
# Studio e sviluppo di tecniche di controllo per droni utilizzando Simulink e Matlab

## Study and development of control techniques for drones using Simulink and Matlab

**Tesi di laurea triennale**

**Candidato:**
**Francesco Silvi**

**Relatore:**
**Gianluca Ippoliti**

**Correlatore:**
**Giuseppe Orlando**

# Indice

# 0. Introduzione

Un quadricottero, come elicotteri ed altri aeromobili dotati di pale, appartiene alla famiglia degli aerogiri. È infatti dotato di quattro eliche in grado di generare una spinta capace di mantenerlo in volo e consentirne gli spostamenti. I quadricotteri, quasi nella totalità dei casi, sono dei mezzi nei quali non è presente un individuo a bordo. Le loro applicazioni coprono diversi campi di azione, dalla semplice perlustrazione, a mansioni più complesse come il trasporto o la lavorazione in ambienti difficilmente accessibili.

In questo progetto, tramite l'utilizzo di Simulink e Matlab, è stato studiato il modello di un minidrone della Parrot, per poi progettare un controllore che fosse in grado di regolarne il rollio e consentire gli spostamenti desiderati.

# 1. Studio a livello fisico di un quadricottero

## 1.1. Movimentazione

Un quadricottero deve i suoi movimenti alla portanza generata dai quattro rotori, i quali a seconda delle loro velocità di rotazione imprimono determinate forze e momenti sul drone.

I possibili movimenti in un quadricottero si sintetizzano in sei gradi di libertà (Figura 1):

- traslazione longitudinale
- traslazione laterale
- traslazione verticale
- rollio
- beccheggio
- imbardata

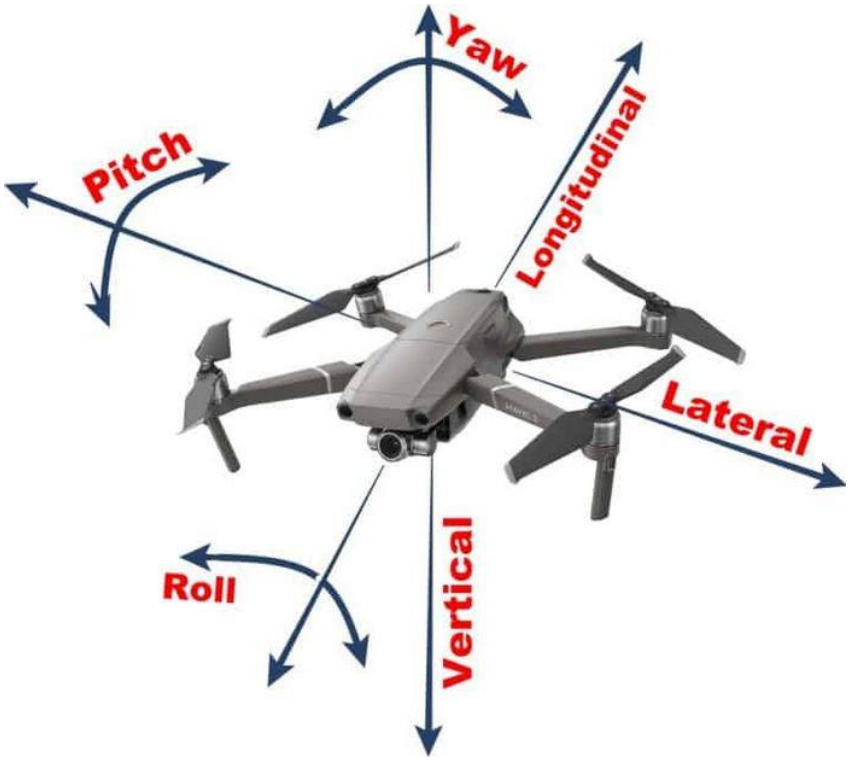


**Figura 1**

## 1.2. Sistemi di riferimento

Per non fare confusione e per lo studio del modello matematico servono dei sistemi di riferimento, per l'esattezza ne verranno utilizzati tre: uno inerziale denominato $I=\{x_i,y_i,z_i\}$, uno con origine nel centro di massa del drone e solidale ad esso, denominato $B=\{x_b,y_b,z_b\}$ e l'ultimo, denominato $V=\{x_v,y_v,z_v\}$, anch'esso avente origine nel centro di massa del quadricottero, ma allineato con il sistema di riferimento inerziale. In Figura 2 si può notare come l'asse delle Z, per tutti e tre i sistemi di riferimento sia rivolto verso il basso e non verso l'alto.

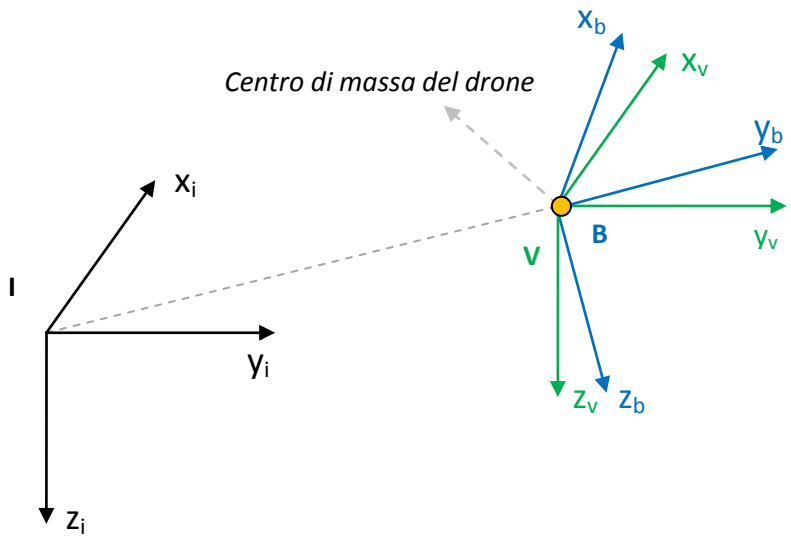


**Figura 2**

Dai sistemi di riferimento si possono successivamente distinguere due possibili configurazioni di un quadricottero: la configurazione "a croce", nella quale le eliche sono disposte lungo gli assi $X_b$ ed $Y_b$, oppure la configurazione "ad X", nella quale le eliche sono sfasate di 45° rispetto la configurazione precedente. La differenza fra le due configurazioni sta nel modo in cui si agisce sulle eliche per determinare gli spostamenti o l'equilibrio del drone.

Per esempio, osservando la Figura 3, nel modello a croce se si desidera compiere un beccheggio basterà lavorare sui motori 1 e 3, mentre nella configurazione ad X si dovrà agire sulle coppie 1-2 e 3-4. Le due configurazioni hanno in comune il fatto che la coppia di eliche opposte secondo la diagonale ha un senso di rotazione inverso a quello del'altra. Avere due sensi di rotazione è

utile ad annullare il momento che le eliche in movimento imprimono sul drone e quindi evitarne la conseguente rotazione su se stesso.

Per annullare il momento basterebbe che una coppia qualsiasi del drone abbia un senso di rotazione inverso rispetto all'altra, ma si sono scelte le coppie diagonali perché l'alternativa porterebbe a dei problemi di controllo, specialmente per l'imbardata, nella quale sarebbe molto difficile evitare beccheggi o rollii indesiderati.

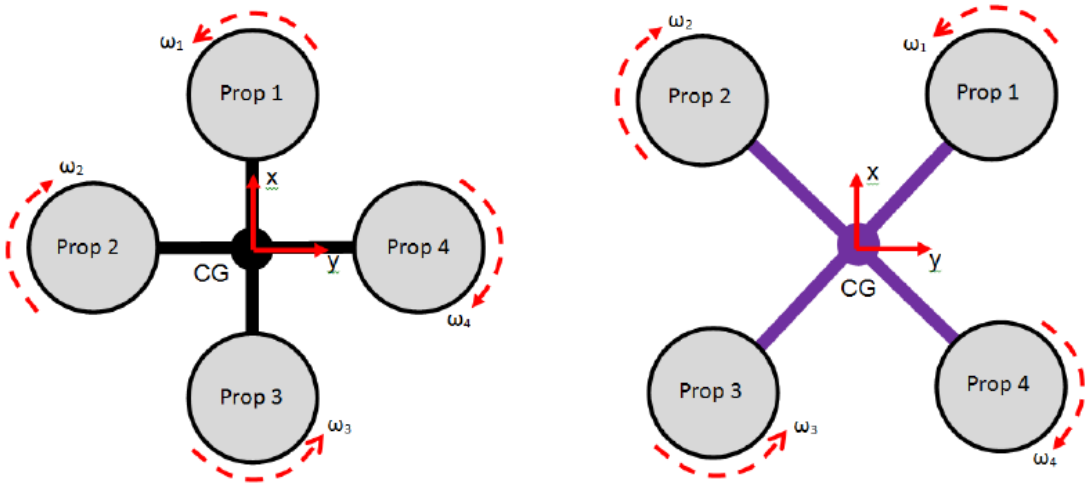


**Figura 3**

Per conoscere l'orientamento di un corpo nello spazio si utilizzano i cosiddetti "Angoli di Eulero" (mostrati in Figura 4), i quali sono una terna di angoli che misurano quanto un sistema di riferimento tridimensionale sia sfasato rispetto ad un altro.

Gli angoli di Eulero sono quindi:

- $\phi$ : indica la rotazione attorno all'asse X
- $\theta$ : indica la rotazione attorno all'asse Y
- $\psi$ : indica la rotazione attorno all'asse Z

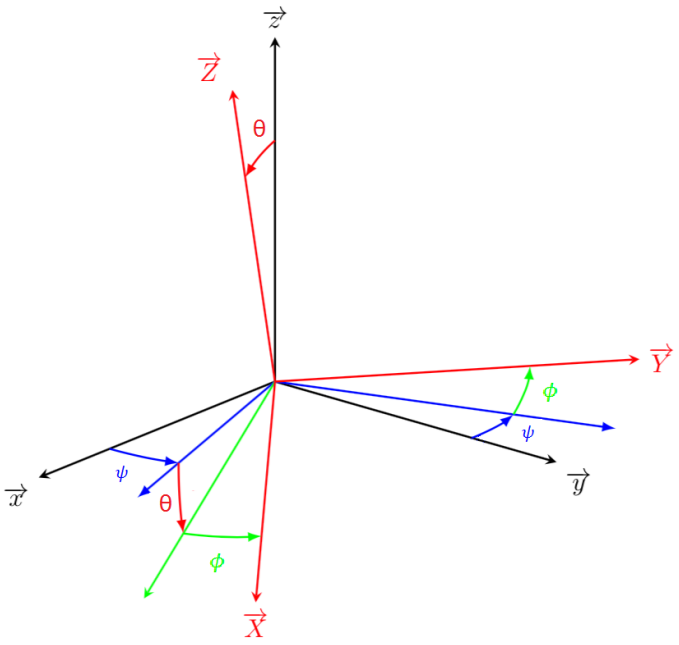


**Figura 4**

Per esprimere le coordinate di un punto che ha subito una rotazione di un determinato angolo attorno ad uno dei tre assi vengono impiegate le tre matrici di rotazione $R_x$, $R_y$ e $R_z$.

Le tre matrici valgono:

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

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

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

ognuna della quali ha come asse di rotazione quello indicato dal pedice e come "assi liberi" gli altri 2. Queste matrici se moltiplicate fra di loro forniscono la matrice di rotazione completa:

$$R_B^I(\phi,\theta,\psi) = R_z(\psi)\cdot R_y(\theta)\cdot R_x(\phi) =$$

$$= \begin{bmatrix} \cos\psi\cos\theta & \cos\psi\sin\theta\sin\phi - \sin\psi\cos\phi & \cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi \\ \cos\theta\sin\phi & \sin\psi\sin\theta\sin\phi + \cos\psi\cos\phi & \sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi \\ -\sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{bmatrix} \quad (4)$$

Con questa matrice è possibile trasformare l'espressione delle coordinate nel sistema di riferimento solidale al drone, all'espressione nel sistema di riferimento inerziale, a patto che si conosca lo sfasamento fra i due sistemi di riferimento. Infatti, se il punto ha coordinate [$x_b$, $y_b$, $z_b$] per il s.d.r. B, rispetto ad I avrà coordinate:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = R_B^I(\phi,\theta,\psi)\cdot\begin{bmatrix} x_b \\ y_b \\ z_b \end{bmatrix} \quad (5)$$

Per il passaggio inverso è sufficiente utilizzare la trasposta della matrice di rotazione $R_B^I$, questo è possibile perché le tre matrici dalla quale deriva sono ortonormali, ossia la loro trasposta coincide con la loro inversa. L'ortonormalità viene quindi ereditata anche da $R_B^I$, rendendo possibile il seguente passaggio dalle coordinate di I a quelle di B:

$$\begin{bmatrix} x_b \\ y_b \\ z_b \end{bmatrix} = (R_B^I(\phi,\theta,\psi))^T\cdot\begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad (6)$$

Oltre alle coordinate, si vogliono conoscere anche le velocità del drone, lineari ed angolari:

- **Velocità lineari**

  Per passare dalle velocità espresse nel sistema di riferimento B al sistema di riferimento inerziale si adopera la matrice di rotazione $R_B^I$ riportata nell'equazione **(4)**:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = R_B^I(\phi,\theta,\psi)\cdot\begin{bmatrix} u \\ v \\ w \end{bmatrix} \quad (7)$$

  il vettore colonna $v_I = [\dot{x} \quad \dot{y} \quad \dot{z}]^T$ è il vettore delle velocità di traslazione nel sistema di riferimento inerziale, mentre il vettore colonna $v_B = [u \quad v \quad w]^T$ è quello delle velocità espresse rispetto a B.

- **Velocità angolari**

  È altresì importante conoscere con che velocità avvengono le rotazioni del quadricottero attorno agli assi del s.d.r. I. Sia $\omega_i = [\dot{\phi} \quad \dot{\theta} \quad \dot{\psi}]^T$ il vettore colonna contenente le tre velocità angolari espresse rispetto al sistema di riferimento inerziale, esso può essere ottenuto a partire da:

$$\omega_b = [p \quad q \quad r]^T \tag{8}$$

  ossia il vettore delle velocità angolari espresse rispetto a B, mediante la matrice di trasformazione $Q_B^I(\phi, \theta, \psi)$.

  Questa matrice è il risultato dell'operazione di inversione della matrice $Q_I^B(\phi, \theta, \psi)$, ottenuta a sua volta con i seguenti passaggi, utili a portare i valori delle velocità angolari dal s.d.r. I al s.d.r. B:

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = R_x(\phi) \cdot \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + R_x(\phi)R_y(\theta) \cdot \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + R_x(\phi)R_y(\theta)R_z(\psi) \cdot \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} \tag{9}$$

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = Q_I^B(\phi, \theta, \psi) \cdot \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{10}$$

  Una volta presi i valori delle matrici dalle equazioni **(1)**, **(2)** e **(3)**:

$$Q_I^B(\phi, \theta, \psi) = \begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \sin\phi\cos\theta \\ 0 & -\sin\phi & \cos\phi\cos\theta \end{bmatrix} \tag{11}$$

  Dopo l'operazione di inversione si ottiene dunque la matrice:

$$Q_B^I(\phi, \theta, \psi) = \begin{bmatrix} 1 & \sin\phi\dfrac{\sin\theta}{\cos\theta} & \cos\phi\dfrac{\sin\theta}{\cos\theta} \\ 0 & \cos\phi & -\sin\theta \\ 0 & \dfrac{\sin\phi}{\cos\theta} & \dfrac{\cos\phi}{\cos\theta} \end{bmatrix} \tag{12}$$

Questa è la matrice che più risulta essere utile nella trasformazione dei dati provenienti dai sensori. Infatti il vettore delle velocità angolari misurate al bordo del drone possono essere portate nel sistema di riferimento inerziale tramite questa matrice.

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = Q_B^I(\phi, \theta, \psi) \cdot \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{13}$$

## 1.3. Forze e momenti

La dinamica del quadricottero è stata studiata su di un generale modello in configurazione a "X" per avere un riscontro con il drone Parrot Mambo. Per questo studio è necessario analizzare le forze a cui esso è sottoposto, perciò saranno da considerare le seguenti grandezze fisiche: accelerazione gravitazionale, attrito viscoso, forze e momenti impressi dai motori.

Dalle equazioni di Eulero si possono ricavare le formule per descrivere la dinamica del mezzo, queste equazioni risultano essere:

$$\boldsymbol{m\dot{v}_I = F_t^I + F_p^I + F_f^I} \tag{14}$$

$$\boldsymbol{I_B\dot{\omega}_b + \omega_b \times (I_b\omega_b) = \tau_r^b} \tag{15}$$

nelle quali $m$ indica la massa del drone, $\dot{v}_I$ l'accelerazione espressa secondo il s.d.r. inerziale, $I_B$ il momento di inerzia del drone e $\tau_r^b$ il momento complessivo generato dai quattro rotori (la r sta per "*rotors*"). $F_t^I$, $F_p^I$ e $F_f^I$ sono rispettivamente: la forza di spinta complessiva dei motori (la t sta per "*thrust*", ossia "*spinta*"), la forza peso e la forza di attrito viscoso (la f sta per "*friction*", ossia "*attrito*").

**Forze**

La spinta dei motori nel s.d.r. solidale al drone risulta sempre di verso opposto all'asse $z_b$, per cui si può adoperare la matrice di rotazione $R_B^I$ per ottenere la risultante nel s.d.r. inerziale:

$$F_t^I = R_B^I(\phi, \theta, \psi) \cdot F_t^B \tag{16}$$

$F_t^B$ vale a sua volta:

$$F_t^B = \begin{bmatrix} 0 \\ 0 \\ -k\sum_{i=1}^{4} \omega_i^2 \end{bmatrix} \tag{17}$$

k è una costante descritta dalla formula $k = \left(\frac{k_v k_\tau}{k_t}\right)^2 2\rho A$, nella quale $k_t$, $k_v$ e $k_\tau$ sono rispettivamente: coefficiente del momento torcente, coefficiente della forza elettromotrice e rapporto fra il momento e la spinta. $A$, invece, rappresenta l'area percorsa dal rotore. La forza ha segno negativo, in quanto ha verso opposto all'asse $z_b$.

Per quanto riguarda la forza peso, questa rimane sempre direzionata verso il centro della terra ed è:

$$F_p^I = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} \tag{18}$$

vista dal s.d.r. solidale al drone risulta invece:

$$F_p^B = R_I^B(\phi, \theta, \psi) \cdot \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} = \begin{bmatrix} -mg\sin\theta \\ mg\cos\theta\sin\phi \\ mg\cos\theta\ \cos\phi \end{bmatrix} \tag{19}$$

L'ultima forza da analizzare è la forza di attrito viscoso, a cui il drone è sottoposto durante i suoi movimenti. La forza di attrito esercitata sul drone assume una forma come quella dell'equazione **(20)**, in quanto il numero di Reynolds per il quadricottero risulta essere $R_e > 10^4$. $c_d$ è un coefficiente di attrito che dipende dalla forma del drone, $\rho$ è la densità dell'aria, $A$ l'area della superficie ad impattare con l'aria durante lo spostamento e $v$ la velocità relativa fra drone ed aria.

$$F_D = \frac{1}{2} c_d \rho A v^2 \tag{20}$$

Perciò la forza di attrito finale sarà:

$$F_f^I = R_B^I(\phi,\theta,\psi)\cdot F_f^b = R_B^I(\phi,\theta,\psi)\cdot\begin{bmatrix}-\frac{1}{2}c_d\rho A^f u^2\\ -\frac{1}{2}c_d\rho A^s v^2\\ -\frac{1}{2}c_d\rho A^t w^2\end{bmatrix} = -\frac{1}{2}c_d\rho\begin{bmatrix}A^f\dot{x}^2\\ A^s\dot{y}^2\\ A^t\dot{z}^2\end{bmatrix} \tag{21}$$

le lettere *f*, *s* e *t* stanno per "*front*", "*side*" e "*top*".

