva che il comportamento di questo controllore è soddisfacente per il volo nel punto stazionario. Lo stesso controllore è stato usato in una simulazione del volo con riferimenti non nulli lungo gli assi x e y, riportati in figura 72.

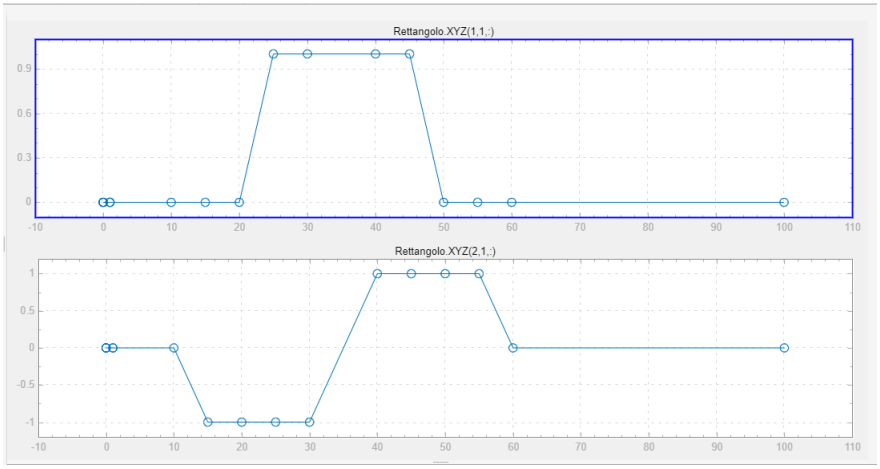
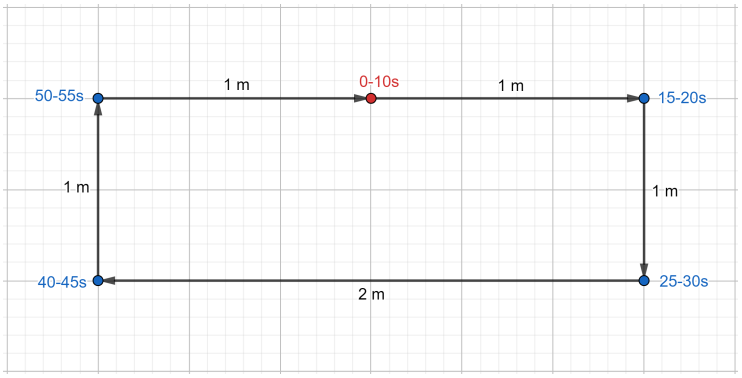
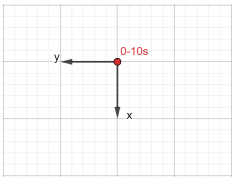


Figura 72: Riferimenti per gli assi x (in alto) e y (in basso).

Con tali riferimenti, il drone segue una traiettoria a "rettangolo", come si può vedere in figura 73a: il punto di decollo e di atterraggio del drone viene indicato dal punto rosso, mentre in blu sono indicati i vertici del rettangolo in cui il drone staziona per 5 secondi. Si considerano inoltre 5 secondi per percorrere una distanza di 1 metro. La simulazione ha una durata di 70 secondi e in corrispondenza di ogni vertice sono indicati gli intervalli di tempo in cui il drone staziona in quel punto.



(a) Traiettoria a "rettangolo".



(b) Orientamento assi cartesiani.

Figura 73: Visualizzazione della traiettoria a "rettangolo".

In questa seconda simulazione sul drone reale si osservano oscillazioni marcate ad ogni cambio di direzione, come visibile nel grafico 74, e si può pertanto concludere che questo controllore non è adatto per il tipo di traiettoria in 72.

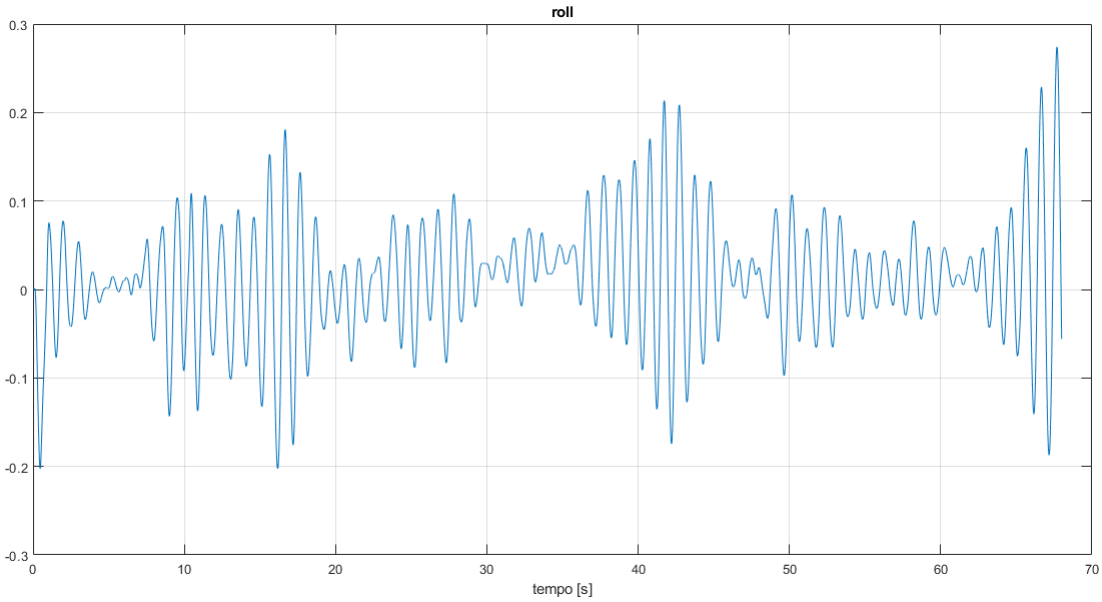


Figura 74: Grafico Roll (simulazione con movimento orizzontale).

Implementazione secondo controllore Per questo motivo è stato implementato un altro controllore, sempre utilizzando la tecnica di sintesi con luogo delle radici, che consente però di avere un comportamento soddisfacente per la simulazione con riferimenti non nulli sul piano orizzontale.
Le specifiche considerate sono le stesse, ad eccezione di quella relativa al posizionamento dei poli a ciclo chiuso. Per questo secondo controllore si desidera che i tutti poli in catena chiusa abbiano parte reale minore di -0.5. In questo modo si otterrà un transitorio più rapido e minori oscillazioni nel momento in cui viene fornito un riferimento non nullo sugli assi $x - y$. Si posizionano i due zeri del controllore in $z_1 = -2.6$ e $z_2 = -0.4$:

$$G(s) = \frac{K \cdot (s + 2.6)(s + 0.4)}{(s - p_1)(s - p_2)} \tag{103}$$

Il centro degli asintoti viene posizionato in $s_0 = -28$:

$$s_0 = \frac{p_1 + p_2 - 2.172 + 2.6 + 0.4}{2} = -28 \Rightarrow \begin{cases} p_1 = -56 \\ p_2 = -0.02 \end{cases} \tag{104}$$

La funzione di trasferimento del controllore risulta essere la seguente:

$$G(s) = \frac{K \cdot (s + 2.6)(s + 0.4)}{(s + 56)(s + 0.02)} \tag{105}$$

Come in precedenza, per scegliere il coefficiente di guadagno K si tiene conto della specifica sull'errore, che fornisce un limite inferiore:

$$K_G = \frac{K \cdot 2.6 \cdot 0.4}{56 \cdot 0.02} \geq 0.024 \Rightarrow K \geq 0.0258 \tag{106}$$

e della specifica riguardante i poli a ciclo chiuso, che fornisce un limite superiore. Dal comando $rlocus(Gs * Ps)$, dopo aver eseguito $Gs = zpk([-2.6 - 0.4], [-56 - 0.02], 1)$, si ottiene:

$$0.0258 \leq K \leq 0.0291 \Rightarrow K = 0.027 \tag{107}$$

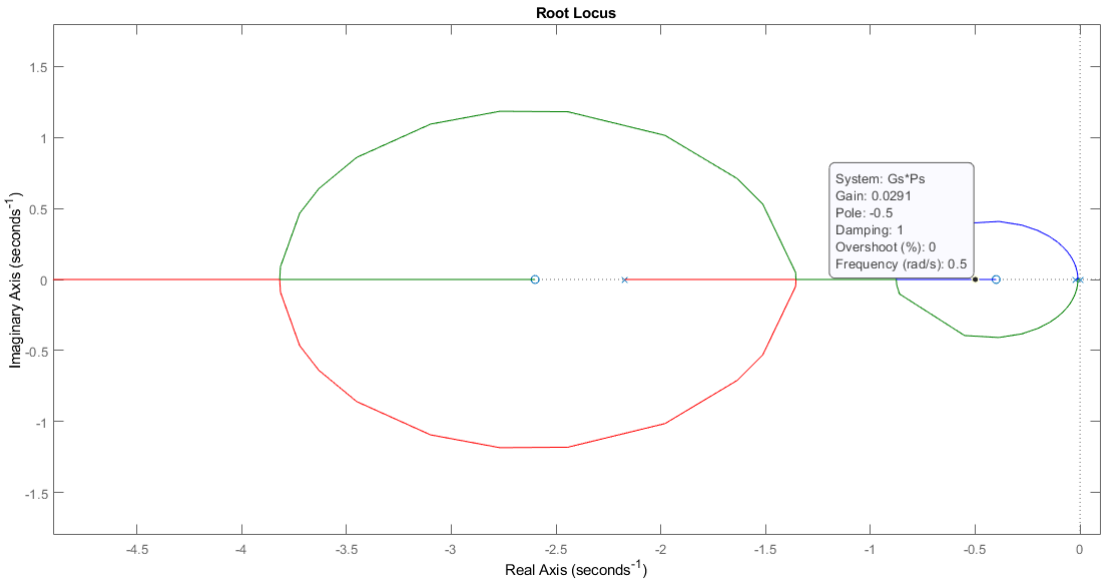


Figura 75: Luogo delle radici $G(s) \cdot P(s)$ (2° controllore).

Si ottengono le seguenti funzioni di trasferimento del controllore e in catena aperta:

$$G(s) = \frac{0.027 \cdot (s + 2.6)(s + 0.4)}{(s + 56)(s + 0.02)} \tag{108}$$

$$F(s) = \frac{98.388 \cdot (s + 2.6)(s + 0.4)}{s(s + 56)(s + 0.02)(s + 2.172)} \tag{109}$$

Verifichiamo a questo punto che le specifiche siano soddisfatte:

- il sistema è di tipo 1;
- $|\tilde{e}_1| = \dfrac{1}{K_F} = \dfrac{56 \cdot 0.02 \cdot 2.172}{3644 \cdot 0.02 \cdot 2.6 \cdot 0.4} = 0.023774$;
- la specifica sui poli a ciclo chiuso è verificata.