Ora si hanno tutte le componenti della prima equazione della dinamica **(14)**, quindi la si può completare utilizzando **(16)**, **(18)** e **(21)**:

$$\begin{bmatrix}\ddot{x}\\ \ddot{y}\\ \ddot{z}\end{bmatrix} = -\frac{k}{m}R_B^I\begin{bmatrix}0\\ 0\\ k\sum_{i=1}^{4}\omega_i^2\end{bmatrix} + \begin{bmatrix}0\\ 0\\ g\end{bmatrix} - \frac{1}{2m}c_d\rho\begin{bmatrix}A^f\dot{x}^2\\ A^s\dot{y}^2\\ A^t\dot{z}^2\end{bmatrix} \tag{22}$$

Adesso si può procedere con l'analisi delle componenti che compongono l'equazione **(15)**.

**Momenti**

Come prima cosa si porta l'equazione **(15)** nella forma tipica delle equazioni descriventi una dinamica:

$$\dot{\omega}_b = I_B^{-1}\left(\tau_r^b - I_B\omega_b\right) \tag{23}$$

La matrice del momento d'inerzia $I_B$, per un drone simmetrico rispetto ai vari assi, appare come una matrice diagonale:

$$I_B = \begin{bmatrix}I_{xx} & 0 & 0\\ 0 & I_{yy} & 0\\ 0 & 0 & I_{zz}\end{bmatrix} \tag{24}$$

Il momento generato $\tau_r^b$ è dovuto sia dalla distribuzione della spinta verticale fra i motori che dall'attrito delle eliche con l'aria.

Le variazioni degli angoli di beccheggio e di rollio sono determinate appunto dalla differenza di spinta nelle coppie di lati opposti del drone. Dunque se $d_{12} = d_{34}$ e $d_{14} = d_{23}$ sono le distanze

dal centro di massa del drone dalla rispettiva coppia di rotori, i momenti di rollio e beccheggio valgono rispettivamente:

$$\tau_{\phi}^{b} = d_{14}k(\omega_1^2 - \omega_2^2 - \omega_3^2 + \omega_4^2) \tag{25}$$

$$\tau_{\theta}^{b} = d_{12}k(\omega_1^2 + \omega_2^2 - \omega_3^2 - \omega_4^2) \tag{26}$$

Per l'angolo di imbardata bisogna invece considerare la risultante del momento prodotto dalle coppie di eliche aventi senso di rotazione opposto. Per ottenere il momento prodotto dalle eliche si parte dall'attrito che queste hanno con l'aria. Dalla formula già vista in **(20)** si può passare al caso delle eliche, sostituendo la sezione delle eliche *S* ad *A* , prendendo come $v$ la velocità tangenziale dell'elica e utilizzando $c_{de}$ come coefficiente d'attrito dipendente dalla forma dell'elica. Dato che non si è a conoscenza della velocità tangenziale delle eliche, ma si conoscono il raggio *R* e la velocità angolare, si effettua una trasformazione:

$$F_D = \frac{1}{2} c_{de} \rho S (R\omega)^2 \tag{27}$$

Per passare al momento si moltiplica la forza per il suo braccio, ossia il raggio *R* dell'elica:

$$\tau = \frac{1}{2} R c_d \rho S \omega_i^2 \tag{28}$$

Il risultato in **(28)** è in realtà approssimato, difatti con il la moltiplicazione per *R* si assume che tutta la forza agisca solamente agli estremi dell'elica. Questa approssimazione però non ha molta rilevanza ai fini del risultato, perché ciò che interessa maggiormente è il legame presente tra momento e velocità angolare.

Il momento sull'angolo di imbardata risulta quindi:

$$\tau_{\psi}^{b} = b(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2) \tag{29}$$

con $b = \frac{1}{2} R c_d \rho S$. Anche in questo caso è stata effettuata una approssimazione, difatti la formula complessiva dovrebbe essere $\tau_\psi^b = b\omega^2 + I_M \dot{\omega}$, nella quale $I_M$ è il momento d'inerzia attorno all'asse centrato nel rotore e parallelo a $z_b$, mentre $\dot{\omega}$ l'accelerazione angolare dell'elica. Dato che però $\dot{\omega}$ è spesso prossima allo zero si decide appunto di trascurare il secondo termine.

Il momento complessivo quindi è il vettore composto dai risultati **(25)**, **(26)**, **(29)**

$$\tau_r^b = \begin{bmatrix} \tau_\phi^b \\ \tau_\theta^b \\ \tau_\psi^b \end{bmatrix} \tag{30}$$

Giunti a questo punto si può riprendere l'equazione **(23)** e completarla con le equazioni **(8)**, **(24)** e **(30)** per ottenere l'equazione della dinamica della velocità angolare del drone:

$$\begin{bmatrix} \dot{\boldsymbol{p}} \\ \dot{\boldsymbol{q}} \\ \dot{\boldsymbol{r}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{I_{xx}^{-1}} \boldsymbol{\tau_\phi^b} \\ \boldsymbol{I_{yy}^{-1}} \boldsymbol{\tau_\theta^b} \\ \boldsymbol{I_{zz}^{-1}} \boldsymbol{\tau_\psi^b} \end{bmatrix} - \begin{bmatrix} \dfrac{\boldsymbol{I_{zz} - I_{yy}}}{\boldsymbol{I_{xx}}} \boldsymbol{rq} \\ \dfrac{\boldsymbol{I_{xx} - I_{zz}}}{\boldsymbol{I_{yy}}} \boldsymbol{pr} \\ \dfrac{\boldsymbol{I_{yy} - I_{xx}}}{\boldsymbol{I_{zz}}} \boldsymbol{rq} \end{bmatrix} \tag{31}$$

# 2. Il minidrone Parrot Mambo

Il minidrone Mambo (Figura 5),di fabbricazione Parrot, è un drone di piccole dimensioni, misura infatti 18x18 cm paraurti compresi ed ha un peso di soli 63g. Il drone è dotato di una sensoristica costituita da:

- Fotocamera a 60 FPS, utilizzata per la stima delle velocità lungo il piano orizzontale
- Sensore ad ultrasuoni, che fornisce una misura diretta dell'altezza da terra
- Sensore di pressione, per la stima dell'altitudine
- IMU, ossia (Unità Misura Inerziale), che attraverso un giroscopio ed un accelerometro, ciascuno a tre assi, rileva le variazioni d'orientamento, le accelerazioni del quadricottero ed eventuali urti

La fotocamera ed il sensore ad ultrasuoni sono posti nel lato inferiore del drone, mentre gli altri due risiedono all'interno.

Il Parrot Mambo è alimentato da una batteria LiPo ricaricabile da 660mAh, in grado di fornire energia al drone per un volo di circa 10 minuti senza accessori collegati.

È possibile caricare sul drone un algoritmo di volo tramite PC attraverso una porta USB e anche collegarsi tramite bluetooth per l'invio dei comandi di volo.

**Figura 5**

# 3. Modello matematico in Simulink

In Simulink è possibile trovare il modello matematico del Parrot Mambo all'interno del blocco "*Airframe*". Questo blocco prende come ingressi le velocità dei motori e le condizioni ambientali per fornire in uscita gli stati del quadricottero, ovvero le posizioni e le relative velocità sui sei gradi di libertà (Figura 6).

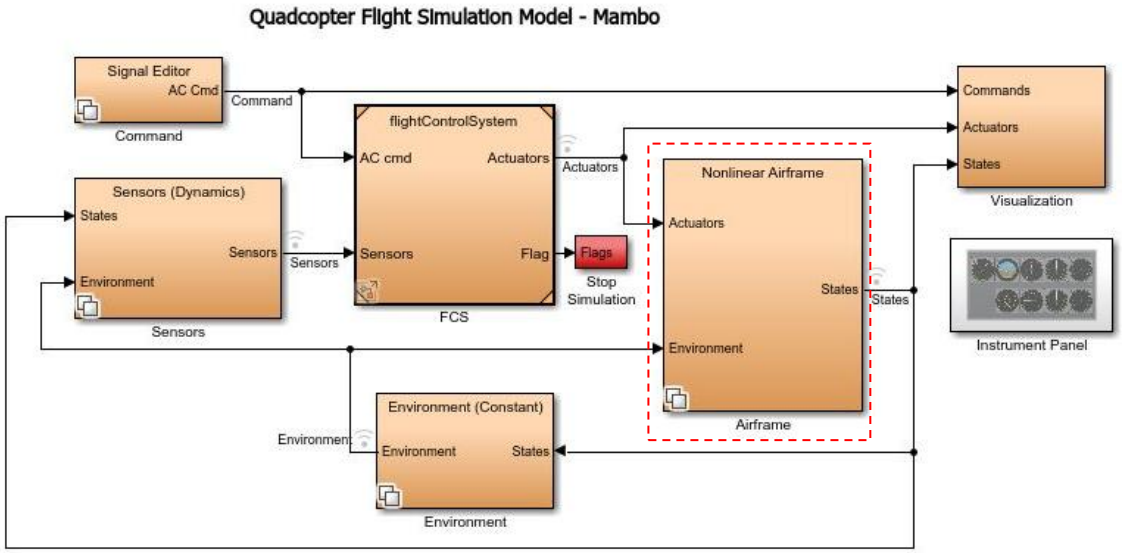


**Figura 6**

Una volta aperto "*Airframe*" è possibile selezionare il modello matematico che si preferisce tra quello non lineare o quello lineare (Figura 7). La selezione avviene attraverso alla variabile *VSS_VEHICLE*, che di default ha un valore pari ad 1. Se la variabile non viene alterata, il modello matematico selezionato è quello non lineare, se invece si desidera utilizzare il modello matematico lineare occorre, tramite la Command Window, portare *VSS_VEHICLE* a 0.

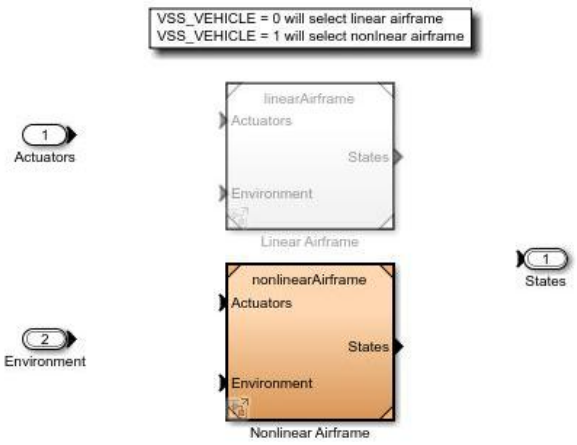


**Figura 7**

# 3.1. Modello matematico non lineare

Se il valore di *VSS_VEHICLE* è pari ad 1 ad essere selezionato è quindi il modello matematico non lineare. Questo modello è contenuto nel blocco "*Nonlinear_Airframe*" ed è formato da altri 4 blocchi, visibili in Figura 8.

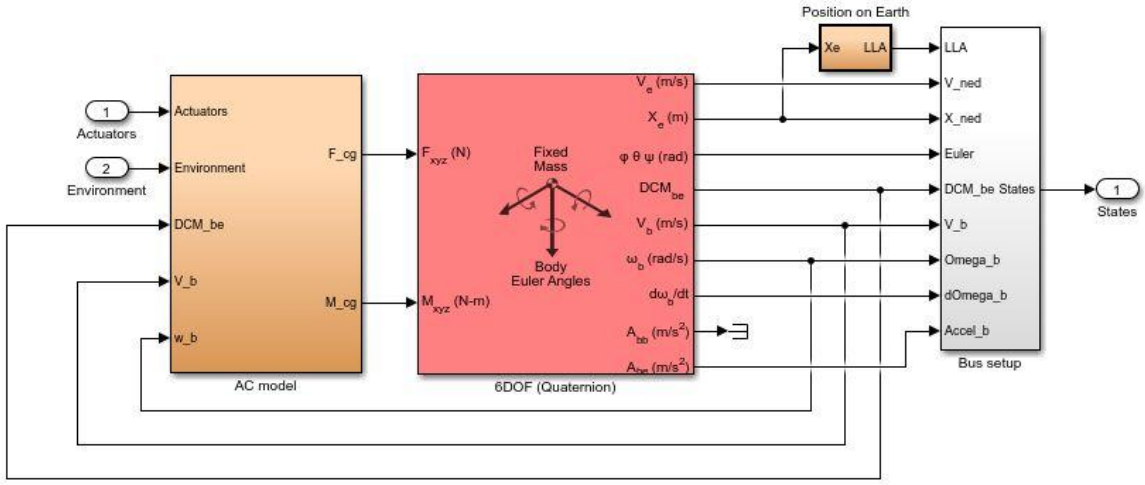


**Figura 8**

Il blocco "*AC model*" prende come ingressi: le velocità dei motori, le condizioni ambientali, le velocità dei sei gradi di libertà espresse rispetto al s.d.r. B e la Direction Cosine Matrix, necessaria per tornare al sistema di riferimento inerziale. Dall'elaborazione di questi dati fornisce in uscita le risultanti delle forze e dei momenti che agiscono sul drone. Queste due risultanti sono portate in ingresso al blocco "*6DOF*", un blocco appartenente all'*Aerospace*

*Blockset*. "*6DOF*" implementa la rappresentazione in quaternioni delle equazioni di un moto a sei gradi di libertà rispetto al s.d.r. solidale al quadricottero (esistono vari blocchi di questo tipo e sono mostrati nel seguente link MathWorks). In uscita il blocco "*6DOF*" fornisce gli stati del sistema, ovvero:

- velocità, posizioni e orientamento rispetto al sistema di riferimento inerziale $(V_e, X_e, \varphi, \theta, \psi)$
- matrice dei coseni direttori $(DCM)$
- velocità e accelerazione di traslazione rispetto B $(V_b, A_{bb})$
- velocità e accelerazioni angolari espresse rispetto B $(\omega_b, \frac{d\omega_b}{dt})$
- accelerazione di traslazione del sistema di riferimento B rispetto ad I $(A_{be})$

Questi stati entrano poi nel blocco "*Bus setup*", nel quale vengono raggruppati in un unico bus denominato "*States*".

Il blocco "*Position on Eatrh*" (Figura 9) ha il compito di passare dalle coordinate del drone espresse rispetto al riferimento inerziale alle coordinate terrestri (*LLA*), ossia longitudine, latitudine e altitudine. Il suo funzionamento prevede la somma delle coordinate terrestri della sede della MathWorks alle coordinate del sistema di riferimento I.

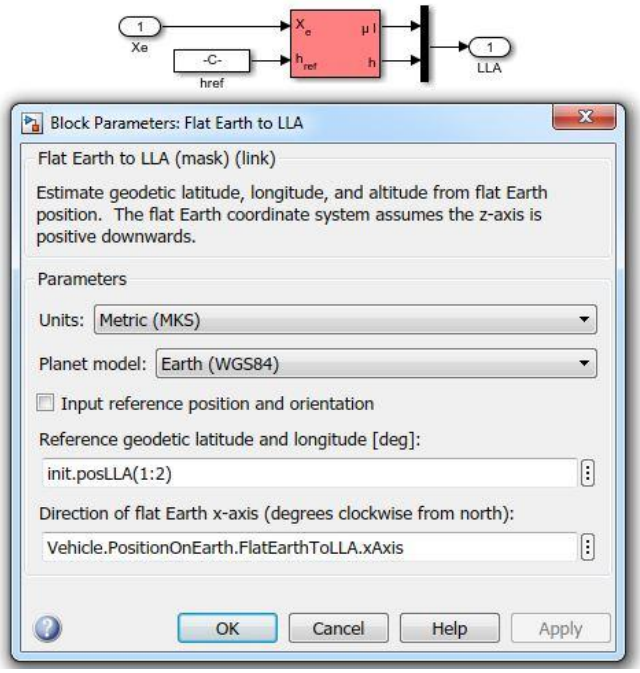


**Figura 9**

Andando ad esaminare più a fondo "*AC model*", in Figura 10 se ne osserva la sua composizione interna e si nota essere costituito da altri quattro blocchi:

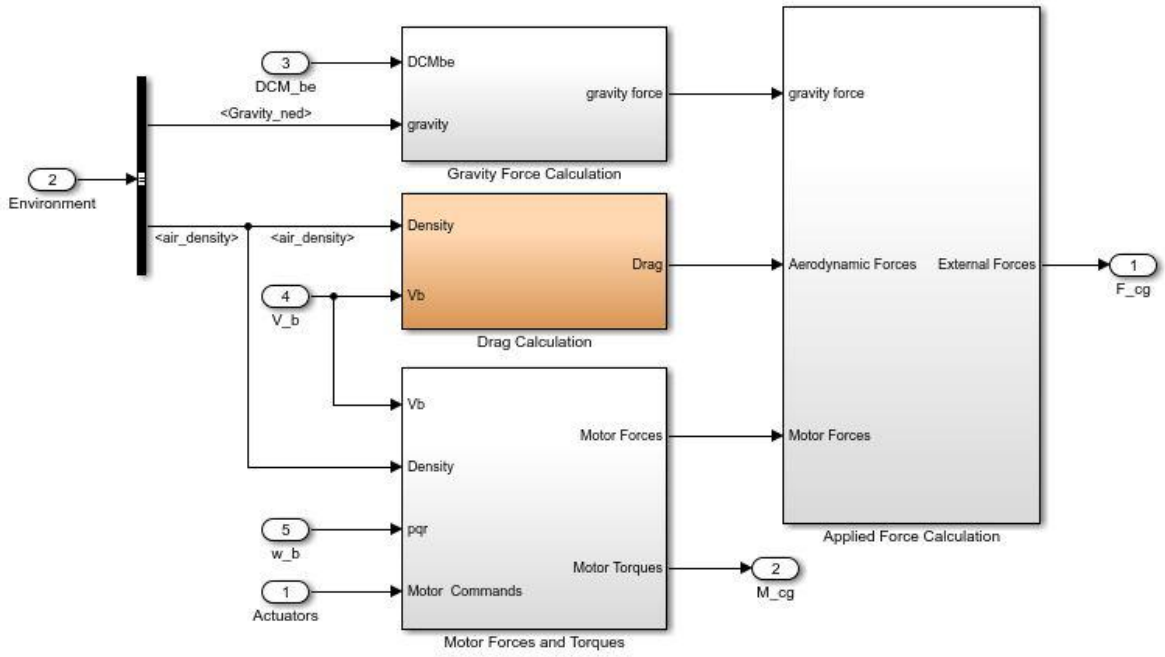


**Figura 10**

- "*Gravity Force Calculation*" (Figura 11)
  questo blocco ottiene quanto riportato dell'equazione **(19)**. Infatti si moltiplica *gravity* per la costante *Vehicle.Airframe.mass*, ossia la massa del drone (63 grammi), ottenendo così la forza peso nel s.d.r. I. Per passare poi nel s.d.r. B si moltiplica per la $DCM_{be}$.

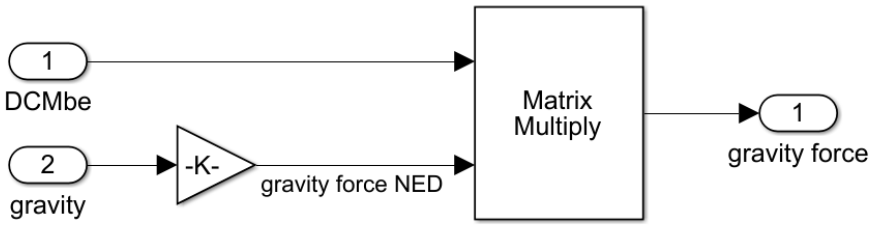


**Figura 11**

- "*Drag Calculation*" (Figura 12)
  in questo blocco si effettua il calcolo della forza d'attrito viscoso che si oppone al moto del quadricottero. La forza è calcolata nel blocco "*Drag Calc*" per poi essere invertita di segno a seconda del verso della velocità.

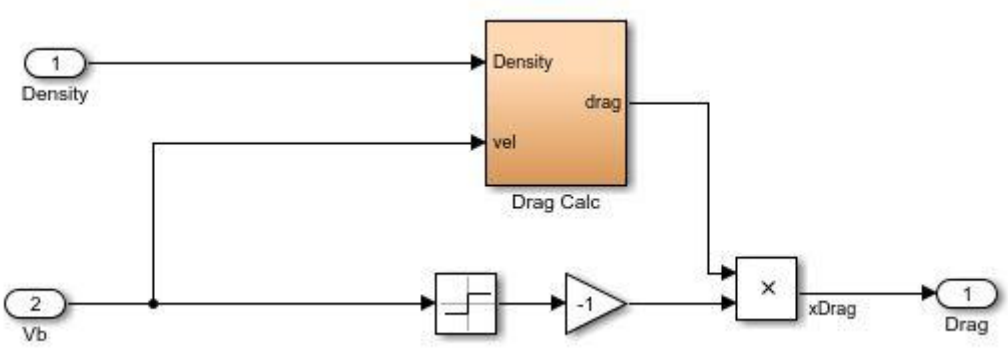


**Figura 12**

Esaminando l'interno del blocco "*Drag Calc*" (Figura 13) si nota che viene utilizzata la formula per l'attrito viscoso vista all'equazione **(20)**. In particolare, *vel* è la velocità del drone espressa nel s.d.r. B, *Cd* è il coefficiente di attrito ed *S* è la superficie ad impattare con l'aria. La fisionomia del drone è approssimata ad un parallelepipedo, infatti *S* è ottenuta dalla moltiplicazione della lunghezza (o larghezza, in quanto si equivalgono) del drone per i suoi lati, i quali sono supposti essere un quarto della lunghezza.

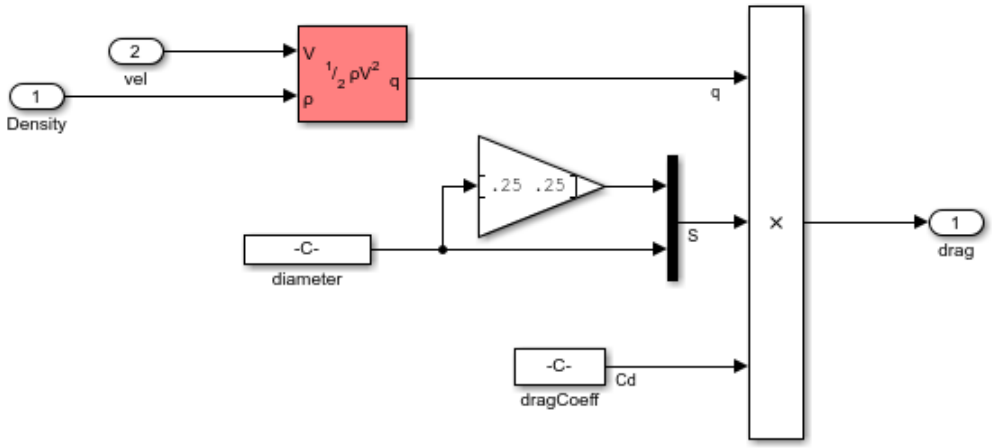


**Figura 13**

- "*Motor Forces and Torques*" (Figura 14)

  è il blocco che calcola le forze ed i momenti provocati dall'azione dei motori. Per avere conoscenza di come stano operando i motori si ha a disposizione il blocco "*Motors ToW*", nel quale sono presenti una serie di guadagni e limitazioni che trasformano il comando ai motori nel numero di giri a cui corrisponde. Quest'informazione, assieme alle velocità lineari e angolari del drone e la densità dell'aria, sono passate in input al blocco principale, che tramite un for each calcola il contributo che ogni singolo motore fornisce per la movimentazione del drone (Figura 15).

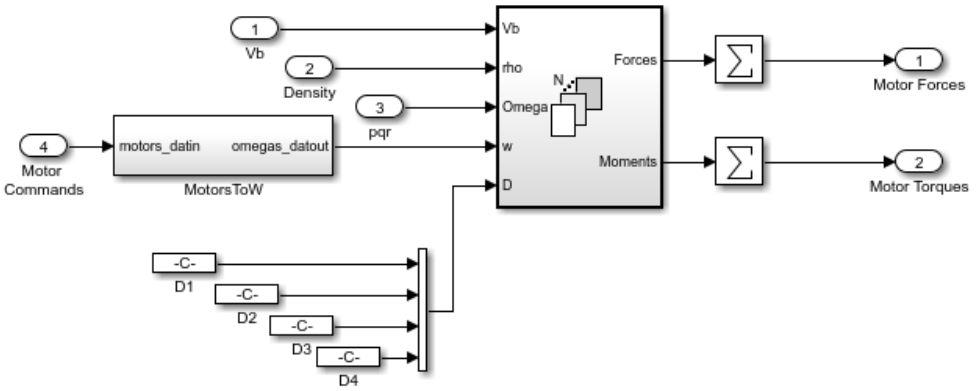


**Figura 14**

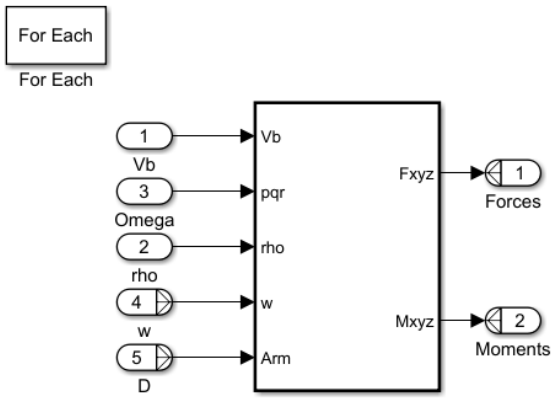


**Figura 15**

All'interno del blocco in Figura 15 si trova una struttura che, date le dimensioni, è stata divisa in due parti, una visibile in Figura 16 e l'altra, collegata a valle della prima, mostrata in Figura 17.

Queste due architetture sono in grado di calcolare le forze e i momenti provocati dal singolo motore a partire dalla sua velocità e posizione rispetto al centro di massa del drone. In particolare, viene all'inizio calcolata la velocità del veicolo attraverso la formula:

$$v_r = \omega_b \times D_i + v_b \tag{32}$$

nella quale $D_i$ rappresenta la distanza dell'i-esimo motore dal centro di massa e $v_r$ rappresenta la velocità del drone espressa rispetto a B. Viene poi calcolato il rapporto di avanzamento:

$$\mu_i = \frac{\left\| v_{r\,x,y} \right\|}{\omega_{mi}} \tag{33}$$

(in cui $\omega_{mi}$ è la velocità angolare dell'i-esimo motore). Questo valore viene usato per ricavare gli angoli di beccheggio e di rollio che il drone assume in base al suo avanzamento. Il risultato è combinato a sua volta con l'angolo che il drone forma con il piano orizzontale del s.d.r. inerziale. Questo angolo viene calcolato con la seguente formula:

$$\eta = \arctan\left(\frac{v_{r_y}}{v_{r_x}}\right) \tag{34}$$

Vengono sottratte poi le velocità angolari $p$ e $q$, precedentemente divise per la velocità del 'i-esimo motore a cui è stata moltiplicata la costante *Vehicle.Rotor.lock*. Dopodiché si procede con il calcolo finale delle forze e dei momenti sviluppati secondo le formule viste in 1.3. Forze e momenti.

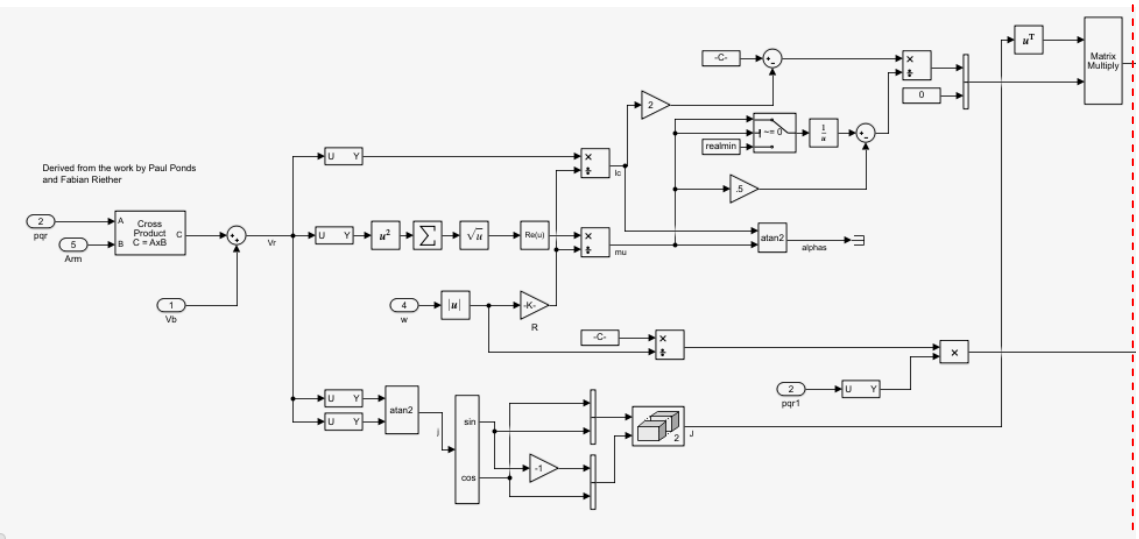


**Figura 16**

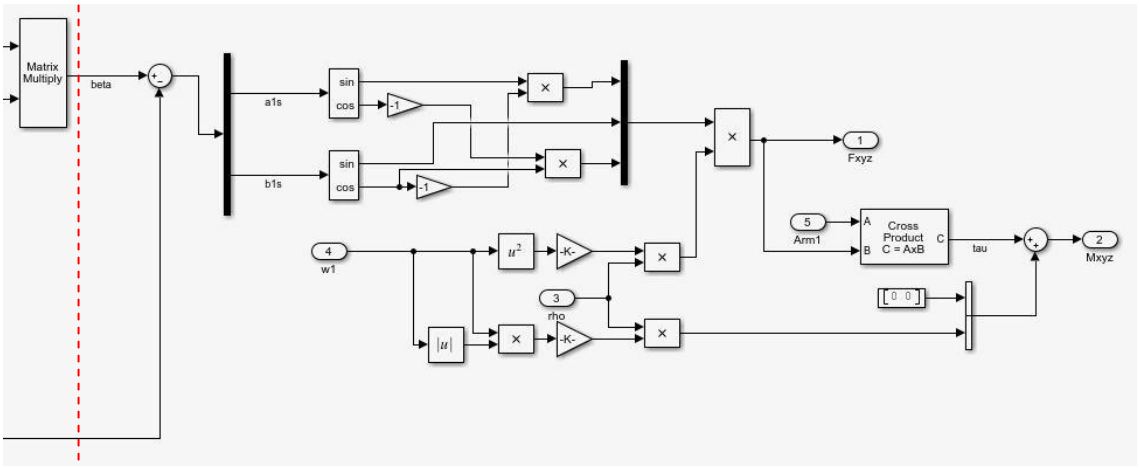


**Figura 17**

## 3.2. Modello matematico lineare

Nel blocco "*Airframe*" (Figura 7) se la variabile *VSS_VEHICLE* è pari a 0 viene selezionato il blocco "*Linear Airframe*", il quale contiene appunto il modello matematico lineare (Figura 18) . Questo modello risulta meno complesso rispetto al precedente, infatti è costituito principalmente da una quaterna di matrici A, B, C e D contenuta nel blocco "*Linear Model*".

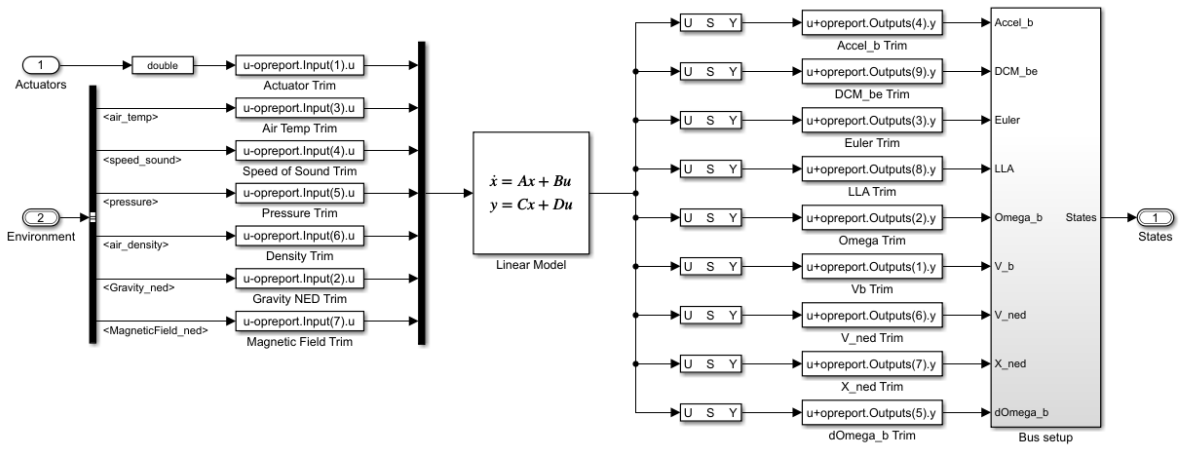


**Figura 18**

A sinistra del blocco "*Linear Model*" sono presenti gli ingressi, composti dai comandi ai motori e dalle variabili ambientali (che saranno poi discusse in 4.1.3. Ambiente esterno), mentre a

destra sono presenti gli stati. Gli stati vengono organizzati tramite dei selettori e poi, all'interno del blocco "*Bus_setup*", inseriti in un unico bus chiamato "*States*".

Il modello lineare è calcolato tramite la funzione '*trimLinearizeOpPoint'* del toolbox *'Simulink Control Design'*, attraverso il principio di *triming*. Questo principio consiste nel trovare la combinazione di stati del sistema e di input che siano in grado di fornire una situazione di stato stazionario in corrispondenza del punto di lavoro desiderato.

Una volta effettuata la simulazione, le matrici A, B, C e D potranno essere trovate all'interno della variabile spazio di stato 33x14 *linsys* (Figura 19). Questa variabile è salvata nel Work Space e al suo interno, oltre alle quattro matrici, si trovano anche i vettori *InputName*, *StateName* e *OutputName*, nei quali sono presenti i nomi degli ingressi, degli stati e delle uscite.

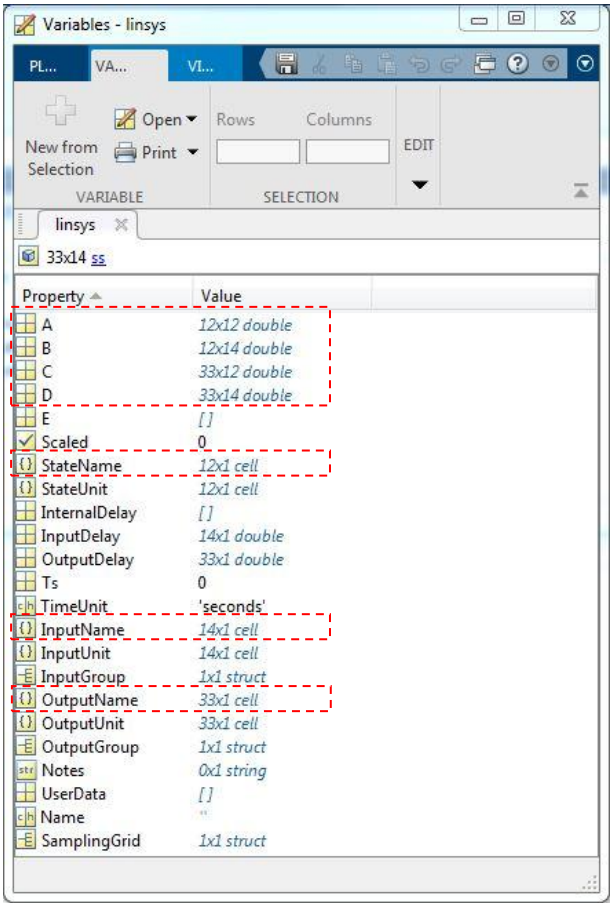


**Figura 19**

# 4. Controllo su Simulink

Facendo nuovamente riferimento alla Figura 6 si prosegue analizzando i restanti blocchi del modello in Simulink. Si continua esaminando quelli che circondano il controllore, per poi passare a quest'ultimo in 4.2. Il controllore FCS.

## 4.1. Composizione del modello complessivo attorno al controllore

Il controllore è il blocco denominato "*FCS*", perciò ora si andranno prima ad analizzare tutti i blocchi ai quali è connesso, al fine di identificare quelli che saranno poi i suoi ingressi e le sue uscite.

### 4.1.1. Generazione dei comandi

Il blocco "*Command*" è addetto alla generazione dei segnali di comando per il drone. Al suo interno si presenta come in Figura 20, e attraverso la variabile *VSS_COMMAND* è possibile selezionare la modalità di generazione dei segnali di riferimento. Le possibilità di scelta sono:

- *VSS_COMMAND* = 0 → segnali generati tramite un generatore di segnali
- *VSS_COMMAND* = 1 → segnali generati tramite un joystick accompagnato da un generatore di segnali
- *VSS_COMMAND* = 2 → segnali generati tramite un file.mat precedentemente salvato
- *VSS_COMMAND* = 3 → segnali generati da un foglio di calcolo precedentemente salvato

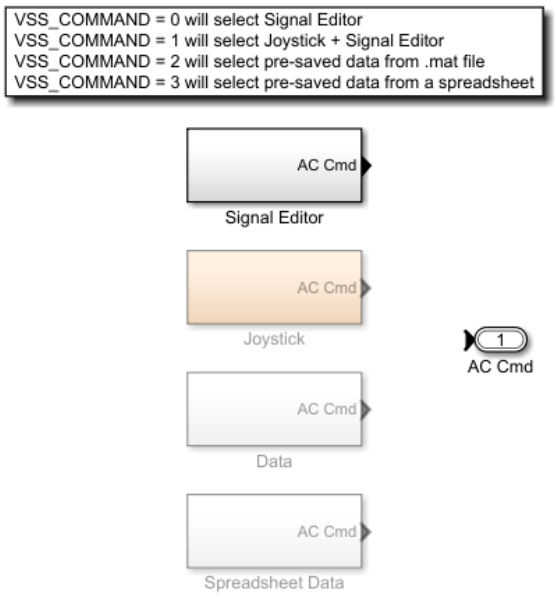


**Figura 20**

Per questo progetto la variabile *VSS_COMMAND* sarà pari a 0, dunque il blocco utilizzato sarà quello denominato "*Signal Editor*". L'interno di questo blocco, mostrato in Figura 21, presenta al suo interno un generatore di segnali, con il quale sono stati impostati i comandi, visibili in Figura 22, che verranno inviati al drone.

Con la porta AND si gestisce la variabile *controlModePosVSOrient* che andrà poi ad agire sui controllori. Più precisamente, se rollio e beccheggio sono entrambi comandati a zero, la porta AND fornirà un valore logico alto in uscita.

All'interno del "*Signal Editor*" si trova anche la generazione della variabile *takeoff_flag*, utile a distinguere una situazione di decollo da una di normale controllo. I comandi delle traslazioni e degli angoli vengono raggruppati in due strutture chiamate rispettivamente *"pos_ref"* ed *"orient_ref"*.