Il calcolo dei poli a ciclo chiuso rivela che la parte immaginaria della coppia di poli complessi coniugati risulta minore rispetto alla coppia di poli coniugati della $W(s)$ relativa al primo controllore: ciò comporta che i poli a ciclo chiuso appartengano ad una regione del piano complesso corrispondente ad uno smorzamento maggiore rispetto al caso precedente, con conseguente riduzione delle oscillazioni del transitorio.

```
>> Ws = feedback(Gs*Ps, 1);
>> pole(Ws)

ans =
          -54.2126 + 0.0000i
          -1.7327 + 0.8204i
          -1.7327 - 0.8204i
          -0.5136 + 0.0000i
```

Analizzando anche in questo caso la risposta al gradino del sistema a ciclo chiuso:

- $\hat{s} \simeq 12.8\%$
- $t_s \simeq 1.01\ s.$

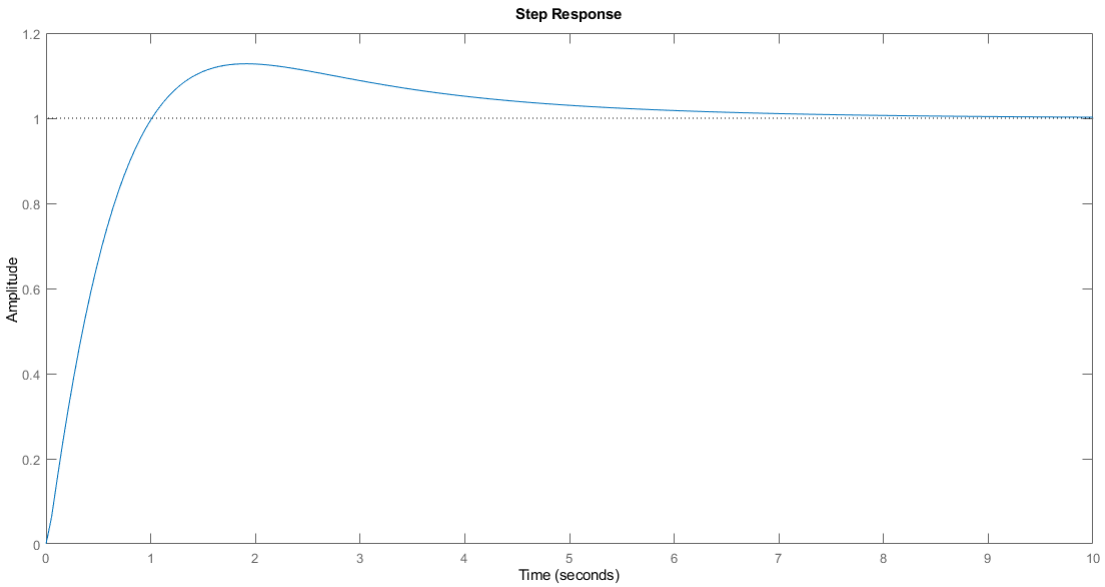


Figura 76: Risposta al gradino (2° controllore).

Si procede con la discretizzazione del controllore:

$$G(z) = \frac{0.027 \cdot (z - 0.9979)(z - 0.9889)}{(z - 1)(z - 0.7558)} \tag{110}$$

e all'inserimento nel sottosistema *Attitude*.

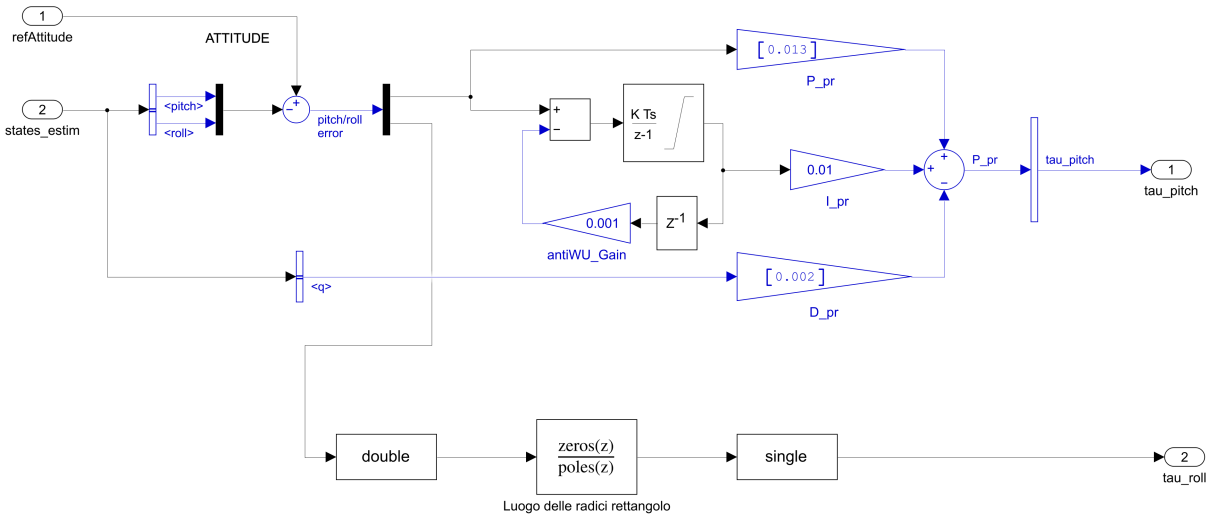


Figura 77: Controllo del *roll* tramite la $G(z)$ (blocco *Attitude*).

Di seguito vengono riportati i grafici relativi alla simulazione reale del volo con riferimenti 72 con questo controllore:

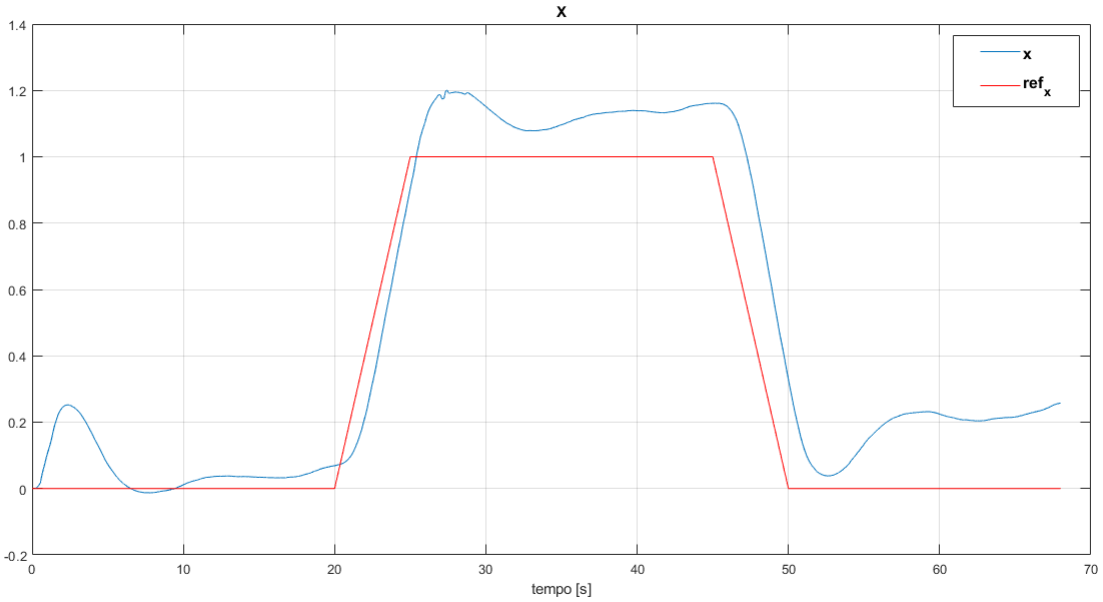


Figura 78: Grafico X (2° controllore).

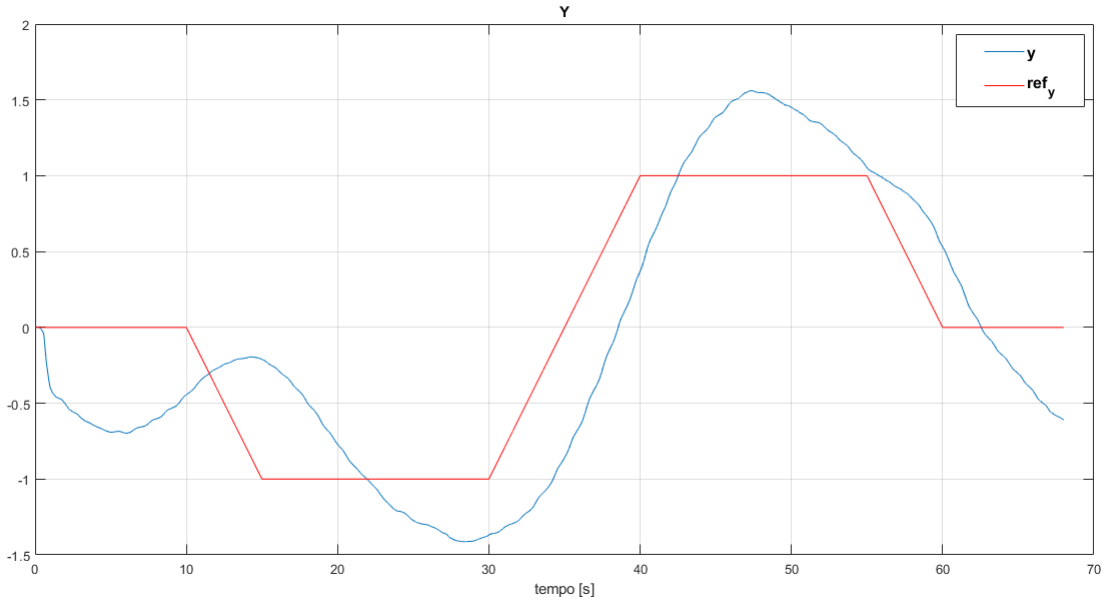


Figura 79: Grafico Y (2° controllore).

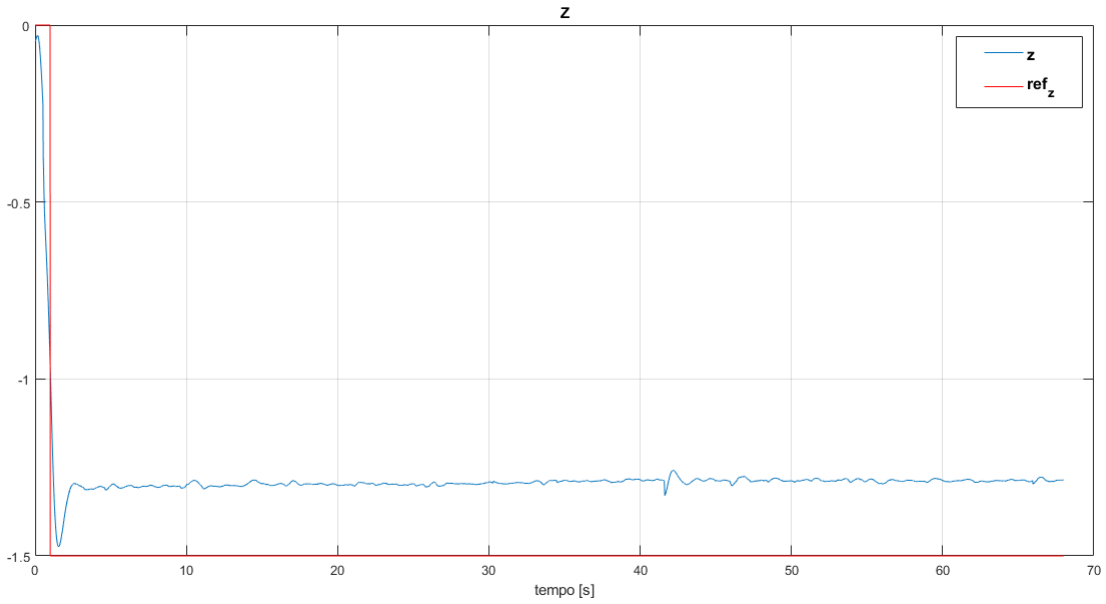


Figura 80: Grafico Z (2° controllore).