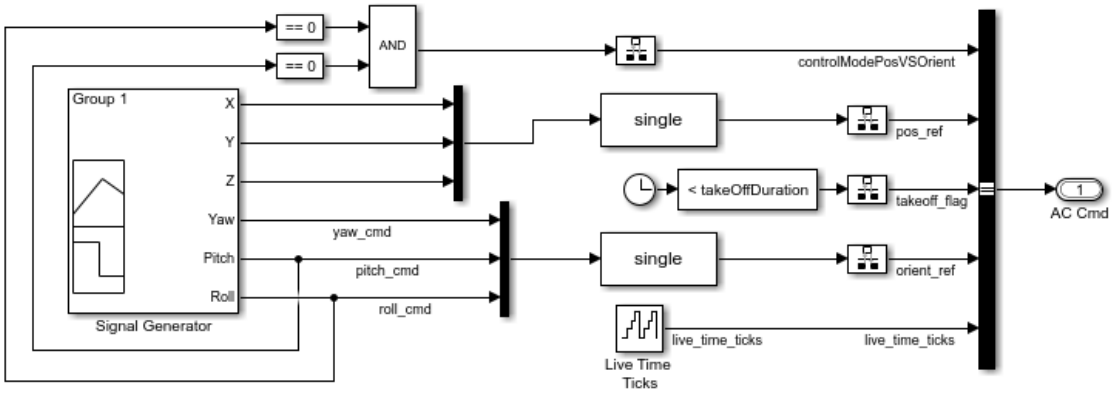


**Figura 21**

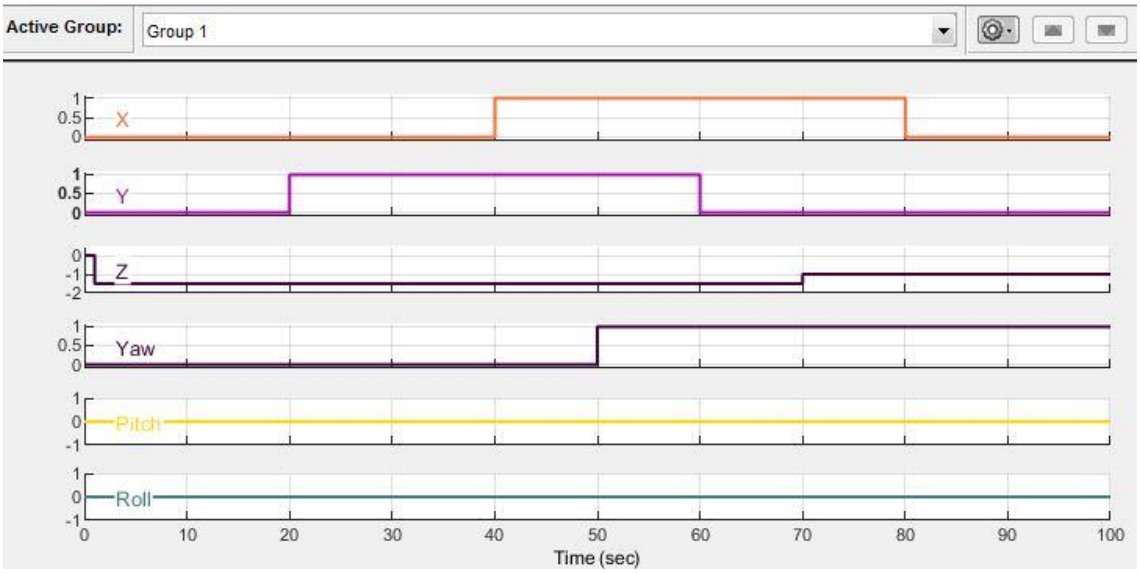


**Figura 22**

## 4.1.2. Sensori

I sensori del quadricottero sono simulati dal blocco "*Sensors*" (Figura 23), anch'esso composto di due blocchi selezionati da una variabile. La variabile *VSS_SENSORS* seleziona, se a 0, il blocco che non tiene conto dei rumori sui sensori, mentre se a 1 (impostazione di default), il blocco che ne tiene contro. Per questo progetto *VSS_SENSORS* rimarrà ad 1, dunque verrà utilizzato il blocco inferiore in Figura 23, che al suo interno appare come in Figura 24.

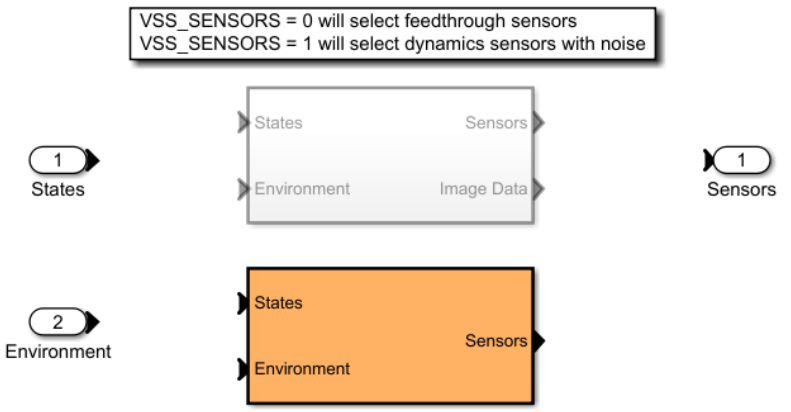


**Figura 23**

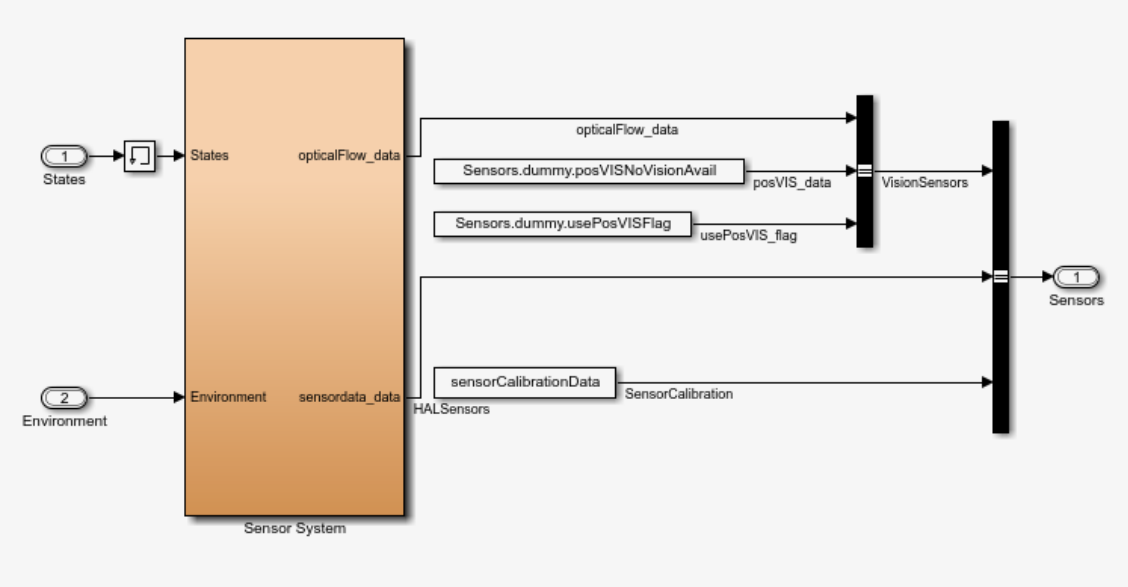


**Figura 24**

Effettuando l'accesso nel blocco "*Sensor System*" si giunge a ciò che è mostrato in Figura 25, qui si trovano i due sotto sistemi che simulano i diversi sensori del drone. In Figura 26 è mostrato il sottosistema della fotocamera, il quale attraverso l'acquisizione degli stati del drone, simula il comportamento che avrebbe la fotocamera in un volo reale e fornisce la misura in uscita. In Figura 27 è presente il sotto sistema dell'unità IMU, che oltre agli stati del sistema usufruisce in ingresso anche del vettore di gravità. In uscita dal blocco sono forniti i valori delle accelerazioni sia di traslazione che angolari espresse nel sistema di riferimento inerziale.

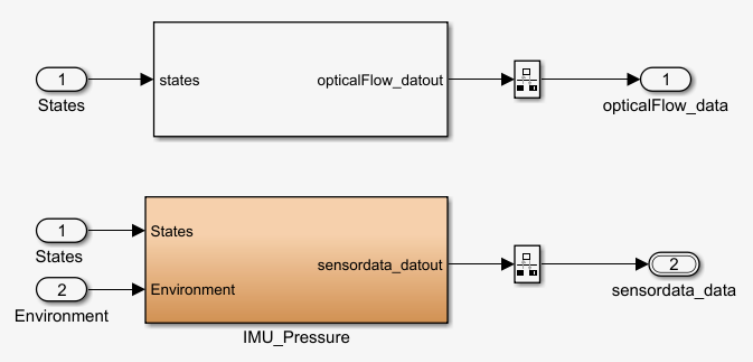


**Figura 25**

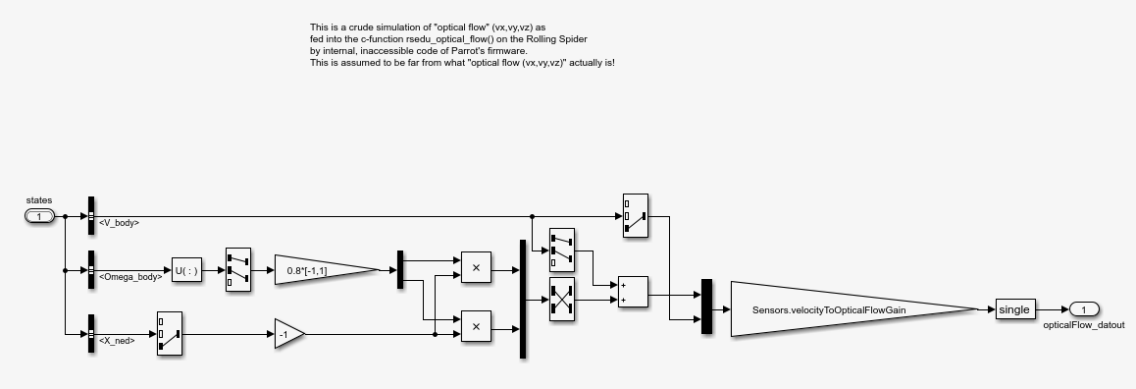


**Figura 26**

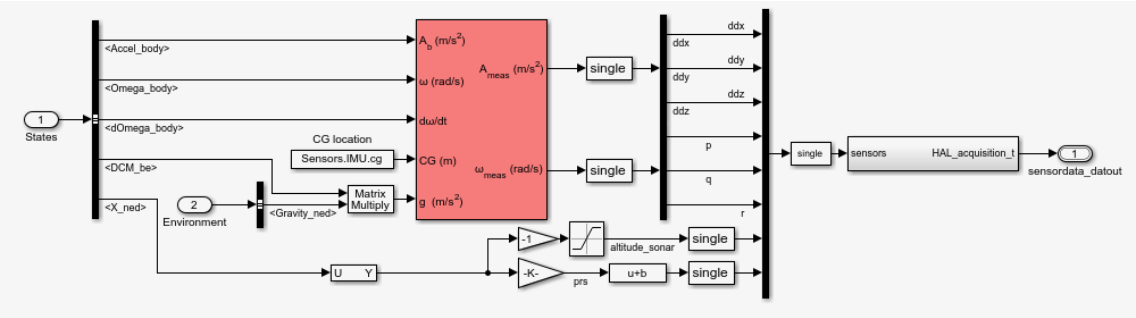


**Figura 27**

## 4.1.3. Ambiente esterno

Nel blocco "*Enviroment*" in Figura 28 sono contenuti i sottoblocchi selezionabili dalla variabile *VSS_ENVIROMENT*, di default a 0. Se il valore della variabile resta 0 viene selezionato il blocco superiore, nel quale le variabili ambientali sono delle costanti e non dipendono dalla posizione del drone nello spazio. Questo sotto blocco è mostrato in Figura 29, al suo interno viene semplicemente costruito un bus contente le costanti ambientali, ossia: vettore gravità,

vettore campo magnetico, velocità del suono, temperatura, densità e pressione dell'aria. Se *VSS_ENVIROMENT* fosse invece a 1, le grandezze precedentemente nominate non sarebbero più costanti, ma sarebbero in funzione della latitudine, della longitudine e dell'altitudine del drone.

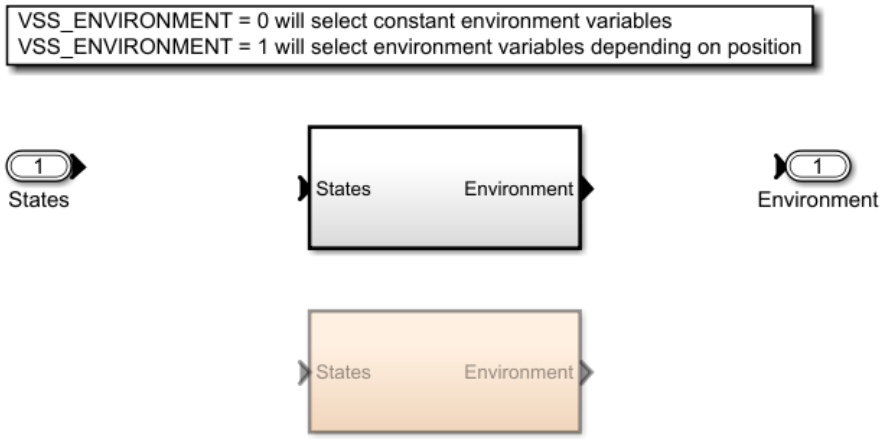


**Figura 28**

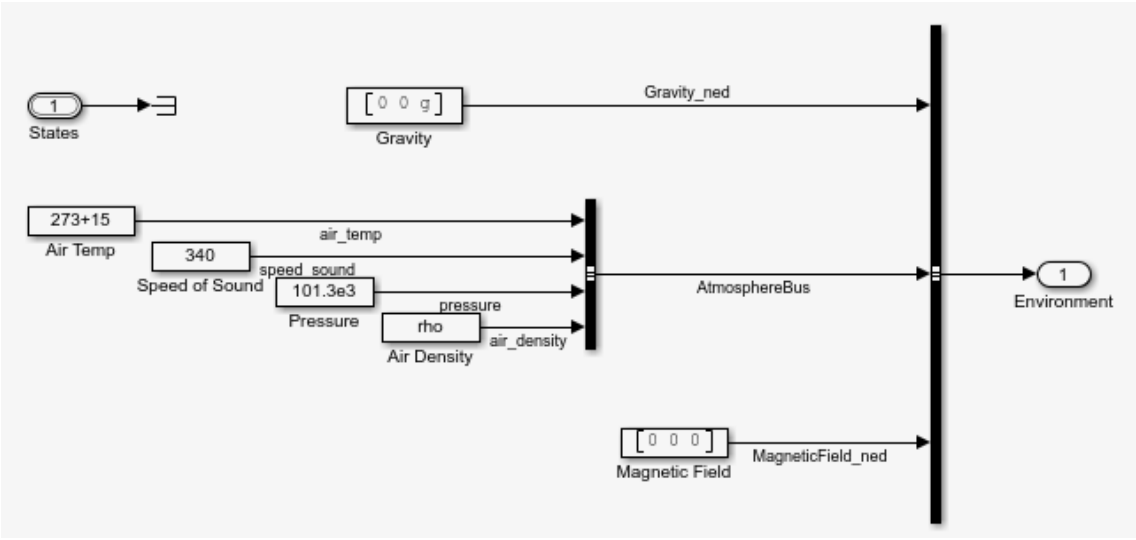


**Figura 29**

### 4.1.4. Interruzione della simulazione per anomalie

Il blocco "*Stop Simulation*" ha il compito di arrestare la simulazione se è stata riscontrata un'anomalia durante il volo (Figura 30). Il funzionamento consiste nel controllare se il suo ingresso differisce da zero e in tal caso arrestare la simulazione.

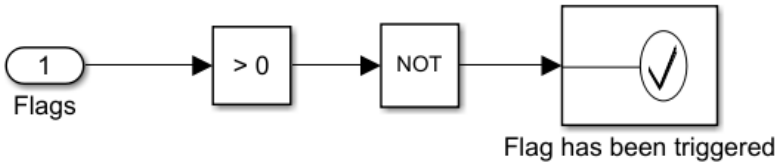


**Figura 30**

### 4.1.5. Visualizzazione della simulazione

Per la visualizzazione della simulazione sono disponibili, all'interno del blocco "*Visualization*", due sotto sistemi, uno per la visualizzazione di valori di volo come altitudine, latitudine e condizioni ambientali, e un altro, omonimo del blocco che lo contiene, per la visualizzazione della simulazione tridimensionale (Figura 31).

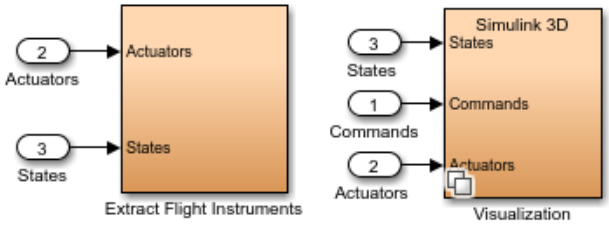


**Figura 31**

Per l'esattezza, nel secondo blocco "*Visualization*" è possibile selezionare tramite *VSS_VISUALIZATION* quale tipo di visualizzazione si desidera. Per questo progetto la variabile è lasciata, come di default a 3 per adoperare la visualizzazione 3D. La simulazione grafica visibile in Figura 32 può essere aperta dal blocco interno "*Simulink 3D*".

Un altro strumento di visualizzazione è il blocco "*Instrument Panel*", non contenuto nel blocco "*Visualization*", ma visibile in Figura 6. Al suo interno si presenta come una plancia di un generale aeromobile (Figura 33).

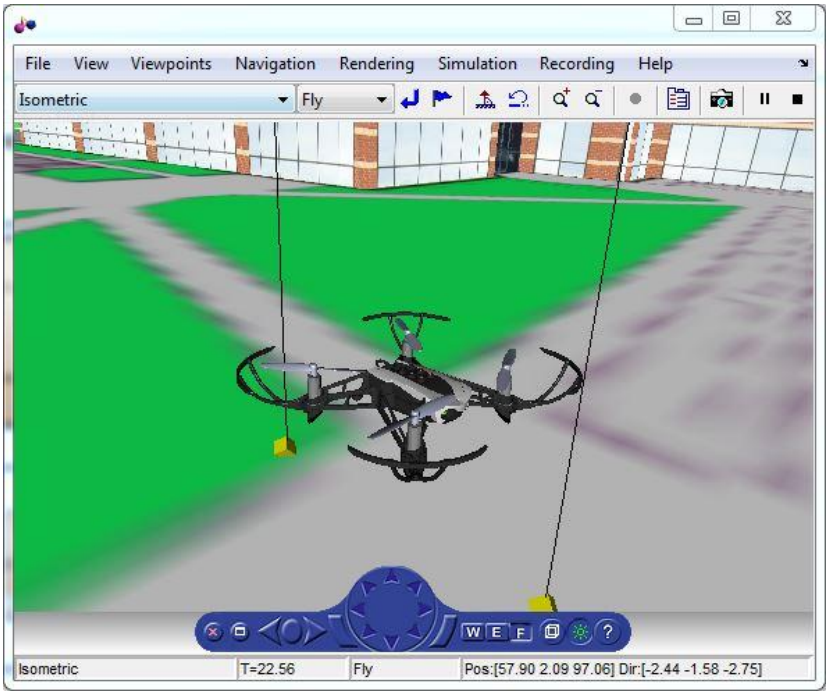


**Figura 32**


**Figura 33**

## 4.2. Il controllore FCS

All'interno del blocco "*FCS*" (*Flight Controll System*) si trova il software che il quadricottero utilizza per controllare i suoi gradi di libertà. Il "*Flight Control System*" è diviso in altri cinque sotto blocchi (Figura 34):

- "*sensordata_group*", nel quale al suo interno vengono organizzati in delle strutture i valori provenienti dai sensori
- "*estimator*", che prende in ingresso l'uscita della porta AND vista in Figura 21, i valori misurati dai sensori e le condizioni ambientali per fornire in uscita la stima degli stati attuali del quadricottero
- "*controller*", il blocco contenente i controllori veri e propri dei gradi di libertà
- "*Crash Predictor Flags*", che attiva il flag per l'arresto della simulazione se rileva degli stati definiti "non sicuri"
- "*Logging*", in cui vengono salvati i dati della simulazione, così che poi sia possibile visualizzarli ed analizzare gli eventi

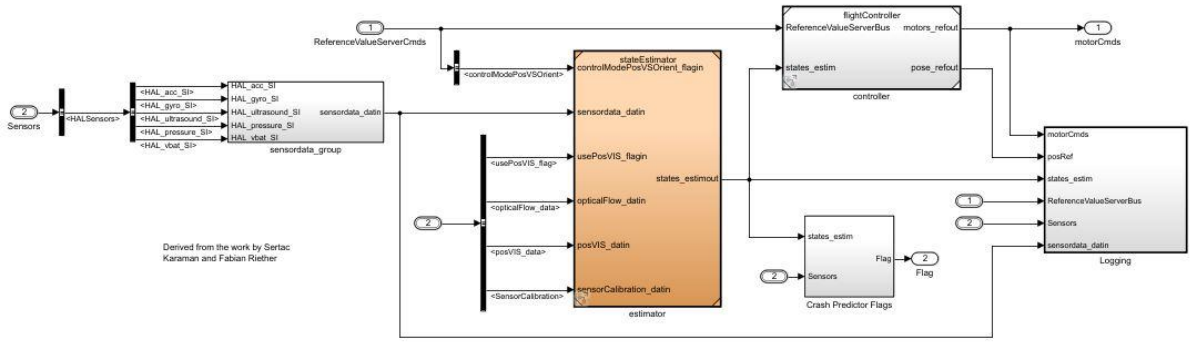


**Figura 34**

Nell'*estimator* gli stati vengono stimati attraverso i valori forniti dai sensori, infatti al suo interno (Figura 35) è presente il blocco "*SensorProcessing*" che elabora i dati dei sensori e fornisce il valore delle accelerazioni lungo i tre assi e le tre velocità angolari. Successivamente questi dati sono passati ad altri tre blocchi, uno è il "*Complementary Filter*" che calcola gli angoli di beccheggio, di imbardata e di rollio, l'altro è "*EstimatorXYPosition*" che fornisce posizione e velocità sugli assi x ed y, mentre l'ultimo è "*EstimatorAltitude*", che si occupa del calcolo dell'altitudine e la sua velocità.

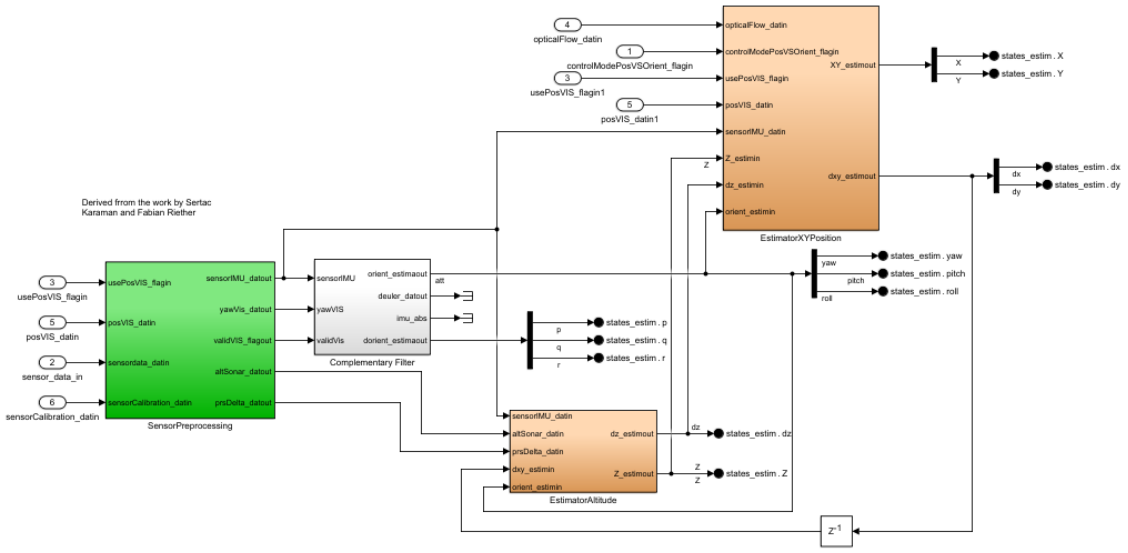


**Figura 35**

All'interno " *Crash Predictor Flags* " (Figura 36), nel caso in cui il drone dovesse finire al di fuori della sua Geo-fence, ossia la sua porzione di spazio consentita, viene impostato il valore del flag a 1, oppure, se il drone assume velocità di spostamento eccessive il flag viene posto a 99. Se non avviene nessuna delle due anomalie il flag resta a 0.

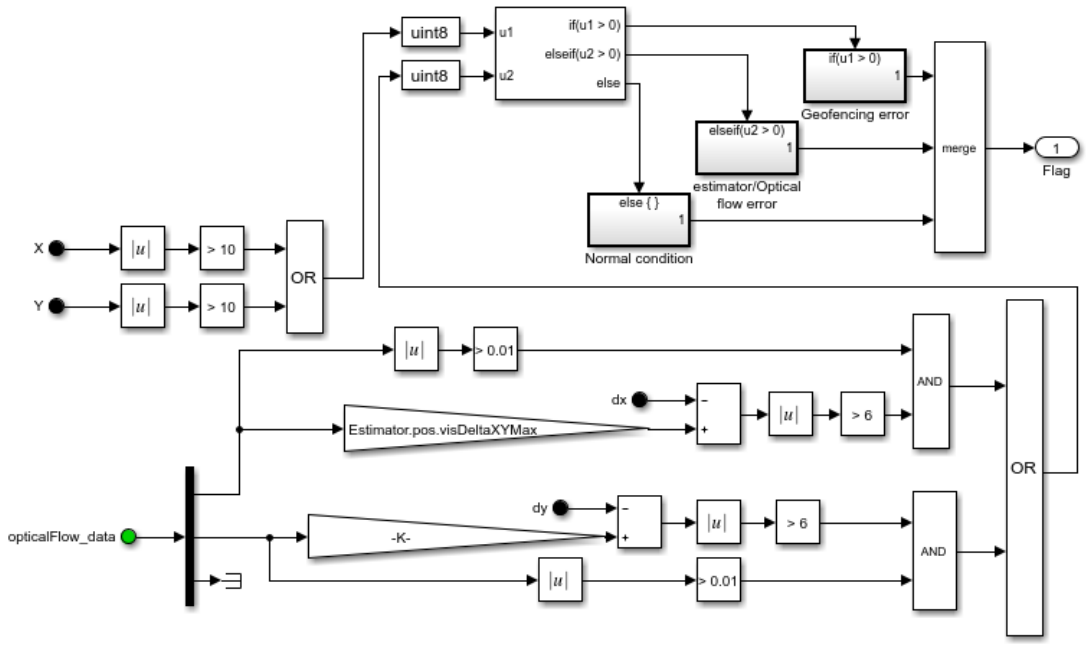


**Figura 36**

All'interno del blocco "*Logging*" sono raccolti i valori assunti dagli ingressi e dalle uscite del blocco che lo contengono ("*FCS*"), per poter visualizzare, durante o dopo la simulazione, i dati di interesse. Al suo interno sono stati posizionati degli scope per visualizzare i sei gradi di libertà, l'ingresso del blocco "*Attitude*" e il comando dei motori (Figura 37).

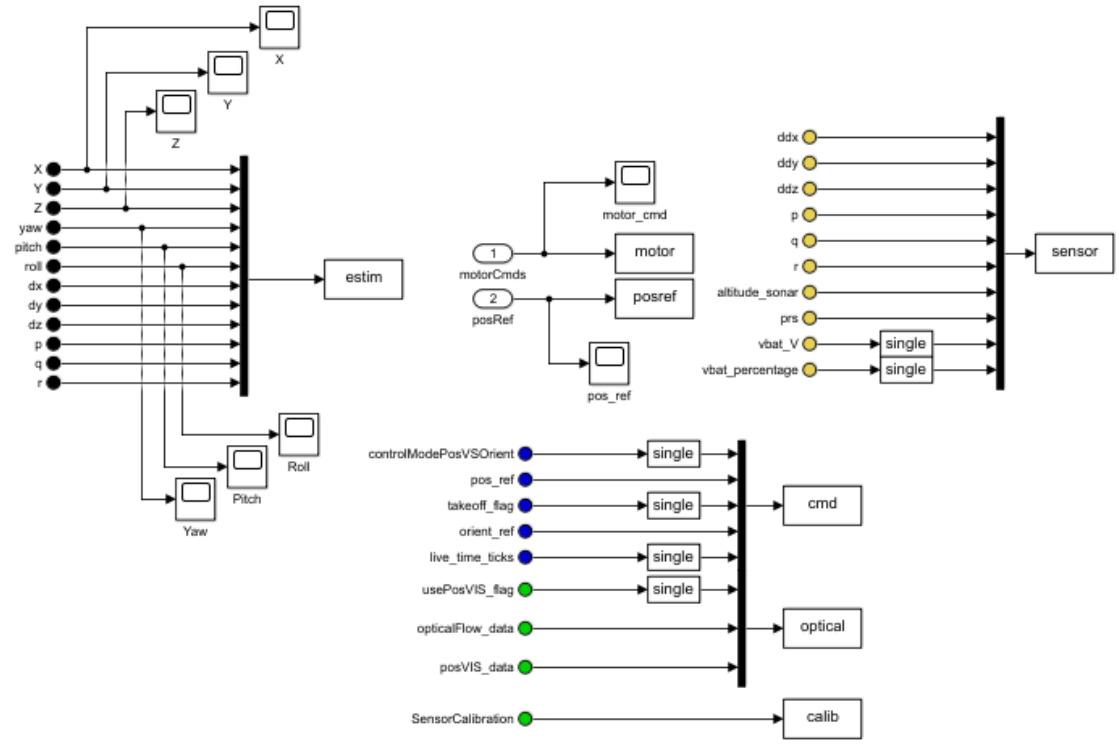


**Figura 37**

Nel blocco "*controller*", come già specificato, sono contenuti i vari controllori per i diversi gradi di libertà del quadricottero, difatti è su questo blocco che si andrà a progettare un nuovo controllore. L'interno di "*controller*" si presenta come in Figura 38 e i suoi vari blocchi sono esaminate nelle successive sottosezioni.

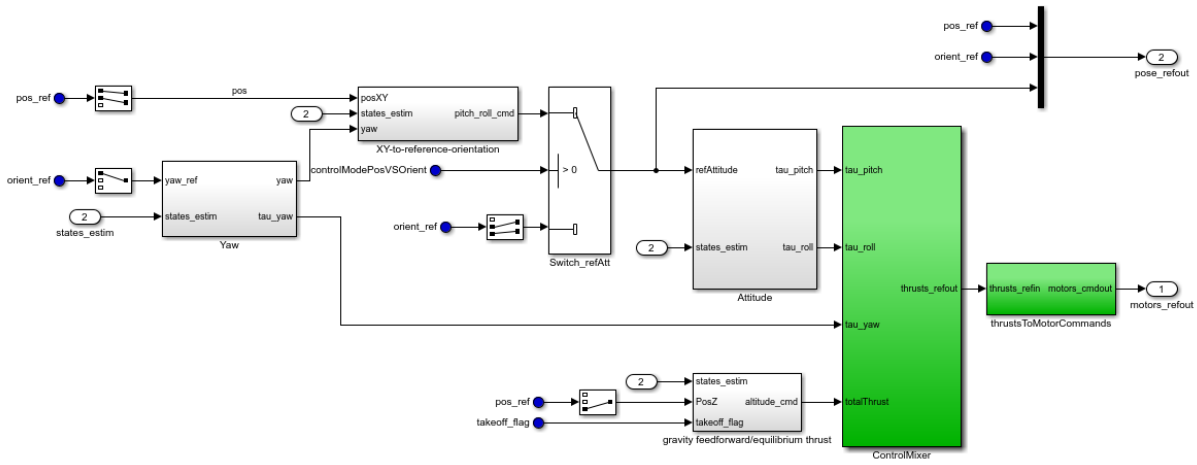


**Figura 38**

### 4.2.1. Il controllore dell'imbardata

Il controllore dell'angolo d'imbardata è rappresentato dal blocco "*Yaw*". In ingresso sono collegati il valore di riferimento per l'imbardata proveniente dal blocco "*Command*" e l'uscita del blocco "*estimator*". Al suo interno (Figura 39) il controllore seleziona degli stati stimati solamente l'angolo di imbardata *yaw* e la sua velocità *r*, poi con un regolatore proporzionale derivativo fornisce in uscita il contributo dei motori per l'angolo in questione. Il valore stimato *yaw* invece è subito portato in uscita, poiché servirà per i controllori successivi.

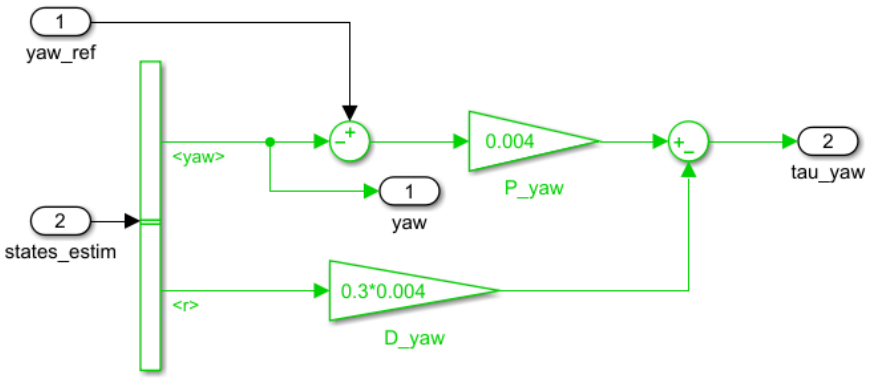


**Figura 39**

### 4.2.2. Il controllore dell'altezza

L'altezza da terra è controllata dal blocco "*gravity feedforward/equilibrium thrust*", questo blocco al suo interno (Figura 40) presenta uno switch che divide in due casi il funzionamento. Per separare i due casi di funzionamento viene utilizzata la variabile *takeoff_flag*, generata come già visto in 4.1.1. Generazione dei comandi, all'interno del blocco "*Command*". Fino a che il decollo non è terminato l'uscita del controllore sarà determinata da un guadagno statico, che garantisce la salita in breve tempo. Mentre una volta terminato il decollo a lavorare sarà un regolatore proporzionale derivativo. Questo regolatore, similarmente al caso dell'imbardata, prende in ingresso l'altitudine di riferimento, quella stimata e la velocità verticale stimata. In output è fornito il contributo alla spinta complessiva che i quattro motori assieme devono fornire. Il guadagno statico garantisce, rispetto al controllore, una spinta costante per un periodo di tempo limitato.

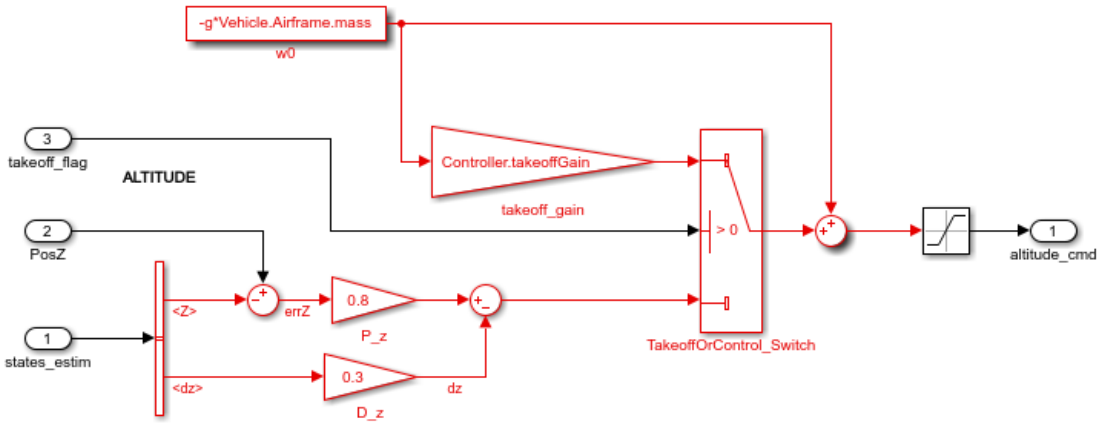


**Figura 40**

## 4.2.3. Il controllore del piano orizzontale

All'interno del blocco "*XY-to-reference-orientation*", in Figura 41, è presente il controllore della posizione sul piano orizzontale. Questo blocco necessita in input anche dell'angolo di imbardata, oltre ai valori stimati della posizione in x e y, delle loro velocità stimate e dei loro valori di riferimento. L'angolo di imbardata è utile ad interfacciare il sistema di rifermento inerziale con quello solidale al drone. Infatti essendo gli spostamenti orizzontali del drone comandati tramite la variazione degli angoli di beccheggio e di rollio, nel caso che l'imbardata non sia nulla, si deve comprendere come variare l'assetto del drone nelle due componenti al fine di compiere lo spostamento desiderato.

Viene quindi aggiunto al controllore proporzionale derivativo, con funzionamento analogo a quelli visti in precedenza, una moltiplicazione matricale sul ramo proporzionale in grado di esprimere lo scostamento dalla posizione desiderata nel s.d.r. B. La moltiplicazione sfrutta appunto il valore dell'angolo di imbardata per effettuare la conversione.

L'uscita prodotta da questo controllore sono i valori che dovranno poi assumere gli angoli di rollio e beccheggio per mantenere o raggiungere la posizione di riferimento.

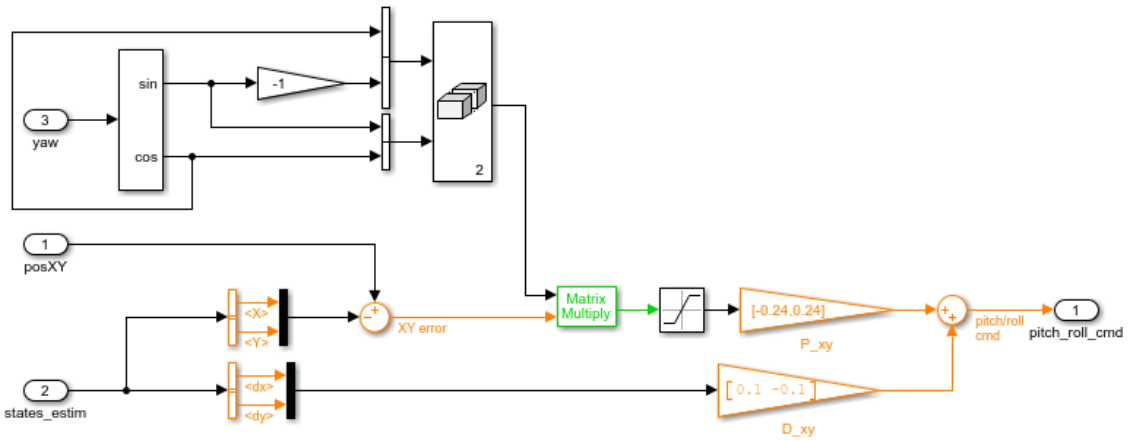


**Figura 41**

## 4.2.4. Il controllore dell'assetto

Il controllore dell'assetto (in inglese "*Attitude*"), ossia degli angoli di beccheggio e di rollio è formato da un PID (Figura 42). Il controllore anche in questo caso sfrutta i valori stimati di velocità e posizioni angolari dei due gradi di libertà interessati e i loro valori di riferimento. I valori di riferimento però, come si può vedere in Figura 38 possono giungere da due differenti sorgenti e di questo si approfondisce in 4.2.5. Lo switch. Il PID è costruito come un'unica struttura per i due angoli, ma la sola parte ad essere uguale per entrambi è la parte integrativa, infatti la parte proporzionale e quella derivativa hanno guadagni differenti. Infine sono forniti in uscita i contributi di spinta dei motori per ciascuno dei due angoli.