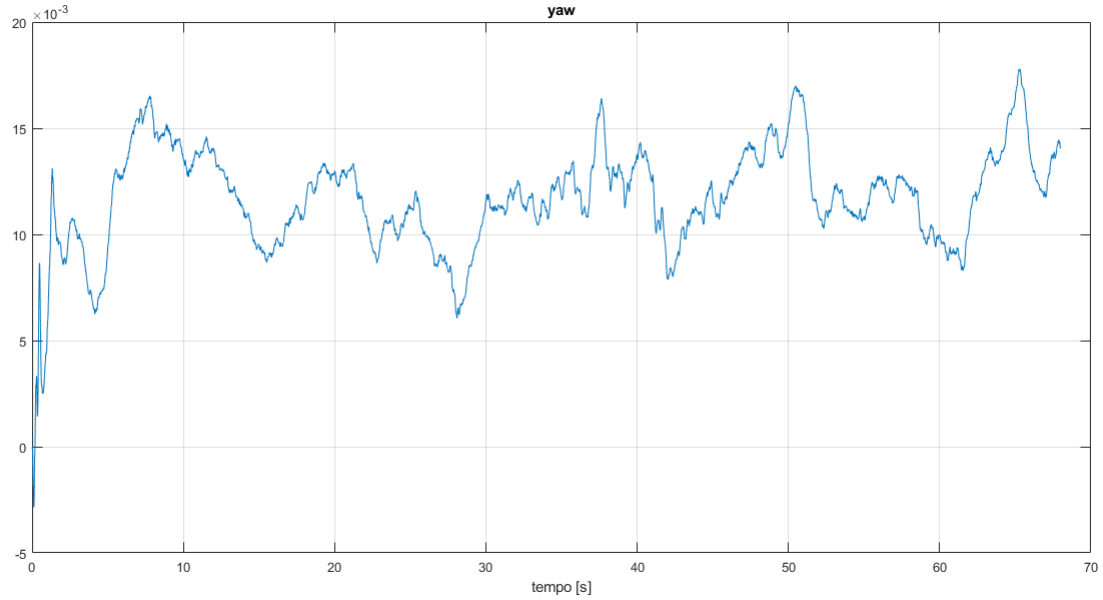


Figura 81: Grafico Yaw (2° controllore).

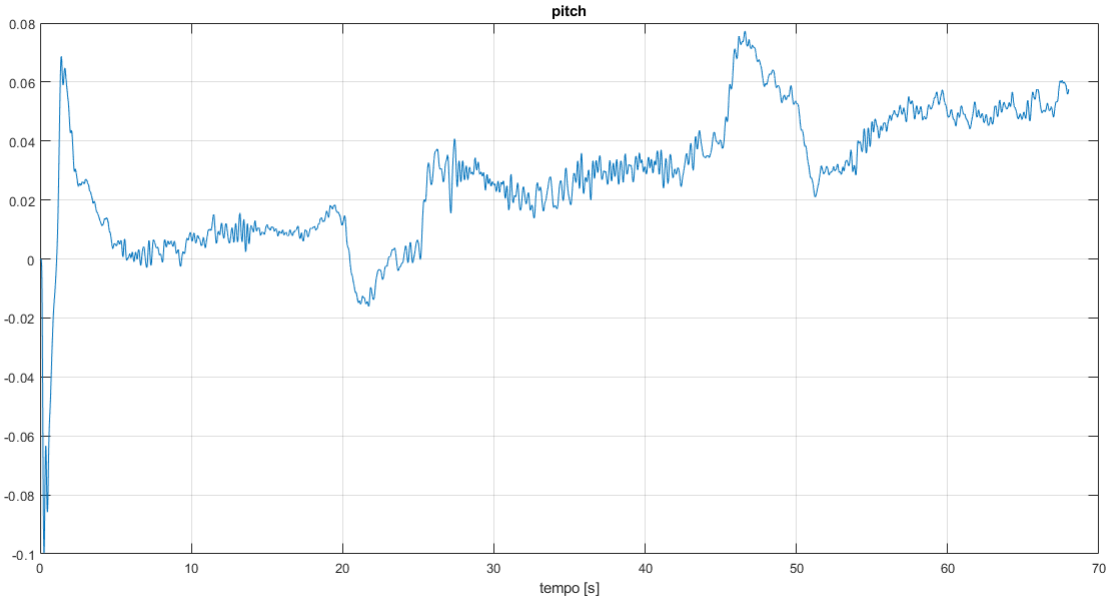


Figura 82: Grafico Pitch (2° controllore).

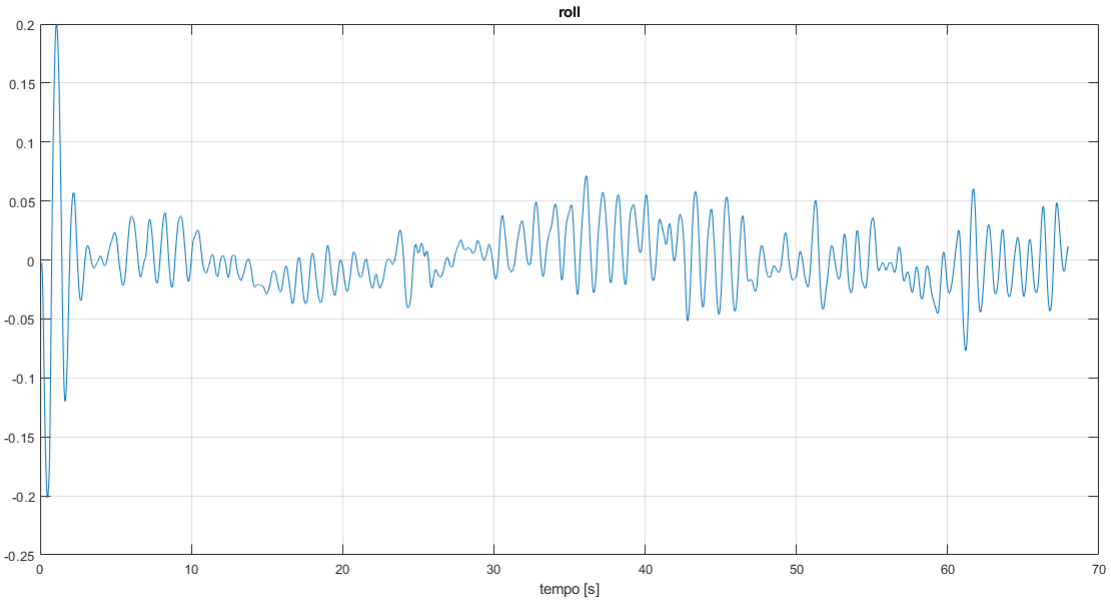


Figura 83: Grafico Roll (2° controllore).

Da quest'ultimo tracciato si conclude che il secondo controllore implementato con luogo delle radici presenta prestazioni soddisfacenti per questo tipo di volo, pertanto gli obiettivi prefissati sono stati raggiunti.

5.3 Sintesi in frequenza

Nella sintesi per tentativi nel dominio della frequenza si mettono in relazione proprietà della funzione di trasferimento in catena aperta $F(s) = G(s) \cdot P(s)$ con quelle della funzione di trasferimento in catena chiusa $W(s)$, e si sceglie il controllore $G(s)$ in modo da modificare opportunamente le caratteristiche in frequenza della funzione $W(j\omega)$. Infine si verifica che la risposta armonica $W(j\omega)$ soddisfi realmente le specifiche desiderate. Le caratteristiche desiderate della funzione $W(s)$, associate alle specifiche del sistema a ciclo chiuso, sono in genere di due tipi:

- specifiche univoche, che si traducono in una struttura ben precisa della funzione $F(s)$, e riguardano solitamente la risposta a regime permanente per ingressi polinomiali;
- specifiche non univoche, che possono essere soddisfatte con diverse scelte della funzione $F(s)$, e riguardano principalmente la risposta transitoria e la risposta a regime ad ingressi sinusoidali.

Implementazione controllore In questo caso, le specifiche imposte sono le seguenti, di cui le prime due rappresentano le specifiche univoche mentre le restanti due sono specifiche non univoche:

- sistema di tipo 1
- $|\tilde{e}_1| \leq 0.022$
- $M_r \leq 2dB$
- $B_3 = 0.5Hz$

Il controllore $G(s)$ presenta la forma $G(s) = \hat{G}(s) \cdot R(s)$, in cui $\hat{G}(s)$ è il controllore di primo tentativo che deve soddisfare le specifiche univoche, mentre $R(s)$, detta rete compensatrice, ha la funzione di modificare opportunamente la $F(s)$ in modo da soddisfare le specifiche non univoche. Dopo il progetto di $G(s)$, la verifica delle proprietà della funzione $F(j\omega)$ viene svolta tramite i diagrammi di Bode. Si devono poi verificare le specifiche di controllo della corrispondente funzione di trasferimento a ciclo chiuso $W(j\omega)$, usando il diagramma di Nichols.
Dalla specifica sull'errore segue che:

$$|\tilde{e}_1| = \frac{1}{K_F} \leq 0.022 \Rightarrow \begin{cases} K_F = K_G \cdot K_P \geq 45.45 \\ K_P = \dfrac{3644}{2.172} \end{cases} \Rightarrow K_G \geq 0.0271 \qquad (111)$$

Le specifiche non univoche date sulla funzione di trasferimento a ciclo chiuso $W(j\omega)$ vengono tradotte in specifiche riguardanti la risposta armonica del sistema in catena aperta $F(j\omega)$:

- visto che il modulo alla risonanza è legato al margine di fase, la specifica $M_r \leq 2$ dB diventa $m_\varphi \geq 47°$;
- per quanto riguarda la banda passante, dalla relazione empirica $\omega_t[rad/s] = [3 \div 5] \cdot B_3[Hz]$, a partire da $B_3 = 0.5$ Hz si può scegliere come specifica a ciclo aperto la pulsazione di attraversamento $\omega_t \simeq 2$ rad/s.

Queste due specifiche servono per limitare la sovraelongazione e il tempo di salita, a cui sono rispettivamente legati modulo alla risonanza e banda passante. Infatti una sovraelongazione moderata corrisponde ad un limitato modulo alla risonanza, mentre il tempo di salita diminuisce all'aumentare della banda passante.
Il controllore di primo tentativo, per soddisfare la specifica sull'errore, è scelto come segue:

$$\hat{G}(s) = 0.028 \qquad (112)$$

La risultante funzione di trasferimento in catena aperta è:

$$\hat{F}(s) = \frac{102.032}{s(s + 2.172)} \tag{113}$$

Questa funzione soddisfa le specifiche univoche, in quanto il sistema è di tipo 1 e l'errore vale $|\tilde{e}_1| = 0.021287$, ma dai diagrammi di Bode in 84 e in 85 si osserva che le specifiche non univoche su margine di fase e pulsazione di attraversamento non sono soddisfatte.