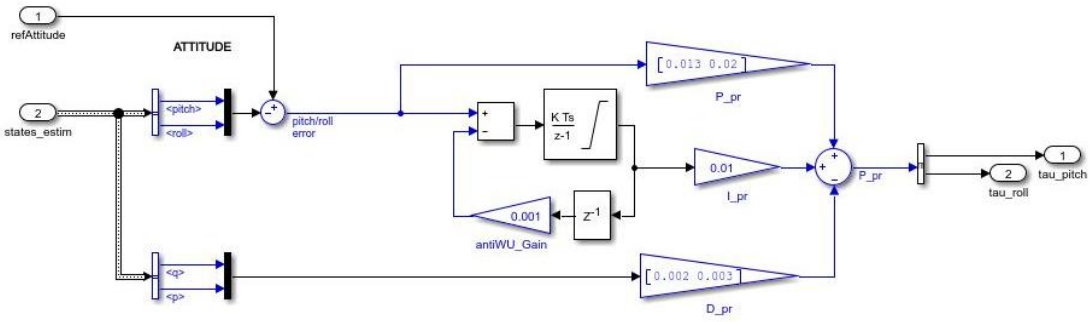


**Figura 42**

### 4.2.5. Lo switch

Lo switch visibile nel diagramma in Figura 38 ha il compito di selezionare quali valori di riferimento dovrà avere in ingresso il blocco "*Attitude*". La sua scelta si basa sulla variabile *controlModePosVSOrient*, generata nel blocco "*Command*", come visto in 4.1.1. Generazione dei comandi. Questa selezione è necessaria, in quanto se sono preseti comandi sugli angoli di assetto, il drone non può riceverne altri sulla posizione, altrimenti i due comandi andrebbero ad intaccarsi a vicenda. Per questo, se vi sono comandi sull'assetto, quelli di posizione vengono ignorati (per opera della porta AND in Figura 21) e ad entrare nel controllore "*Attitude*" sono i valori di riferimento di beccheggio ed imbardata generati dal *Signal Editor*. Altrimenti, se non sono presenti comandi sugli angoli di beccheggio e rollio, in ingresso al blocco "*Attitude*" entrano i valori che devono assumere i due angoli per soddisfare il controllo della posizione in X-Y.

### 4.1.6. Ottenimento dei comandi ai motori

Gli ultimi due blocchi del "*controller*" sono il "*ControlMixer*" e il "*thrustsToMotorCommands*", nei quali avviene la fusione dei vari controlli e la trasformazione in un segnale di comando ai motori.

In particolare, il "*ControlMixer*", come visibile in Figura 43 moltiplica i contributi alla spinta verticale, all' imbardata, al beccheggio e al rollio per la matrice *Controller.Q2Ts*, dal valore di seguito riportato:

$$Controller.Q2Ts = \begin{bmatrix} 0{,}25 & 103{,}5736 & -5{,}6659 & -5{,}6659 \\ 0{,}25 & -103{,}5736 & -5{,}6659 & 5{,}6659 \\ 0{,}25 & 103{,}5736 & 5{,}6659 & 5{,}6659 \\ 0{,}25 & -103{,}5736 & 5{,}6659 & -5{,}6659 \end{bmatrix} \quad (35)$$

In questa matrice le colonne corrispondono al contributo che dovranno rispettivamente poi avere la propulsione verticale, l'imbardata, il beccheggio ed il rollio sulla spinta finale dei motori. Quest'ultimi sono infatti rappresentati dalle righe.

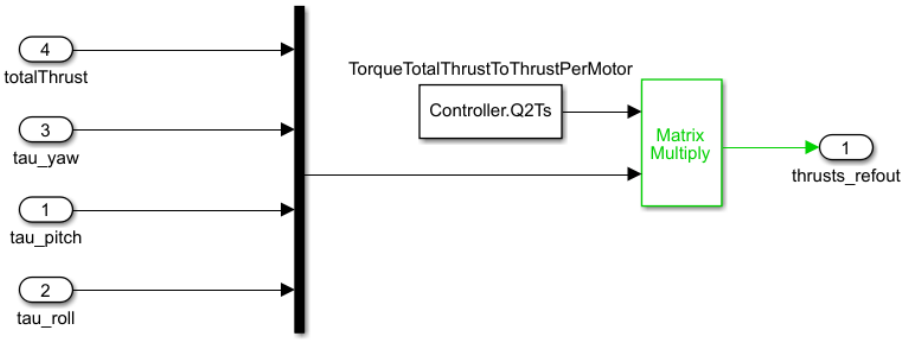


**Figura 43**

In " *thrustsToMotorCommands* " invece si effettuano due semplici moltiplicazioni per passare dalla spinta che i motori devono generare a quelli che saranno i loro comandi. In Figura 44 sono mostrate appunto due moltiplicazioni, una per la variabile *Vehicle.Motor.thrustToMotorCommand*, di valore:

$$Vehicle.Motor.thrustToMotorCommand = 1530{,}7 \qquad (36)$$

che fornisce il modulo del comando ai motori e l'altra per il vettore $[1 \quad -1 \quad 1 \quad -1]$, che determina il loro senso di rotazione.

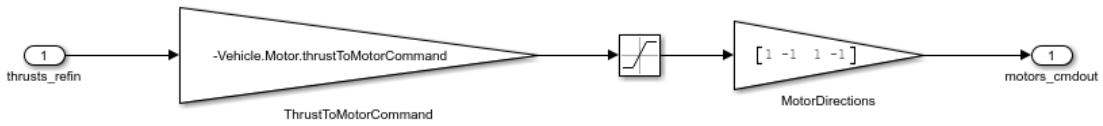


**Figura 44**

# 5. Progetto di un nuovo controllore

Si è scelto di progettare un controllore in grado di regolare il rollio, ossia l'angolo che permette al drone di spostarsi lungo il suo asse trasversale.

**Figura 45**

Originalmente il controllo è gestito dal controllore PID riportato in Figura 42 già esaminato in 4.2.4. Il controllore dell'assetto. In simulazione, questo controllore, se è selezionato il modello matematico lineare, restituisce il rollio e la posizione in Y riportate in Figura 46 e in Figura 47, oppure i risultati visibili a Figura 48 e a Figura 49 nel caso di modello matematico non lineare.

Rollio con modello lineare

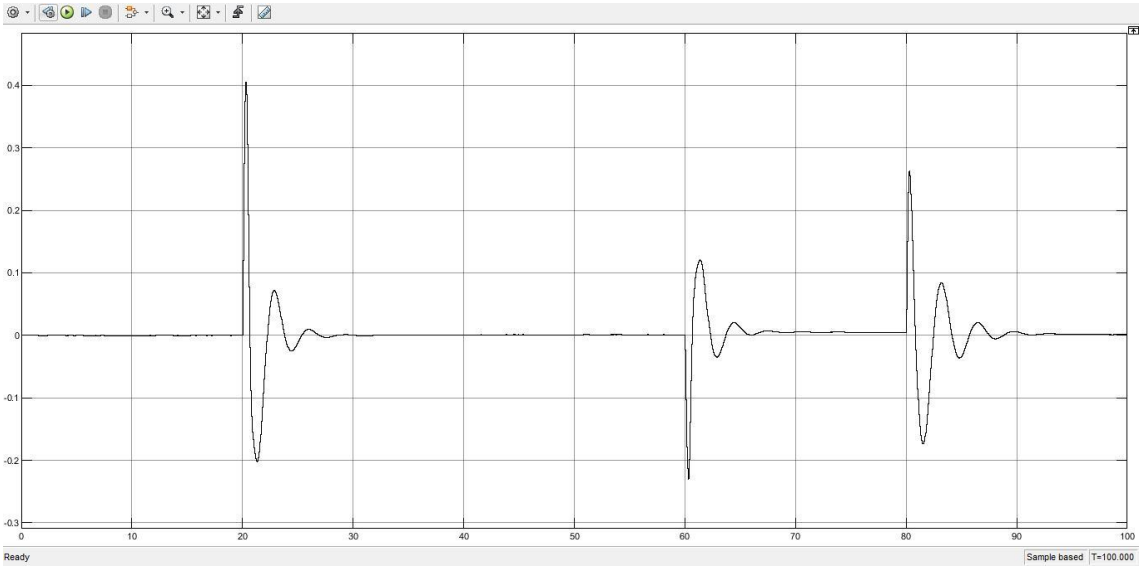


**Figura 46**

Posizione in Y con modello lineare

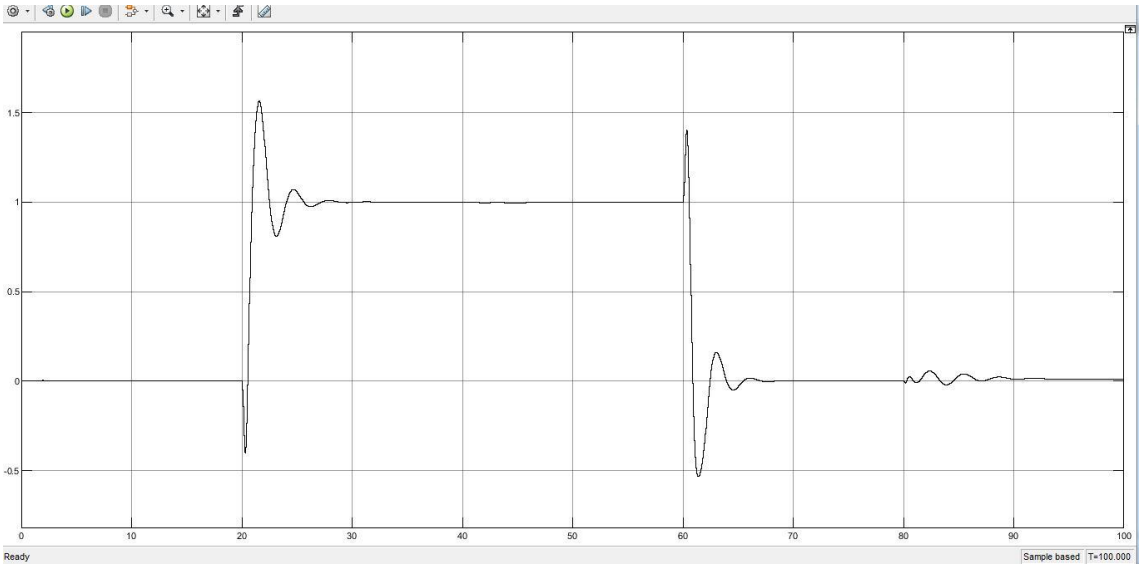


**Figura 47**

Rollio con modello non lineare

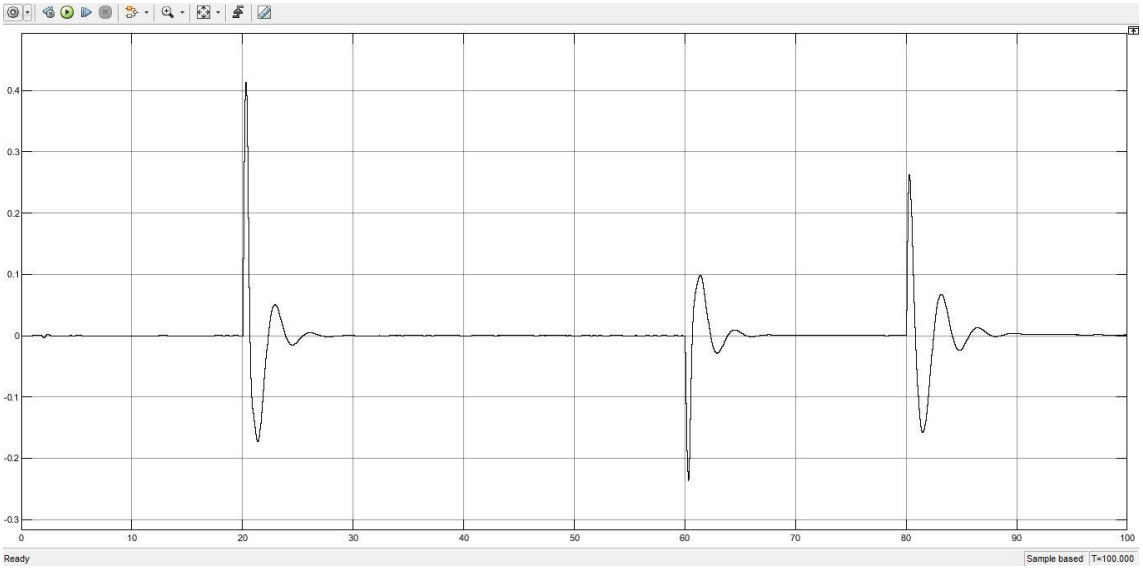


**Figura 48**

Posizione in Y con modello non lineare

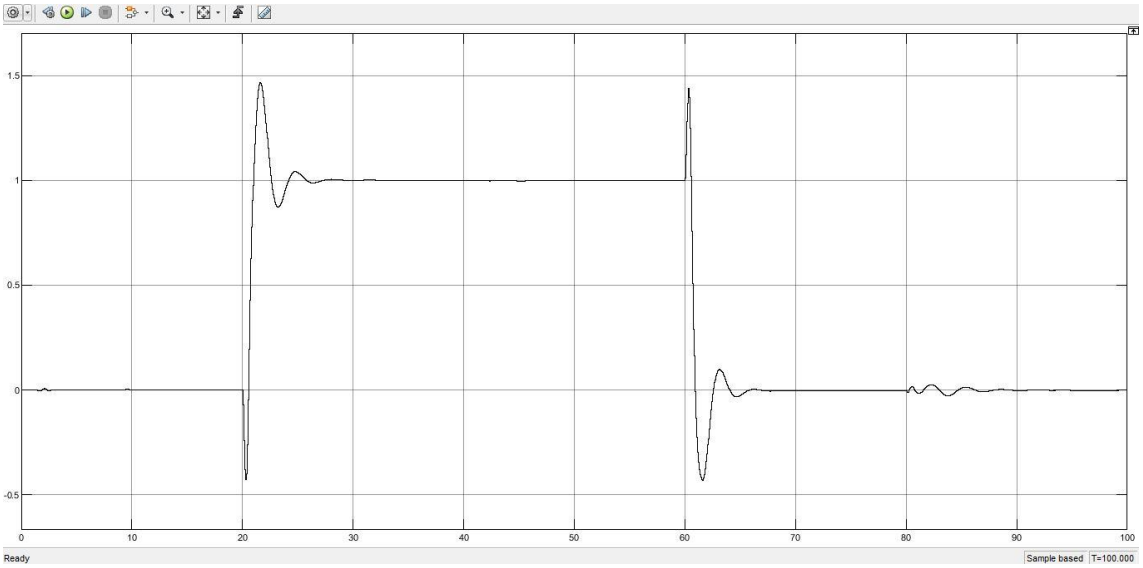


**Figura 49**

## 5.1. Il metodo della sintesi in frequenza

Per progettare un nuovo controllore si è utilizzata la tecnica della sintesi in frequenza. Questa pratica consiste nel progettare una rete correttrice in grado di offrire determinate caratteristiche a catena aperta, che corrisponderanno poi a delle desiderate specifiche in catena chiusa. Una rete correttrice è solitamente composta da un guadagno statico, una rete anticipatrice ed una rete ritardatrice.

Una rete anticipatrice è una funzione di trasferimento con numeratore e denominatore entrambi di primo grado. Questa rete deve il suo nome al fatto che lo zero ha una pulsazione minore di quella del polo, il ché si traduce in una fase dal segno positivo. Viceversa, la rete ritardatrice avrà il polo con pulsazione minore di quella dello zero e di conseguenza la sua fase avrà segno negativo. Le due reti sono descritte dalle espressioni seguenti:

$$R_a(s) = \frac{1 + \tau_a s}{1 + \frac{\tau_a}{m_a} s} \tag{37}$$

$$R_r(s) = \frac{1 + \frac{\tau_r}{m_r} s}{1 + \tau_r s} \tag{38}$$

I valori $\tau_a, \tau_r, m_a$ e $m_r$ sono calcolati durante la fase di progettazione attraverso i grafici caratteristici delle due reti. In Figura 50 è riportato il diagramma della rete anticipatrice, se per esempio si volesse ottenere, ad una certa frequenza $\omega$, dei determinati innalzamenti di modulo e di fase, si identifica il punto che soddisfa le due condizioni e si trovano i corrispondenti $m_a$ ed $\omega\tau_a$. A questo punto, sapendo che $\omega$ è la frequenza desiderata si può procedere alla costruzione della rete. Per la rete ritardatrice il ragionamento è analogo, ma il diagramma è specchiato rispetto all'asse orizzontale.

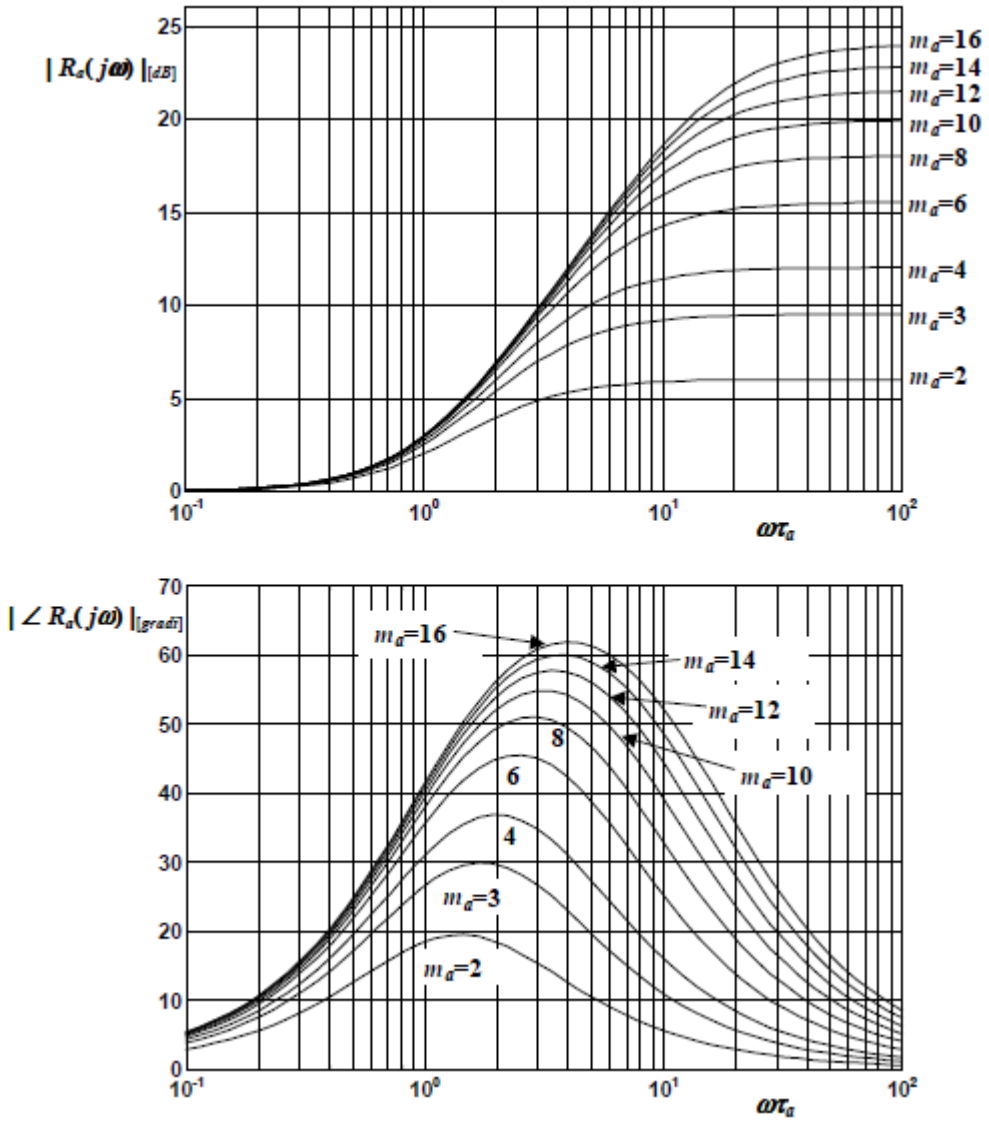


**Figura 50**

## 5.2. Lo strumento di Matlab SISOTool

In Matlab è presente un'applicazione di progettazione grafica per controllori chiamata "SISOTool". Per avviarla basta digitare '*sisotool*' nella Command Window, dopodiché al suo interno, potrà essere modellata l'architettura della rete con la quale si desidera lavorare e caricare le funzioni di trasferimento presenti nel Work Space. Questa applicazione consente di

visualizzare diversi tipi di grafici selezionandone anche la zona del diagramma sulla quale devono lavorare.