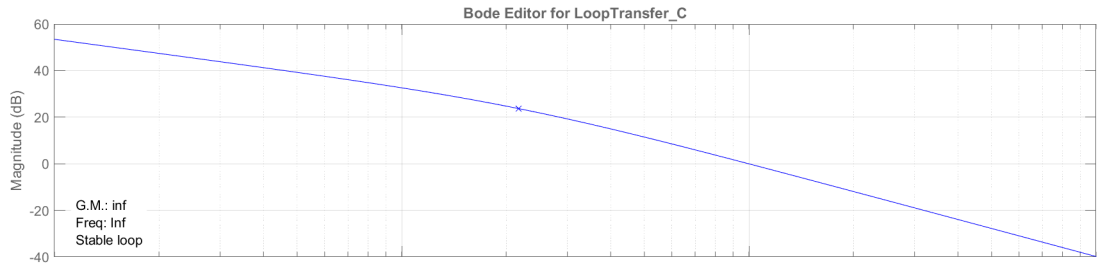


Figura 84: Diagramma di Bode del modulo di $\hat{F}(s)$.

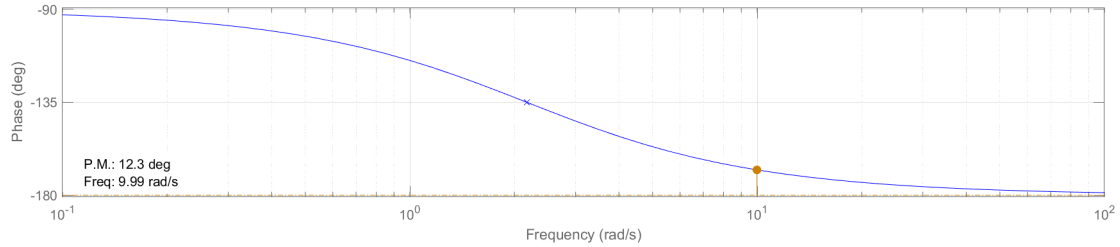


Figura 85: Diagramma di Bode della fase di $\hat{F}(s)$.

E' necessario quindi progettare una funzione anticipatrice, per aumentare la fase in corrispondenza della pulsazione di attraversamento desiderata. Infatti è opportuno aumentare la fase di una quantità superiore a quella strettamente necessaria, perché si deve tenere conto che la successiva introduzione della funzione attenuatrice comporterà una diminuzione della fase. Dai diagrammi universali si scelgono i parametri $m_a = 16$ e $\tau_a \cdot \omega_t = 0.8$, pertanto $\tau_a = 0.4$:

$$R_a(s) = \frac{1 + \tau_a s}{1 + \dfrac{\tau_a}{m_a} s} = \frac{1 + 0.4s}{1 + 0.025s} = \frac{16 \cdot (s + 2.5)}{s + 40} \tag{114}$$

In questo modo la fase di $F(j\omega)$ aumenta di circa 35° in corrispondenza di $\omega = 2$ rad/s. La nuova funzione di trasferimento in catena aperta è:

$$\bar{F}(s) = \hat{F}(s) \cdot R_a(s) = \frac{1632.512 \cdot (s + 2.5)}{s(s + 2.172)(s + 40)} \tag{115}$$

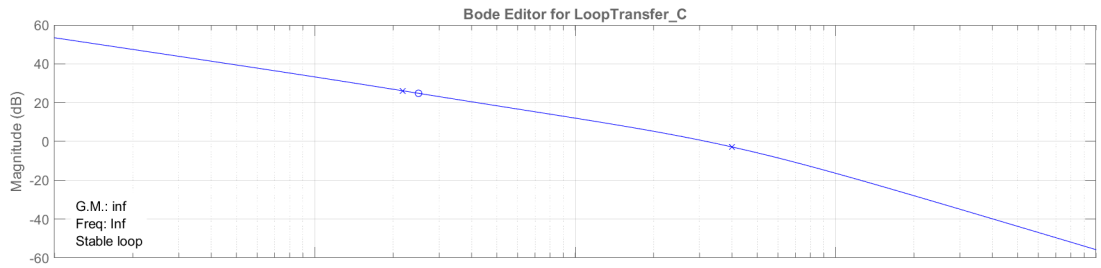


Figura 86: Diagramma di Bode del modulo di $\bar{F}(s)$.

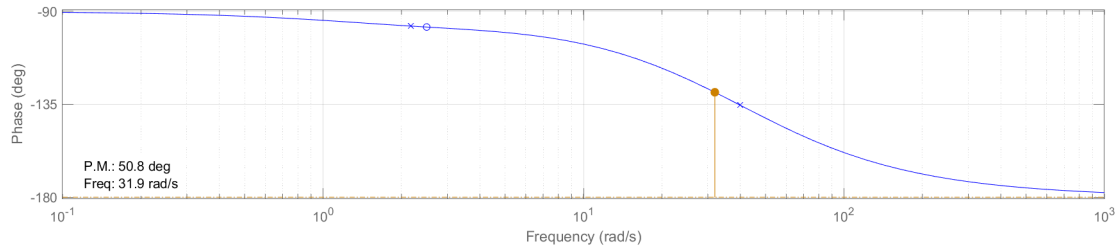


Figura 87: Diagramma di Bode della fase di $\bar{F}(s)$.

Affinché la pulsazione di attraversamento sia quella desiderata, è necessario progettare una funzione attenuatrice, che diminuisca il modulo di circa 30 dB in $\omega = 2$ rad/s, senza però che la fase scenda al di sotto di -133° in corrispondenza di tale pulsazione. Dai diagrammi universali si scelgono $m_i = 22$ e $\tau_i \cdot \omega_t = 200$, pertanto $\tau_i = 100$:

$$R_i(s) = \frac{1 + \frac{\tau_i}{m_i}s}{1 + \tau_i s} = \frac{1 + \frac{100}{22}s}{1 + 100s} = \frac{s + 0.22}{22(s + 0.01)} \tag{116}$$

Le funzioni di trasferimento complessive del controllore e in catena aperta risultano essere:

$$G(s) = \hat{G}(s) \cdot R_a(s) \cdot R_i(s) = \frac{0.020364 \cdot (s + 2.5)(s + 0.22)}{(s + 40)(s + 0.01)} \tag{117}$$

$$F(s) = G(s) \cdot P(s) = \frac{74.2 \cdot (s + 2.5)(s + 0.22)}{s(s + 2.172)(s + 40)(s + 0.01)} \tag{118}$$

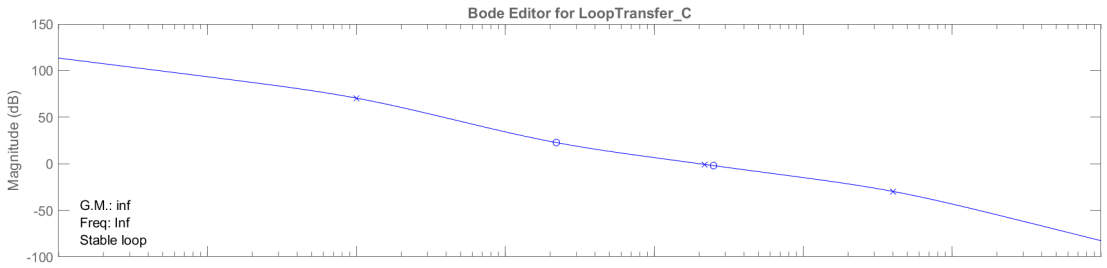


Figura 88: Diagramma di Bode del modulo di $F(s)$.

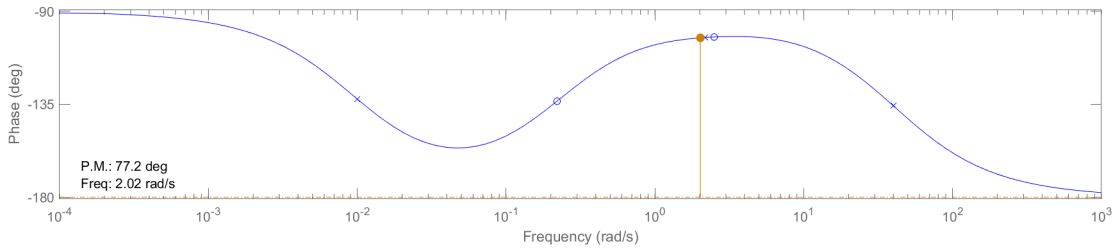


Figura 89: Diagramma di Bode della fase di $F(s)$.

Dai grafici 88 e 89 risulta $m_\varphi = 77.2°$ e $\omega_t = 2.02$ rad/s.
Il procedimento di sintesi non si esaurisce con la verifica delle specifiche a ciclo aperto: bisogna accertarsi che siano soddisfatte le specifiche a ciclo chiuso, che in questo caso sono $M_r \leq 2$ dB e $B_3 = 0.5$ Hz. Ciò si ottiene analizzando il diagramma di Nichols di $F(j\omega)$, che rappresenta il modulo di $F(j\omega)[dB]$ in funzione della fase di $F(j\omega)[°]$, tracciato da Matlab con il comando $nichols(Gs * Ps)$, dove Gs deve essere preventivamente caricata sul Workspace con i comandi seguenti:

```
Gcaps = 0.028;
Ra = tf([0.4 1],[0.025 1]);
Ri = tf([100/22 1], [100 1]);
Gs = Gcaps*Ra*Ri;
```

```
Gs = zpk(Gs)

Gs =

  0.020364 (s+2.5) (s+0.22)
  -------------------------
    (s+40) (s+0.01)

Continuous-time zero/pole/gain model.
```

Visto che il sistema è di tipo 1, e quindi $W(j0) = 1 = 0$ dB, la banda passante si ottiene come la pulsazione corrispondente all'intersezione tra il luogo a modulo costante a ciclo chiuso pari a -3 dB con il diagramma di Nichols di $F(j\omega)$. Dal diagramma di Nichols in 90 si osserva che la specifica sul modulo alla risonanza è verificata e che la banda passante vale $B_3 = 2.46$ rad/s $\simeq 0.4$ Hz.

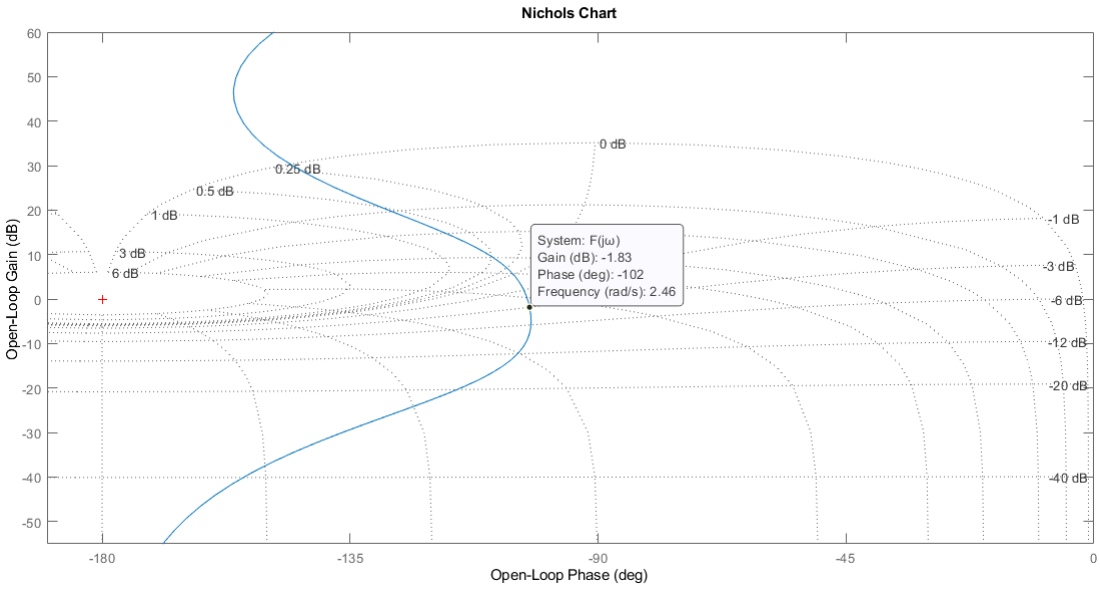


Figura 90: Diagramma di Nichols $F(j\omega)$.

A questo punto si può analizzare la risposta al gradino del sistema a ciclo chiuso per determinare quanto valgono sovraelongazione e tempo di salita:

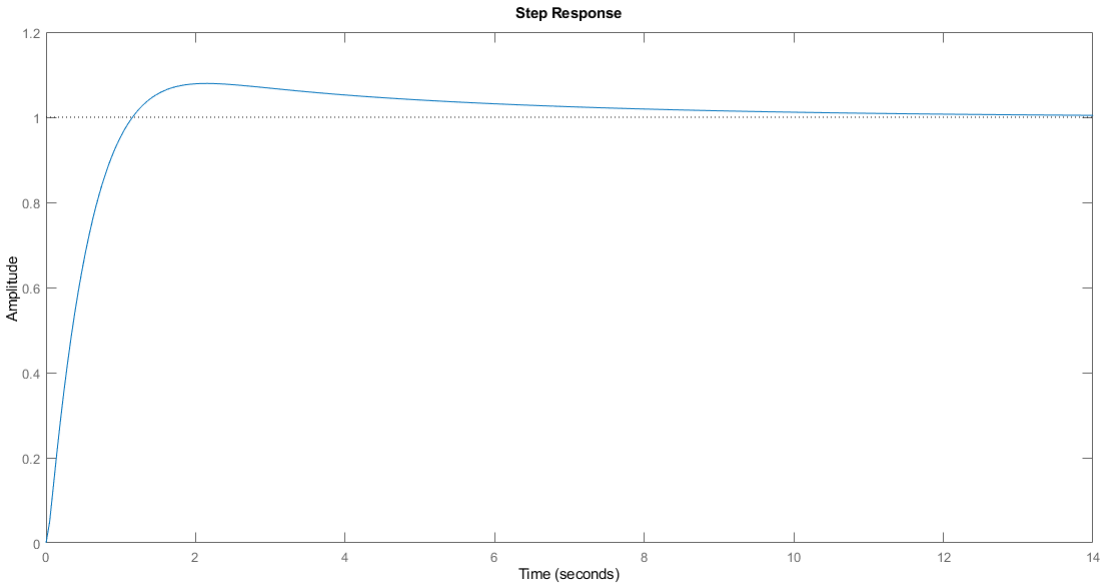


Figura 91: Risposta al gradino (sintesi in frequenza).

Si nota che:

- $\hat{s} = 0.8\%$
- $t_s = 1.15\ s$.

Visto che le specifiche imposte risultano tutte soddisfatte, si procede con la simulazione tramite Matlab, per accertarsi che questa vada a buon fine. A questo punto si discretizza il controllore complessivo in (117), ancora una volta con il comando $Gz = c2d(Gs, Ts,' zoh')$, e si ottiene:

$$G(z) = \frac{0.020364 \cdot (z - 0.9989)(z - 0.9888)}{(z - 1)(z - 0.8187)} \tag{119}$$

Come fatto in precedenza, si inserisce la funzione di trasferimento appena ottenuta nel blocco *Attitude* per il controllo del *rollio*.

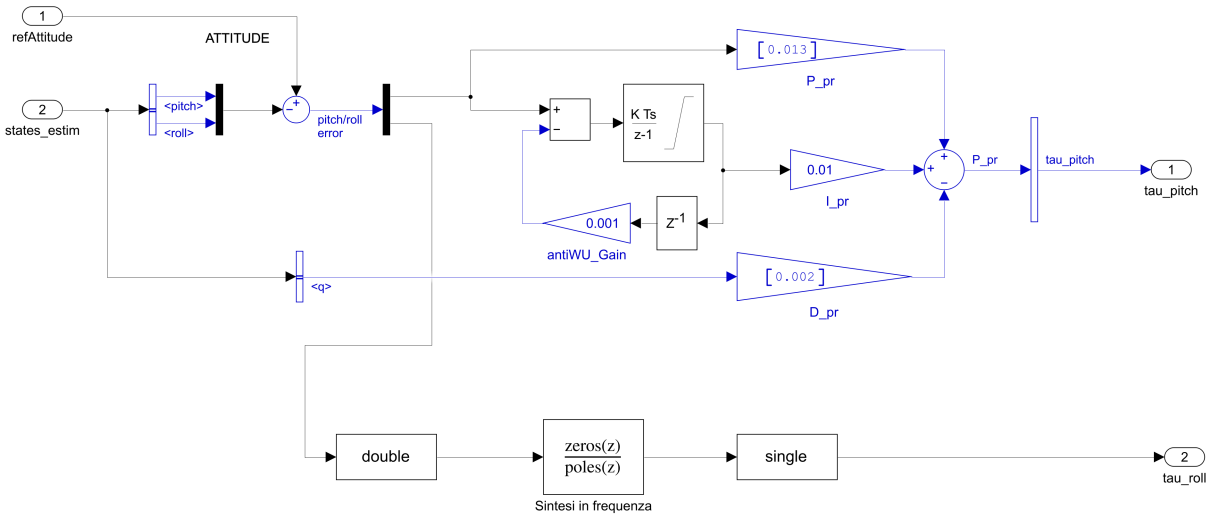


Figura 92: Controllo del *roll* tramite la $G(z)$ (blocco *Attitude*).

Le prestazioni del controllore implementato tramite sintesi in frequenza, nella simulazione su drone del volo stazionario, sono riportate di seguito:

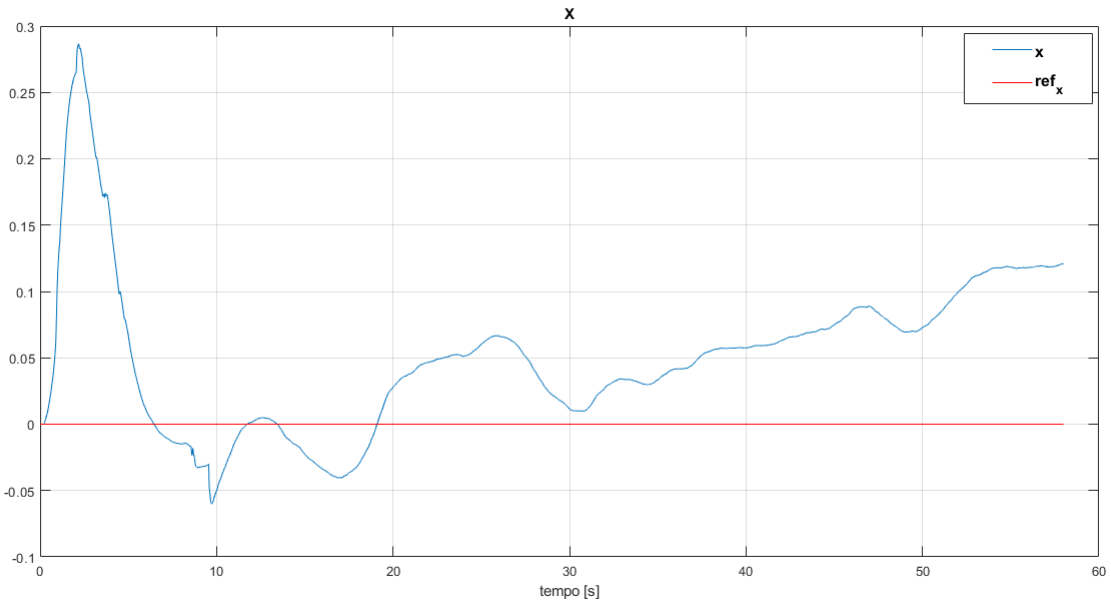


Figura 93: Grafico X (sintesi in frequenza).

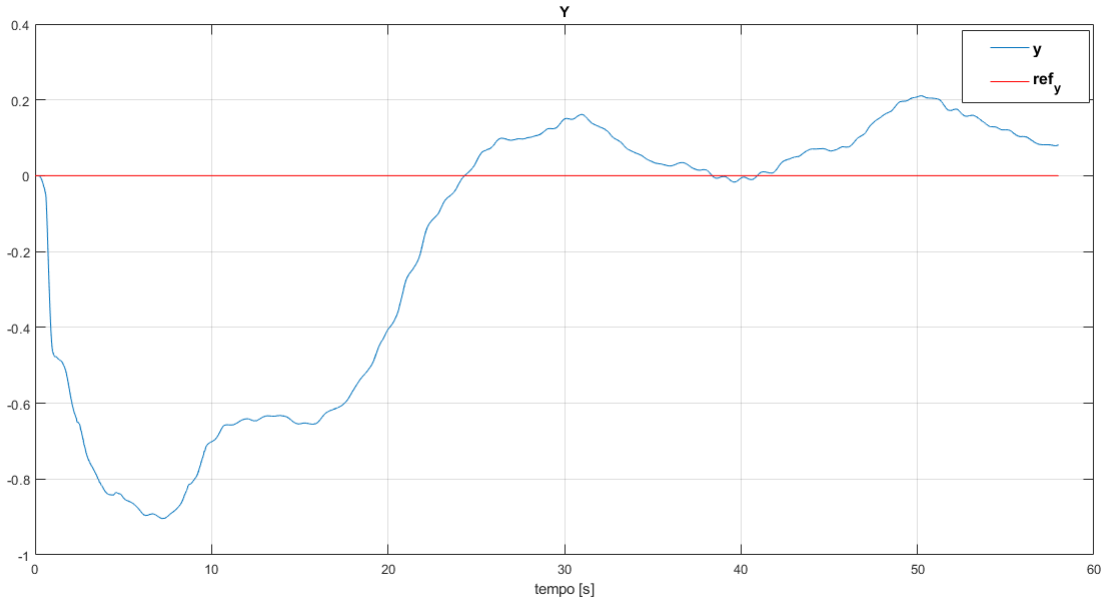


Figura 94: Grafico Y (sintesi in frequenza).

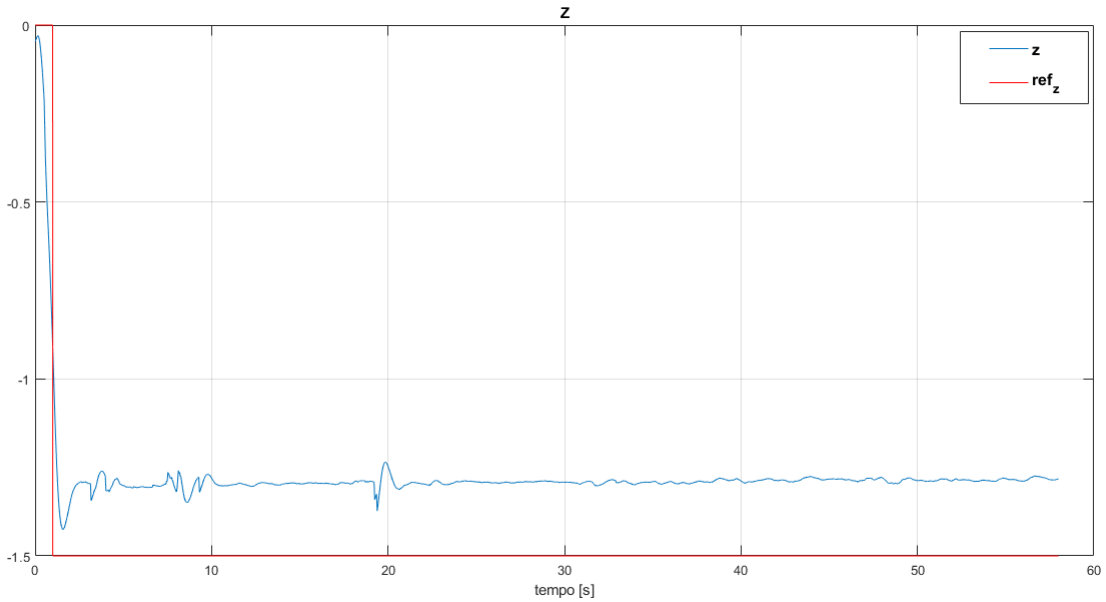


Figura 95: Grafico Z (sintesi in frequenza).