Si può iniziare caricando la funzione di trasferimento del sottosistema dell'angolo di rollio nell'applicazione. Prima di fare ciò si ricavano le 4 sottomatrici dalla variabile *linsys*, vista in Figura 19. Delle quattro matrici A, B, C e D andranno considerati solamente gli stati denominati *phi* (angolo di rollio) e *p* (velocità di rollio), gli ingressi *Actuators(1÷4)* (i quattro motori) e l'uscita *Euler(1)* (l'angolo di rollio uscente). Le matrici ottenute saranno dunque:

$$A = \begin{bmatrix} 0 & 1 \\ 0 & -2{,}1715 \end{bmatrix} \qquad B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0{,}4202 & 0{,}4202 & -0{,}4202 & -0{,}4202 \end{bmatrix}$$

$$C = [1 \quad 0] \qquad D = [0 \quad 0 \quad 0 \quad 0] \tag{39}$$

delle quali, B e D possono essere semplificate ad una sola colonna.

Ottenute queste matrici si costruisce lo spazio di stato con l'istruzione *ss(A,B,C,D)* di Matlab e passando il risultato alla funzione *tf()* se ne calcola la funzione di trasferimento.

La G(s) finale la si trova moltiplicando per le ultime due costanti che si trovano in serie al controllo del rollio, ovvero l'ultima colonna della matrice in **(35)** e la costante in **(36)**. Si ottiene:

$$G(s) = \frac{3644}{s^2 + 2{,}152s} \tag{40}$$

Questa funzione può essere caricata nel blocco *'G'* del SISOTool in Figura 51 e selezionando il diagramma di Bode a catena aperta e la risposta al gradino a catena chiusa si visualizzano i grafici in Figura 52.

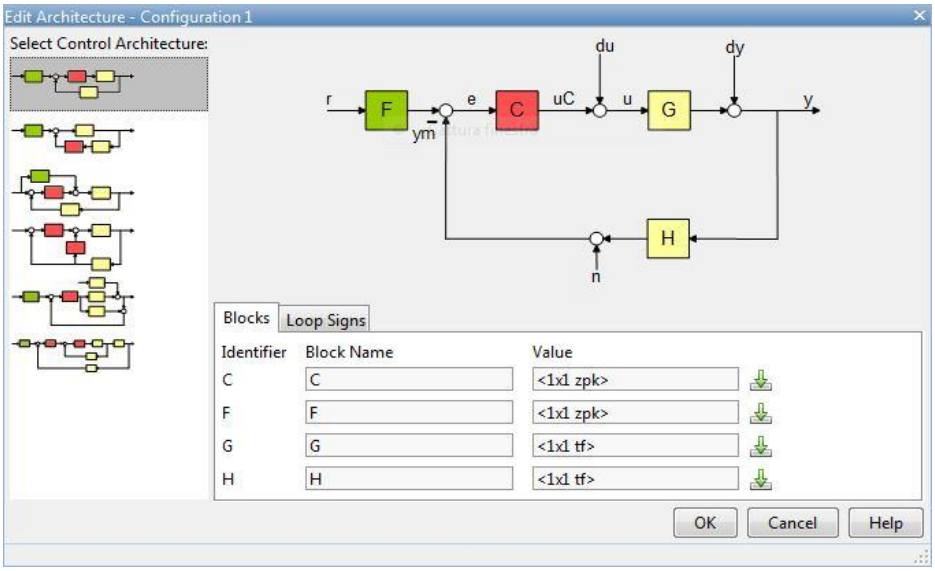


**Figura 51**

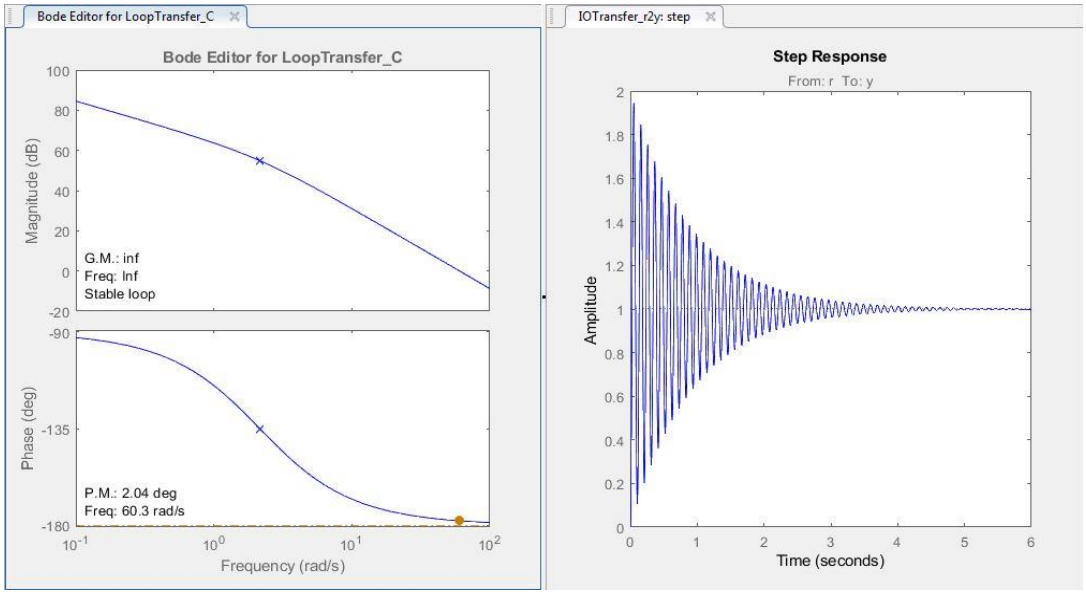


**Figura 52**

## 5.3. Specifiche del controllore

Del controllore si devono fissare quali saranno le specifiche che dovrà soddisfare in catena chiusa. Le specifiche scelte sono:

1. Sistema di tipo 1

2. Errore a regime $e \leq 0{,}01$
3. $M_r \leq 2dB$
4. $B_3 = 2Hz$

Le prime due sono dette univoche e corrispondono a degli aspetti che il sistema dovrà soddisfare a regime, ossia avere un errore nullo per un ingresso a gradino e minore o uguale all'1% per un ingresso a rampa. Le altre due sono dette lasche e riguardano il transitorio, in particolare garantiscono delle limitazioni rispettivamente per: la sovraelongazione e il tempo di salita.

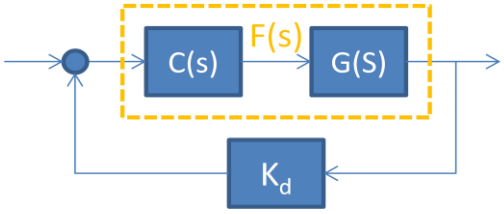


**Figura 53**

Facendo riferimento alla Figura 53 le specifiche passate in catena aperta invece risultano:

1. F(s) con un polo in s=0
2. Errore a regime $e = \frac{K_d^2}{K_f} \leq 0{,}01$
3. $M_\phi \geq 42°$
4. $\omega_t = 10\, rad/sec$

La terza è ottenuta tramite l'utilizzo della carta di Nichols, mentre la quarta tramite la relazione $[\omega_t]rad/sec \cong 3 \div 5[B_3]Hz$.

## 5.4. Primo approccio con rete anticipatrice

Passando al soddisfacimento delle specifiche elencate precedentemente si nota che la 1 è già soddisfatta, in quanto è già presente un polo in s=0 in G(s), la 2 invece risulta essere:

$$K_d = 1 \rightarrow e = \frac{1}{K_f} = \frac{1}{K_c K_g} \rightarrow K_c = \frac{K_f}{K_g} = \frac{1}{eK_g} = \frac{1}{0.01 * 3644} \geq 0{,}0274 \qquad (41)$$

Per cui si prende un guadagno statico C1=0,0274.

Soddisfatte le specifiche univoche si passa a quelle lasche, per soddisfarle si devono osservare il margine di fase e la frequenza di attraversamento del diagramma di Bode di C1*G(s). Si carica il guadagno statico C1 nel blocco *'C'* di SISOTool e si osserva in Figura 54 che il margine di fase è di 12,3° e la frequenza di attraversamento di 9,88 radianti al secondo. Inoltre sono presenti notevoli oscillazioni e una elevata sovraelongazione.

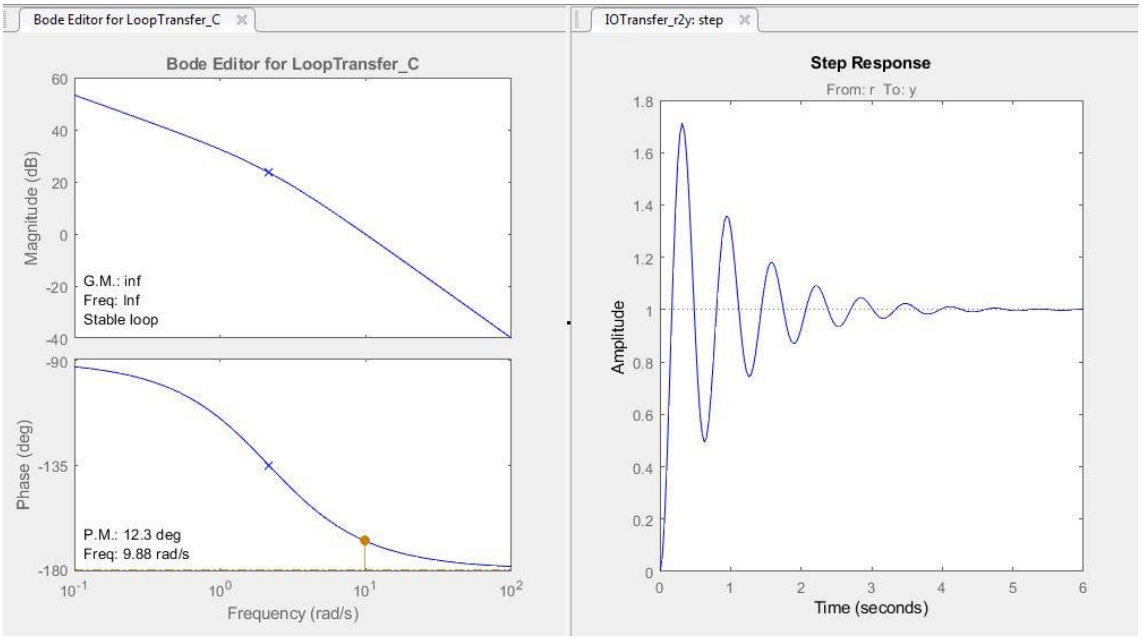


**Figura 54**

## 5.4.1. Progettazione della rete

Come si è visto in Figura 54, il margine di fase deve essere alzato di almeno $42° - 12{,}3° \cong 30°$, mentre il modulo può rimanere invariato, in quanto la frequenza di taglio si discosta poco dal valore desiderato.

Si progetta dunque una rete anticipatrice, cercando nei grafici a Figura 50, un punto per una frequenza $\omega = 10\frac{rad}{sec}$ che alzi la fase di almeno 30° e che non sia troppo incisivo sul modulo. I valori corrispondenti sono $m_a = 8$ e $\omega\tau_a = 0{,}7 \rightarrow \tau_a = 0{,}07$, ai quali corrisponde la rete:

$$R_a(s) = \frac{1 + 0.07s}{1 + \frac{0.07}{8}s} \tag{42}$$

A questo punto il blocco *'C'* viene aggiornato con il controllore C2(s):

$$C2(s) = C1 * R_a(s) = 0{,}0274 * \frac{1 + 0.07s}{1 + \frac{0.07}{8}s} \tag{43}$$

Quello che si osserva dal SISOTool è riportato a Figura 55 ed è visibile che rispetto al caso precedente, con solamente il guadagno statico, si ha un margine di fase maggiore dei 42° richiesti, una frequenza di attraversamento che non si discosta molto dalla desiderata e sia le oscillazioni che la sovraelongazione sono diminuite.

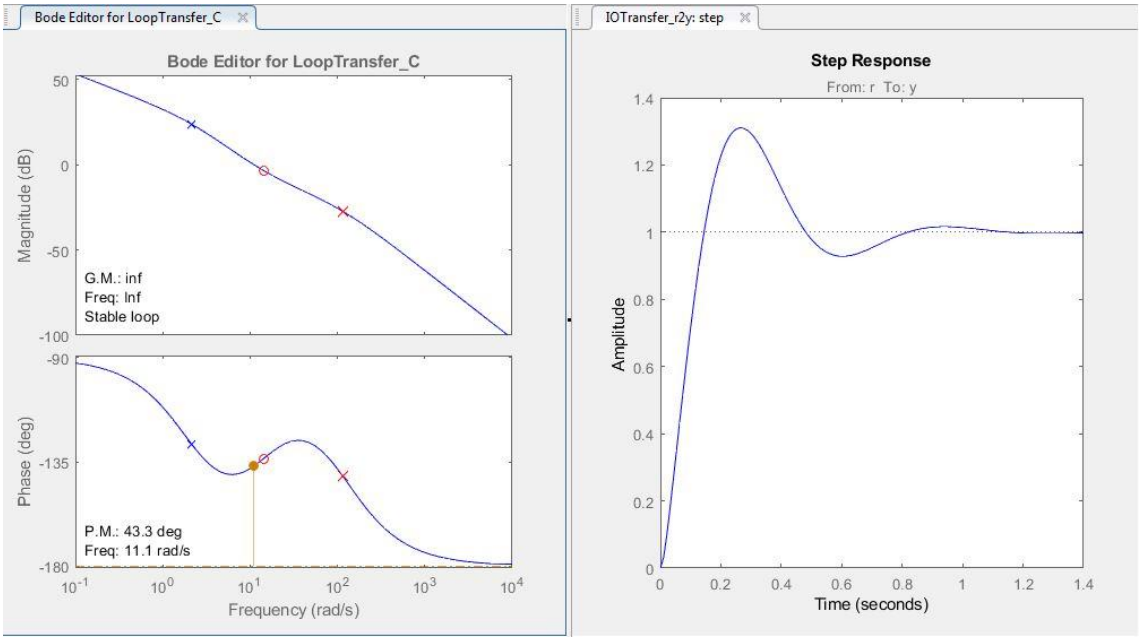


**Figura 55**

Per implementare questo controllore su Simulink si deve modificare il PID che agiva precedentemente sul rollio visibile in Figura 42. Si lascia per il beccheggio il controllore PID originale e lo si mette in un blocco a parte per snellire la struttura, mentre per il rollio se ne prende l'errore e lo si collega ad un blocco zero-poli nel quale andranno immessi il vettore degli zeri, il vettore dei poli e il guadagno statico del controllore C2(s). Al fine di inserire il nuovo

controllore nel blocco zero-poli, si dovrà prima esprimerlo in forma poli-zeri tramite la funzione *zpk(C2s)*, ottenendo il risultato seguente:

$$zpk(C2s) = 0{,}2192\frac{(s + 14{,}2857)}{(s + 114{,}2857)} \tag{44}$$

In Figura 56 è riportato lo schema che si ottiene al termine della modifica, mentre in Figura 57 è riportato l'inserimento del controllore all'interno del blocco zero-poli.

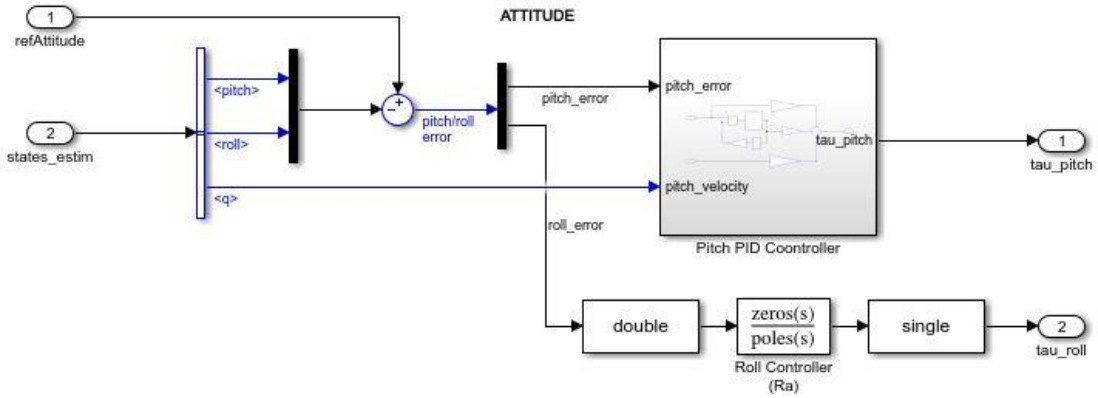


**Figura 56**

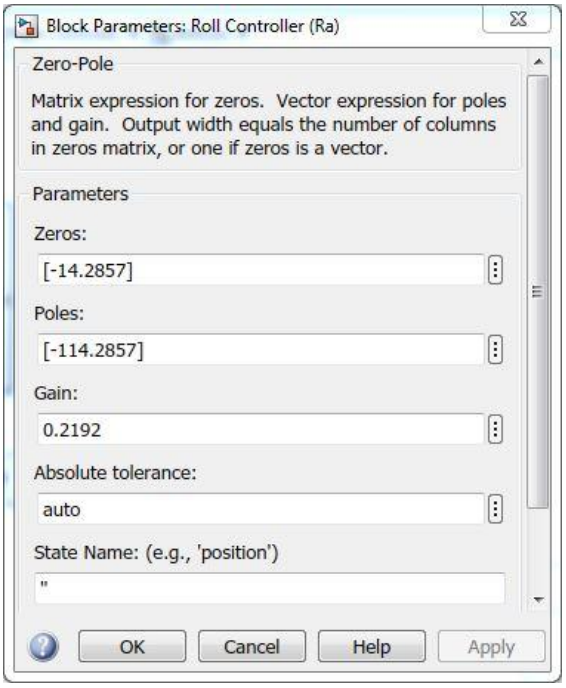


**Figura 57**

## 5.4.2. Costruzione di un by-pass

Prima di procedere con la simulazione, ci si vuole accertare che il controllore appena costruito sia privo di errori e che sia almeno in grado di stabilizzare a zero l'assetto del rollio. Per far questo si effettua un by-pass forzato sul controllo della posizione X-Y, cosicché il controllore "*Attitude*" non prenda come ingresso di riferimento l'uscita del controllore della posizione X-Y. La costruzione del by-pass prevede l'applicazione di un selettore, collegato al valore di una costante, che ha la possibilità di forzare a zero l'uscita della porta AND del blocco "*Signal Editor*". In Figura 58 è riportata la modifica effettuata nel blocco di generazione dei segnali, nella quale ora è presente il selettore che attiva e disattiva il by-pass.