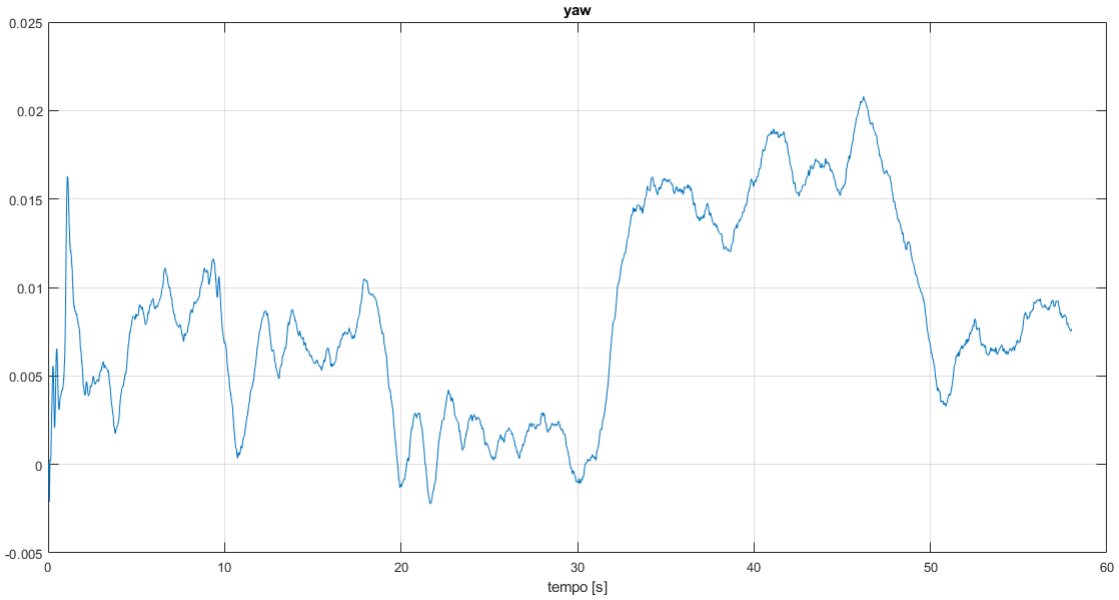


Figura 96: Grafico Yaw (sintesi in frequenza).

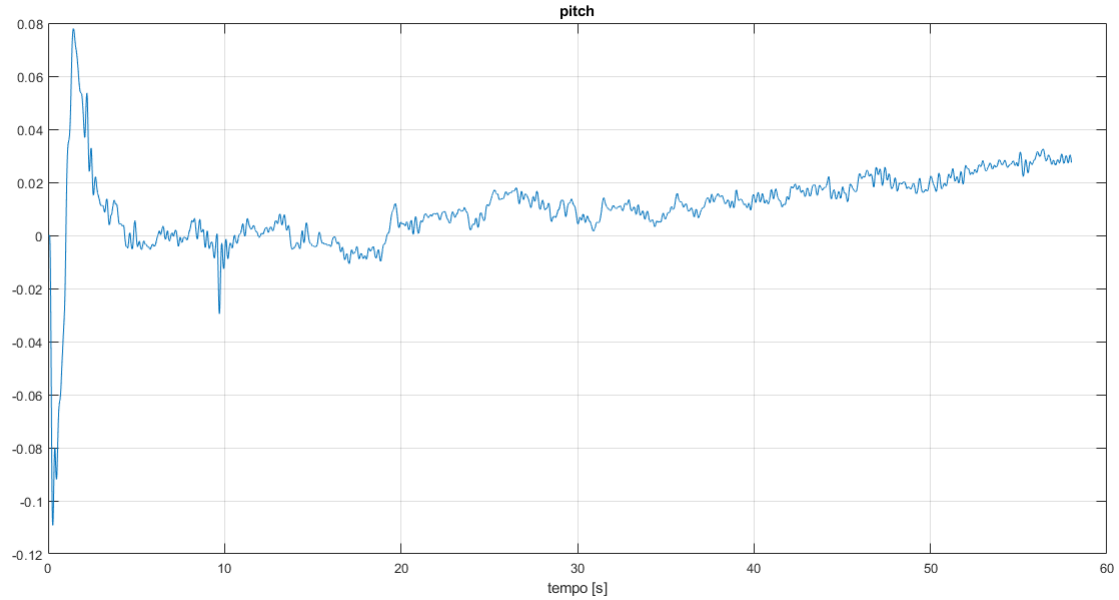


Figura 97: Grafico Pitch (sintesi in frequenza).

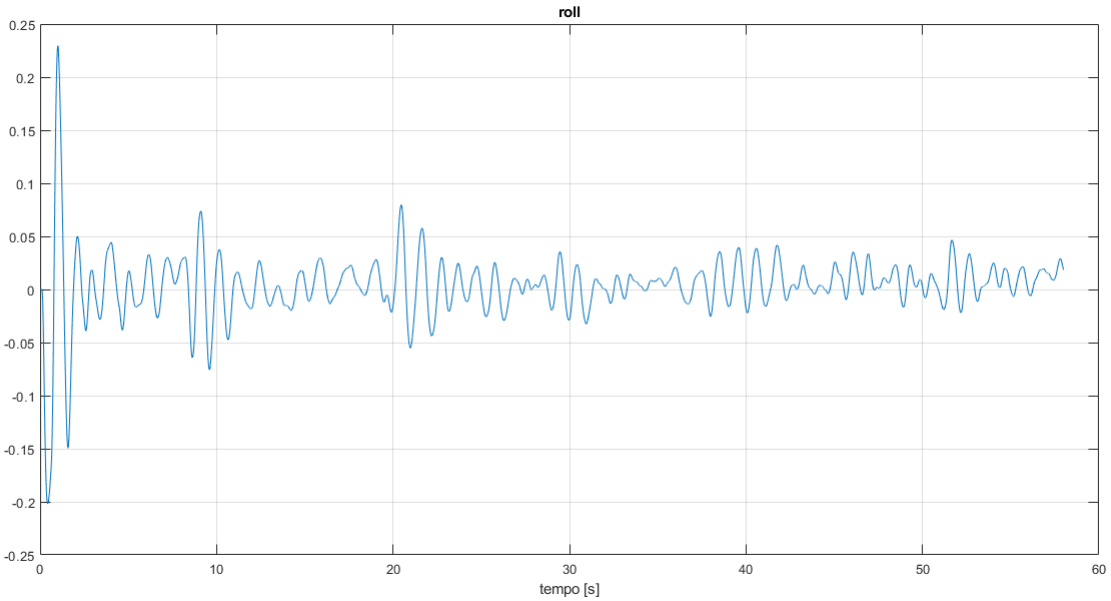


Figura 98: Grafico Roll (sintesi in frequenza).

Lo stesso controllore è stato poi utilizzato in una simulazione di volo con riferimenti non nulli sul piano orizzontale, analogamente a quanto fatto in precedenza per il luogo delle radici. I riferimenti forniti per le coordinate x e y sono leggermente differenti rispetto al caso precedente (figura 72), e sono stati scelti in modo tale da far seguire al drone una traiettoria a "quadrato".

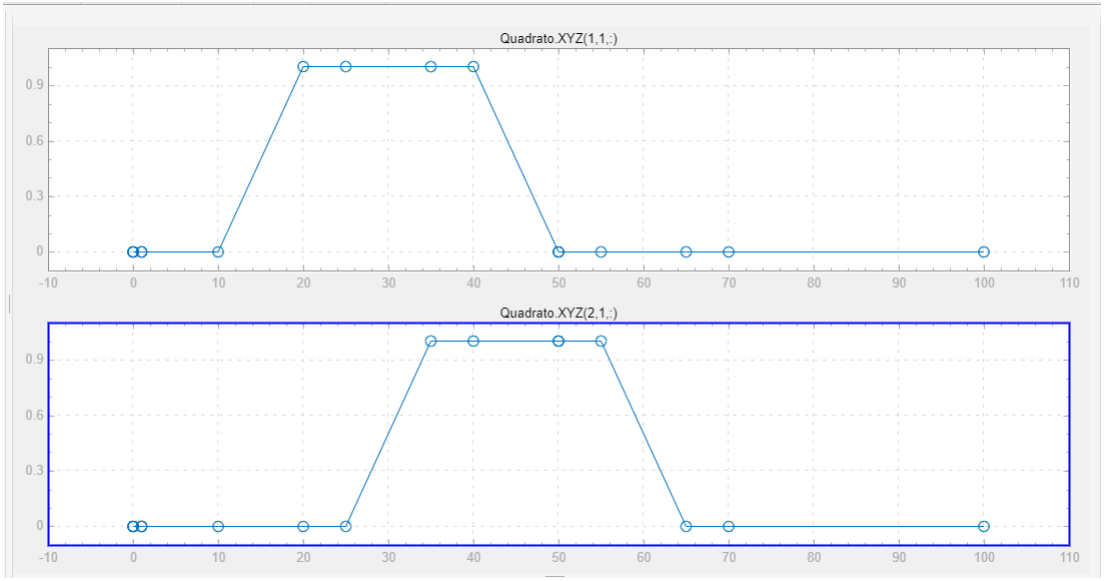
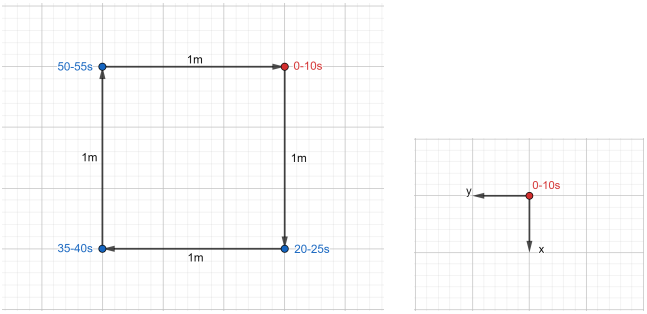


Figura 99: Nuovi riferimenti per gli assi x (in alto) e y (in basso).

La figura 100a consente di visualizzare tale traiettoria: il punto di decollo e di atterraggio del drone viene indicato dal punto rosso, mentre in blu sono indicati i vertici del quadrato in cui il drone staziona per 5 secondi. Si considerano inoltre 10 secondi per percorrere una distanza di 1 metro. La simulazione ha una durata di 70 secondi e in corrispondenza di ogni vertice sono indicati gli intervalli di tempo in cui il drone staziona in quel punto.



(a) (b)

Figura 100: Visualizzazione della traiettoria a "quadrato".

Le prestazioni del controllore implementato tramite sintesi in frequenza, nella simulazione su drone del volo con riferimenti non nulli sul piano orizzontale, sono riportate di seguito:

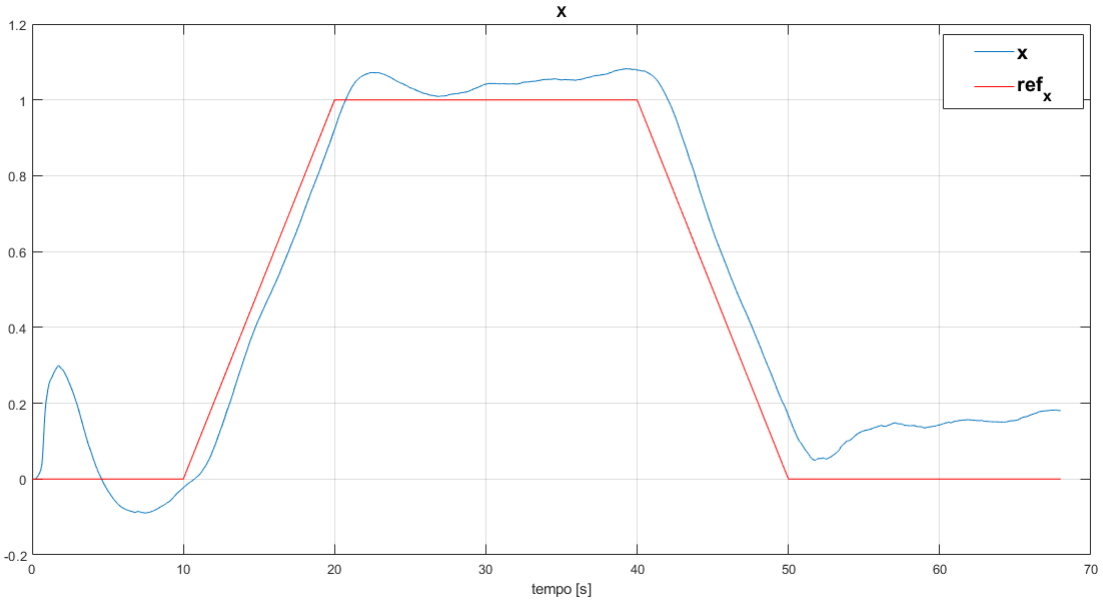


Figura 101: Grafico X (sintesi in frequenza, volo sul piano orizzontale).

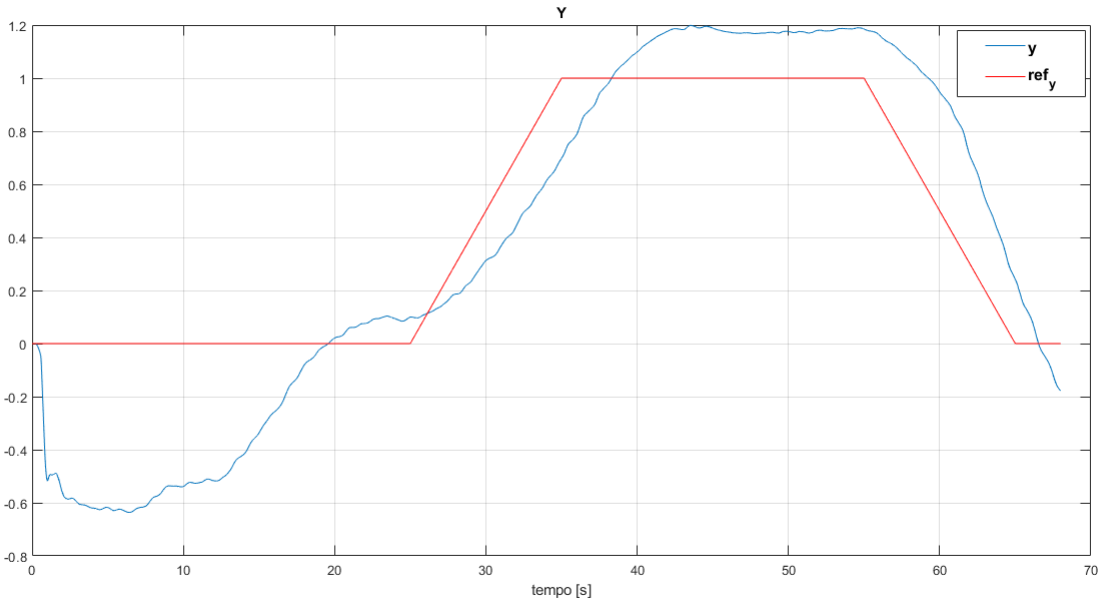


Figura 102: Grafico Y (sintesi in frequenza, volo sul piano orizzontale).

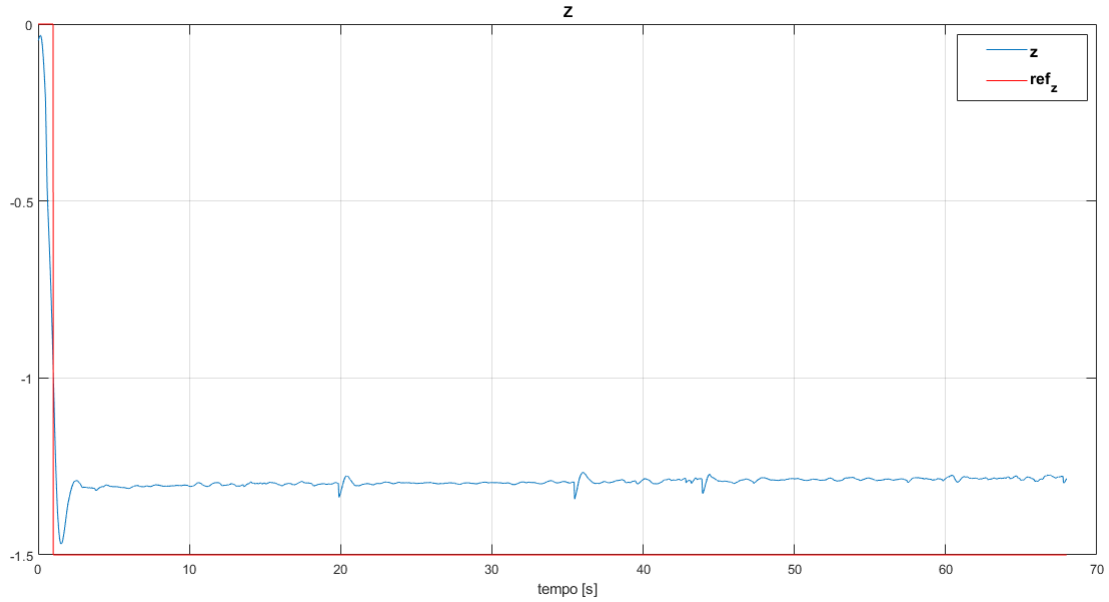


Figura 103: Grafico Z (sintesi in frequenza, volo sul piano orizzontale).

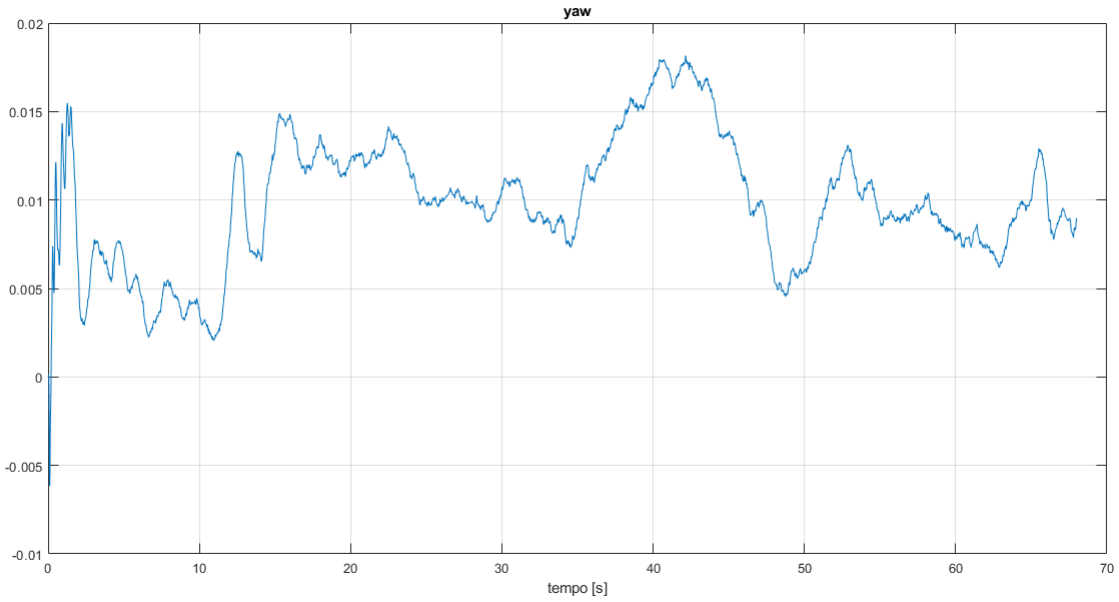


Figura 104: Grafico Yaw (sintesi in frequenza, volo sul piano orizzontale).

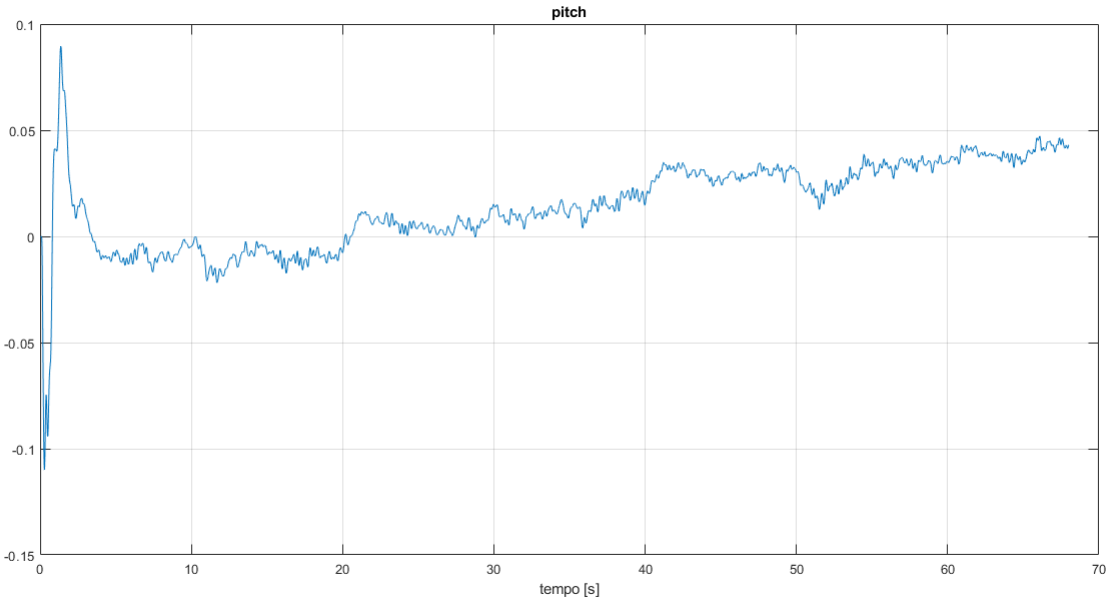


Figura 105: Grafico Pitch (sintesi in frequenza, volo sul piano orizzontale).

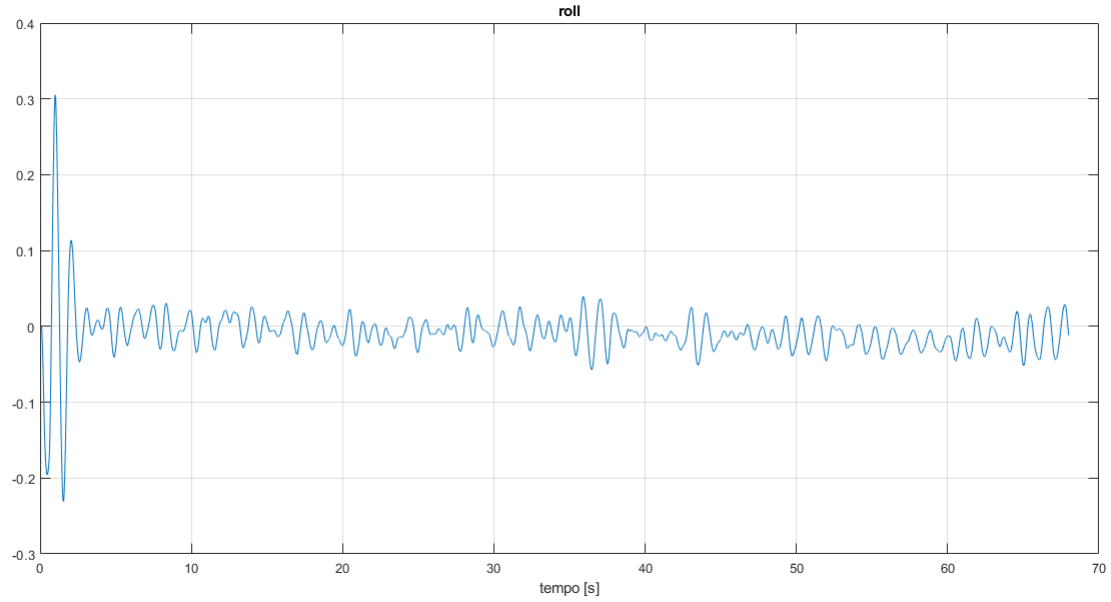


Figura 106: Grafico Roll (sintesi in frequenza, volo sul piano orizzontale).

A differenza di (102), il controllore (119) presenta un comportamento soddisfacente sia per il volo stazionario sia per il movimento lungo x e y. Per quanto riguarda quest'ultimo tipo di volo, il drone presenta un transitorio peggiore rispetto al volo con il secondo controllore implementato con sintesi tramite luogo delle radici (110), ma ha un comportamento a regime migliore.

6 Conclusioni

In questo elaborato è stata presentata l'implementazione di tre funzioni di trasferimento per il controllo del *rollio* del drone Parrot Mambo e l'analisi delle loro prestazioni rispetto al volo stazionario e al volo con movimenti sul piano orizzontale. Sebbene il primo controllore progettato sia stato ritenuto non soddisfacente per il secondo tipo di volo, relativo alla traiettoria a "rettangolo", un ulteriore sviluppo potrebbe essere la verifica del comportamento del suddetto controllore, e dei restanti due, in una simulazione con un diverso tipo di riferimenti per x e y, per esempio si potrebbe considerare un maggiore intervallo di tempo per percorrere la stessa distanza, cioè si potrebbe fornire come riferimento una rampa di minore pendenza.
Per le analisi delle prestazioni si è preferito considerare i dati salvati nei *MAT file*, scaricati al termine delle relative simulazioni sul drone, in quanto, dopo numerose prove pratiche, è emerso che le simulazioni di Matlab non consentono di determinare se un particolare controllore sia valido o meno, in quanto il comportamento reale risulta notevolmente differente. Ciò è dovuto in parte alla necessità di discretizzare, con inevitabile approssimazione, la funzione di trasferimento del controllore, implementata a tempo continuo, e in parte alle condizioni ambientali in cui si esegue il volo. Le simulazioni a cui si è fatto riferimento sopra sono state effettuate in un ambiente indoor di dimensioni $10 \times 6 \times 3$ metri circa. Nell'appendice B sono riportate alcune considerazioni sui problemi riscontrati durante le prove pratiche di volo.

A Legenda

Simbolo	Significato
r	Posizione del drone rispetto all'*Earth Frame*
$\dot{r}$	Velocità lineare rispetto all'*Earth Frame*
v	Velocità lineare rispetto al *Body Frame*
R_E^B	Matrice di rotazione $E \Rightarrow B$
R_B^E	Matrice di rotazione $B \Rightarrow E$
CdM	Centro di Massa
ω	Velocità angolare rispetto al *Body Frame*
$\dot{\omega}$	Accelerazione angolare rispetto al *Body Frame*
η	Angoli di Eulero del quadricottero (rispetto all'*Earth Frame*)
$\dot{\eta}$	Derivata prima rispetto al tempo degli angoli di Eulero
S_E^B	Matrice di rotazione $E \Rightarrow B$ per la velocità angolare
S_B^E	Matrice di rotazione $B \Rightarrow E$ per la velocità angolare
J	Matrice di inerzia
M_G	Effetto giroscopico
J_r	Inerzia dei rotori
Ω_r	Velocità relativa dei rotori
M_B	Momento torcente applicato al drone dai rotori
F_i	Forza aerodinamica di spinta generata dai rotori
M_i	Momento aerodinamico generato dai rotori
ρ	Densità dell'aria
A	Area della sezione dell'elica
C_T, C_D	Coefficienti aerodinamici
d	Diametro dell'elica
R	Raggio dell'elica
Ω_i	Velocità angolare dell'i-esimo rotore
K_F	Coefficiente di portanza
K_M	Coefficiente di resistenza aerodinamica
l	Distanza fra il centro di ciascun rotore e l'origine del sistema B
l_{xy}	Distanza sugli assi x e y fra il centro di ciascun rotore e il CdM
h	Distanza sull'asse z fra il centro di ciascun rotore e il CdM
m	Massa del drone
g	Accelerazione di gravità
F_B	Forza generata dai rotori
K_{dx}, K_{dy}, K_{dz}	Coefficienti di attrito traslazionale
X	Vettore degli stati
U	Vettore degli ingressi

Simbolo	Significato
F_D	Forza di resistenza dell'aria
Q_i	Momento torcente dovuto alla resistenza dell'aria
D_i	Posizionamento del rotore i-esimo rispetto al CdM
a	Pendenza della curva di portanza del profilo alare
α	Angolo di attacco del rotore
θ_{tip}	Angolo geometrico dell'elica sulla punta del rotore
v_i	Velocità indotta attraverso il rotore
σ	Rigidità del rotore
$a_{1_s i}$	Angolo di inclinazione longitudinale dell'i-esimo rotore, rispetto a B
$b_{1_s i}$	Angolo di inclinazione laterale dell'i-esimo rotore, rispetto a B
$u_{1_s i}$	Angolo di inclinazione longitudinale dell'i-esimo rotore, rispetto a B_i
$v_{1_s i}$	Angolo di inclinazione laterale dell'i-esimo rotore, rispetto a B_i
v_{ri}	Velocità lineare dell'i-esimo rotore rispetto a B
μ_{ri}	Advance ratio
ψ_{ri}	Direzione azimutale del rotore
λ_{hi}	Inflow ratio
γ	Lock number
$J^B_{B_i}$	Matrice di trasformazione $B_i \Rightarrow B$

B Problemi di volo

Le due problematiche principali riscontrate durante le prove di volo sul drone fanno riferimento alla durata della batteria al litio e all'illuminazione dell'ambiente. Per quanto riguarda la batteria inserita nel drone, essa deve essere carica al 100% e consente di effettuare circa 3 o 4 voli di durata di 60/70 secondi prima di presentare comportamenti anomali. Se la batteria scende al di sotto della soglia di carica del 70-75% circa, il drone presenta oscillazioni molto marcate ed è quindi necessario disporre di diverse batterie cariche e pronte per l'uso durante le prove pratiche di volo. E' inoltre consigliabile disporre di un caricatore esterno, così da evitare di dover attendere il caricamento della batteria all'interno del drone e di surriscaldare eccessivamente il dispositivo.
Il secondo fattore da tenere in considerazione è l'illuminazione del luogo in cui si effettuano le prove di volo: l'ambiente deve essere ben illuminato ma bisogna evitare la luce solare diretta. Si è infatti notato durante le simulazioni di volo di questo progetto che il drone presenta comportamenti anomali, come la terminazione forzata della simulazione, se esposto a luce solare diretta, per esempio attraverso una vetrata.
Per ulteriori informazioni riguardanti l'utilizzo del progetto qui analizzato e le modalità di connessione del drone al PC si consiglia di fare riferimento al file *User guide* allegato al progetto e alla documentazione Matlab.

Riferimenti bibliografici

[1] Alberto Isidori, *Sistemi di controllo.*

[2] M.L. Corradini, G. Orlando, *Fondamenti di Automatica.*

[3] Luca Mezzanotti, *Studio e sviluppo di controllori per l'assetto di mini droni.*

[4] Daniele Gambini, *Studio e sviluppo in Simulink di tecniche di controllo per minidroni.*

[5] Paolo Ceppi, *Model-based Design of a Line-tracking Algorithm for a Low-cost Mini Drone through Vision-based Control.*

[6] N. Nowshin, H.Md.A. Kabir, A.S. Jannat, K.K. Fatema, *Designing and Implementation of a Multi-purpose Quadcopter.*

[7] V.S.D. Yeedi, C.C. Veedhi, *Estimation of altitude using ultrasonic and pressure sensors.*

[8] István Lovas, András Molnár, *Quadcopter power consumption analyzation at different landing trajectories.*

[9] N. Miladi, T. Ladhari, S.H. Said, F. M'Sahli, *Tracking control of quadcopter using explicit nonlinear model predictive control.*

[10] Ganga G, Meher Madhu Dharmana, *MPC Controller for Trajectory Tracking Control of Quadcopter.*

[11] Jun Li, Yuntang Li, *Dynamic Analysis and PID Control for a Quadrotor.*

[12] E.H. Khadija, E.K. Abdeljalil, M. Mostafa, A. Hassan, *Adapting Parameters for Flight Control of a Quadcopter using Reference Model and Fuzzy Logic.*

[13] Samir Bouabdallah, *Design and Control of Quadrotors with application to autonomous flying.*

[14] Alakananda BG, Venugopal N, *Development of a Programmable System on Chip (Psoc) based Quadcopter.*

[15] Hongping Liu, Guanbin Gao, *Dynamic Modeling and Analyzing for a Novel X-Quadrotor.*

[16] K.M. Thu, G.A. Igorevich, *Modeling and Design Optimization for Quadcopter Control System Using L1 Adaptive Control.*

[17] P. Pounds, R. Mahony, P. Corke, *Modelling and Control of a Large Quadrotor Robot.*

[18] D.S. Vamsi, T.V.S.P. Tanoj, U.M. Krishna, M. Nithya, *Performance Analysis of PID controller for Path Planning of a Quadcopter.*

[19] *Quadcopter Dynamics, Simulation, and Control.*

[20] Davide Meneghel, *Regolatori PID: tecniche di sintesi e problematiche implementative.*

[21] `https://support.parrot.com/it/support/prodotti/parrot-mambo`

[22] `https://www.mathworks.com/help/aeroblks`

[23] `http://www.pieter-jan.com/node/11`