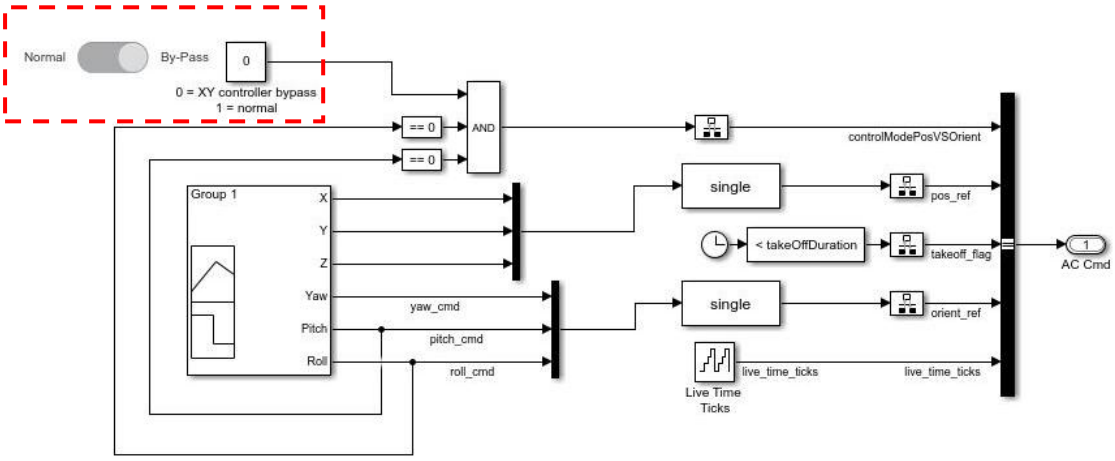


**Figura 58**

## 5.4.3. Simulazioni

Una volta attivato il by-pass si iniziano ad effettuare le simulazioni, sia per il caso di modello matematico lineare che non lineare, andando a studiare l'andamento del rollio e della posizione in Y dagli oscilloscopi del blocco "*Logging*". In Figura 59 e Figura 60 sono riportati rispettivamente il grafico del rollio e della posizione in Y nel caso lineare, mentre in Figura 61 e Figura 62 sono mostrati nel caso di modello non lineare. È chiaro che il controllore è in grado di mantenere l'assetto del drone per entrambi i modelli, infatti le differenze nel rollio sono contenute nell'ordine del $10^{-4}$, perciò trascurabili. Le differenze sulla Y non sono da considerare, dato che il controllore sulla posizione è stato by-passato.

Rollio con modello lineare

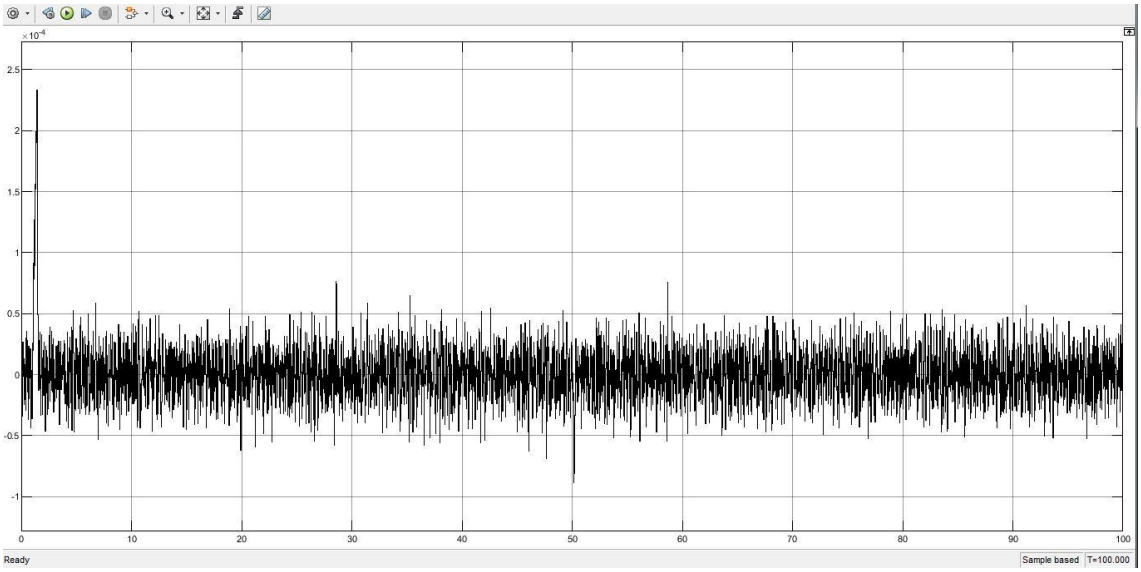


**Figura 59**

Posizione in Y con modello lineare

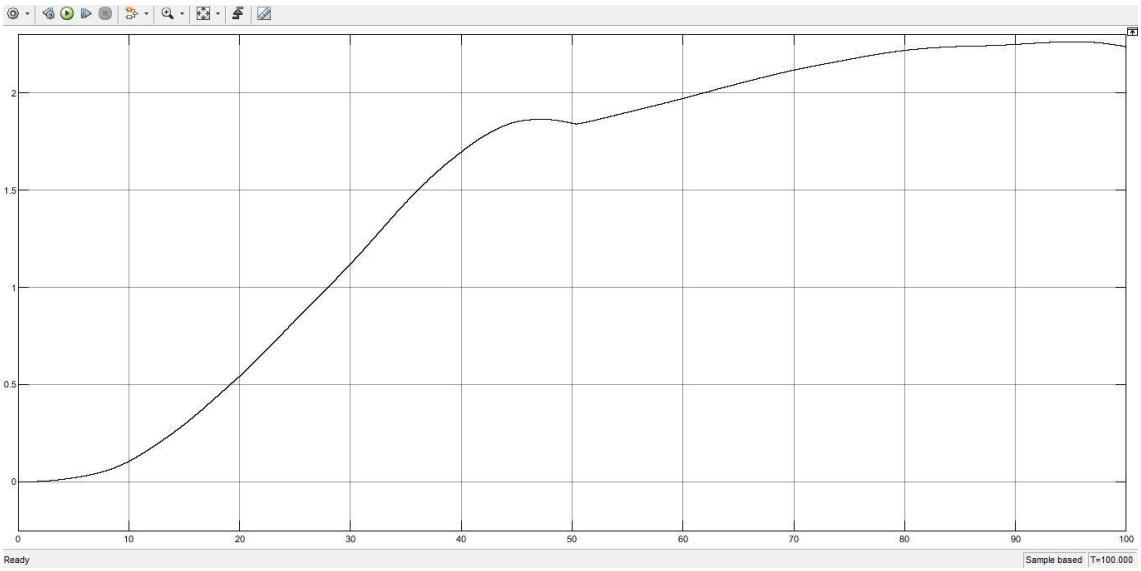


**Figura 60**

Rollio con modello non lineare

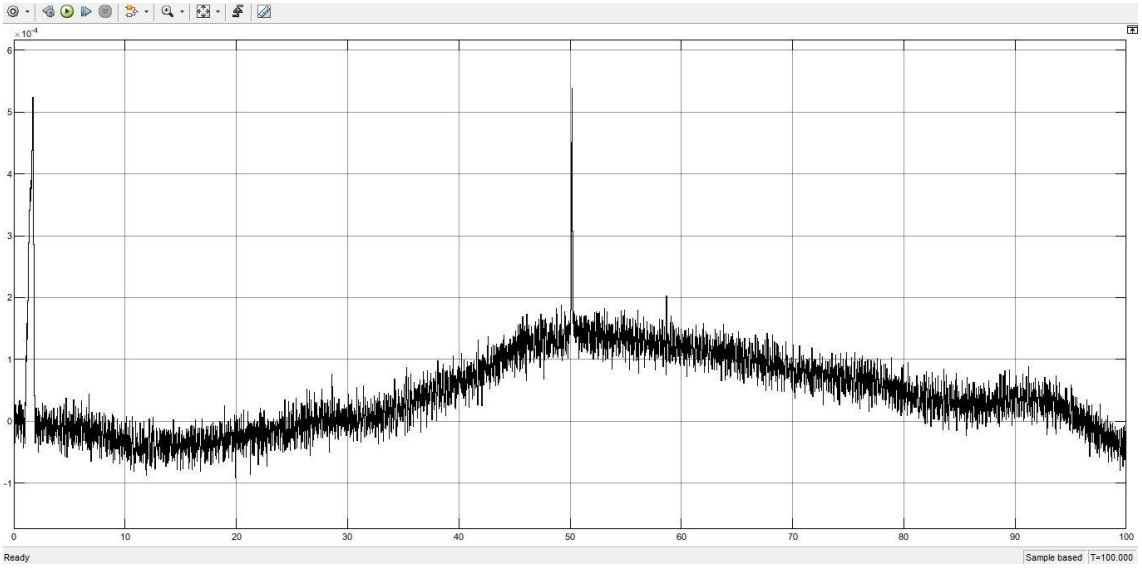


**Figura 61**

Posizione in Y con modello non lineare

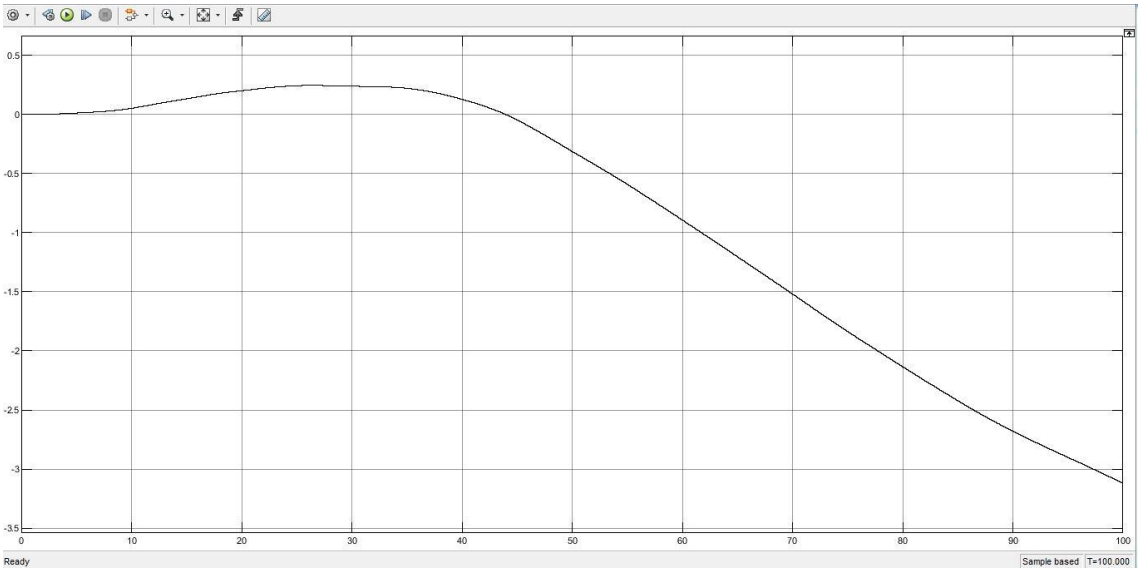


**Figura 62**

Una volta osservato che il controllore agisce bene in caso di ingresso costante a zero, si disattiva il by-pass e si avvia la simulazione. Dopo pochi secondi che la simulazione è stata avviata, un flag ne provocherà il suo arresto (Figura 63). Questo arresto è dovuto dal fatto che il controllore C2(s) non riesce a soddisfare le richieste inviate dal controllore di posizione X-Y, perciò occorre progettarne uno più efficiente, che abbia:

- minor sovraelongazione → occorre aumentare il margine di fase
- ridotte oscillazioni → occorre ridurre la sovraelongazione, ma anche aumentare il tempo di salita → per aumentare il tempo di salita si deve diminuire la frequenza di attraversamento

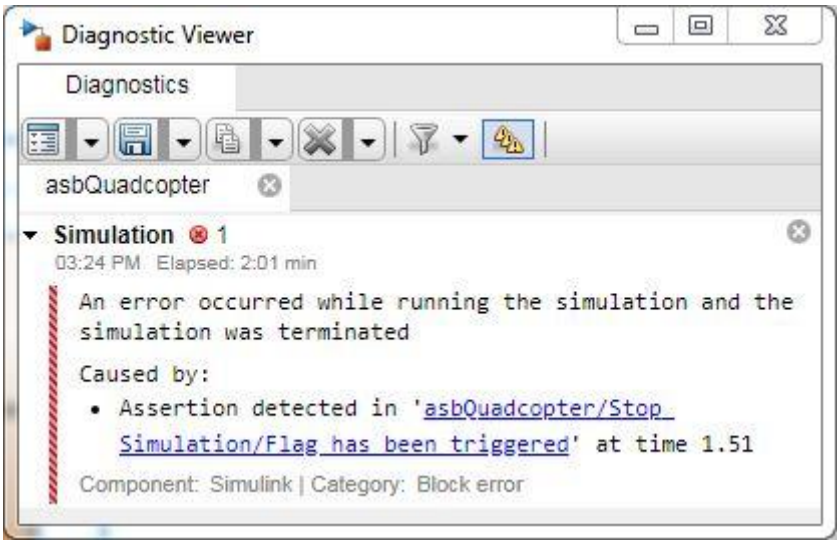


**Figura 63**

## 5.5. Costruzione di una rete correttrice completa

Il successivo controllore sarà costituito, come il precedente, da un guadagno e da una rete anticipatrice, ma avrà in aggiunta anche una rete ritardatrice. Il guadagno resta C1, mentre la rete anticipatrice viene scelta con lo scopo di alzare di molto la fase, senza però condizioni sul modulo, che sarà aggiustato poi dalla ritardatrice. Per la nuova anticipatrice $R_{a1}$ vengono quindi scelti un polo in $s = -40$ e uno zero in $s = -4$.

Dato che si vorrebbe diminuire la frequenza di attraversamento a circa 1 radiante al secondo e il modulo del diagramma di Bode della funzione $C1 * R_{a1}(s) * G(s)$ vale 11dB per $\omega = 1$, la rete ritardatrice dovrà abbassare il modulo di 11dB in $\omega = 1$.

Dai grafici in Figura 50, invertiti di segno, si ottengono: $m_r = 3{,}5$ e $\omega\tau_r = 30 \rightarrow \tau_r = 30$, che forniscono una rete ritardatrice:

$$R_{r1}(s) = \frac{1 + \frac{30}{3.5}s}{1 + 30s} \tag{45}$$

Il controllore complessivo espresso in forma poli e zeri risulta:

$$C3(s) = C1 * R_{a1}(s) * R_{r1}(s) = 0.0078 * \frac{(s + 0.1167)(s + 4)}{(s + 0.0333)(s + 40)} \tag{46}$$

Caricando C3(s) nel blocco *'C'* si osservano i grafici di Bode e la risposta al gradino in Figura 64. Si notano una diminuzione della sovraelongazione, dovuta al notevole aumento del margine di fase e l'assenza delle oscillazioni, collegata alla diminuzione della frequenza di attraversamento.

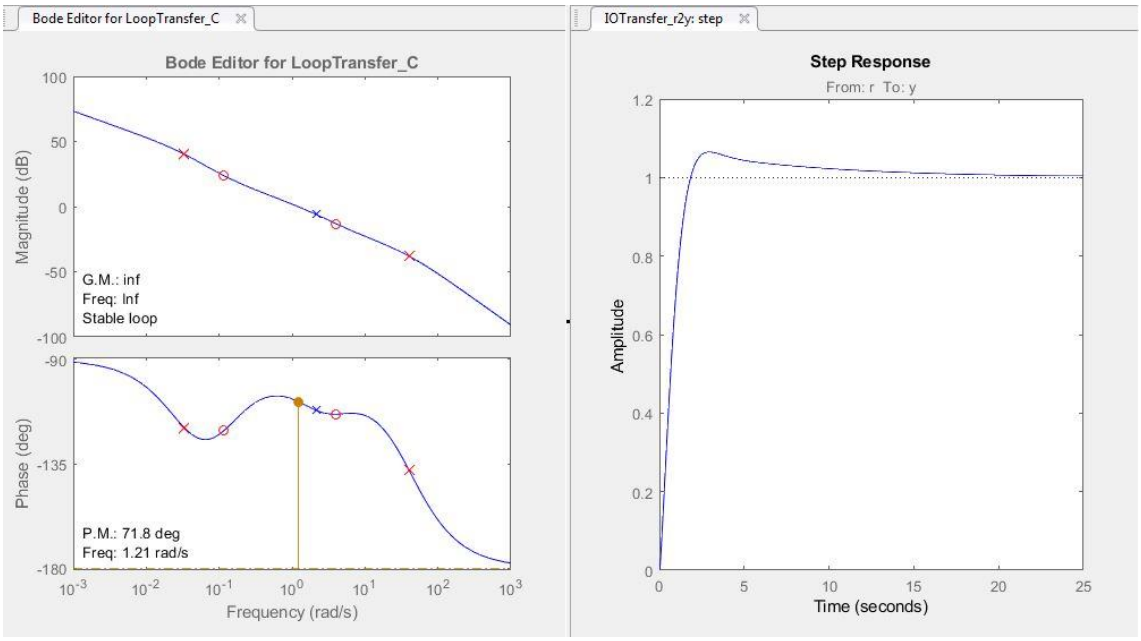


**Figura 64**

### 5.5.1. Simulazioni

Il controllore C3(s) viene caricato nel blocco zero-poli (Figura 65) e si avviano le simulazioni per i due diversi modelli.

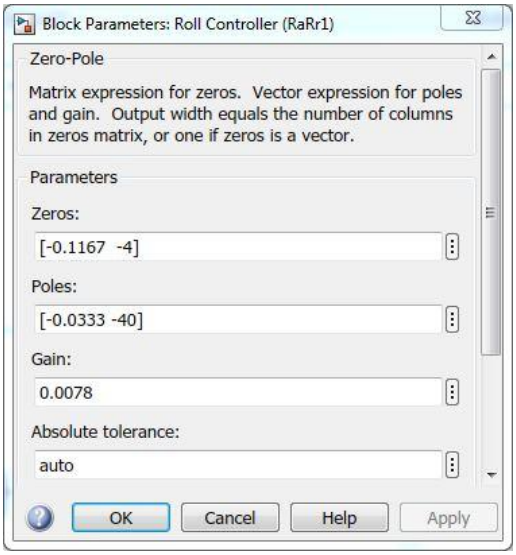


**Figura 65**

Nelle immagini a Figura 66 e a Figura 67 sono riportati i grafici del rollio e della posizione in Y nel caso di modello matematico lineare, mentre in Figura 68 e in Figura 69 i medesimi grafici, ma per il modello non lineare. In questo caso la differenza che risalta fra i due risultati è che nel caso di modello lineare sono presenti maggiori oscillazioni e sovraelongazioni rispetto al caso di modello non lineare. Per entrambi i risultati però questo risultato non è accettabile, in quanto sono presenti troppe oscillazioni e le sovraelongazioni sono eccessive.

Rollio con modello lineare

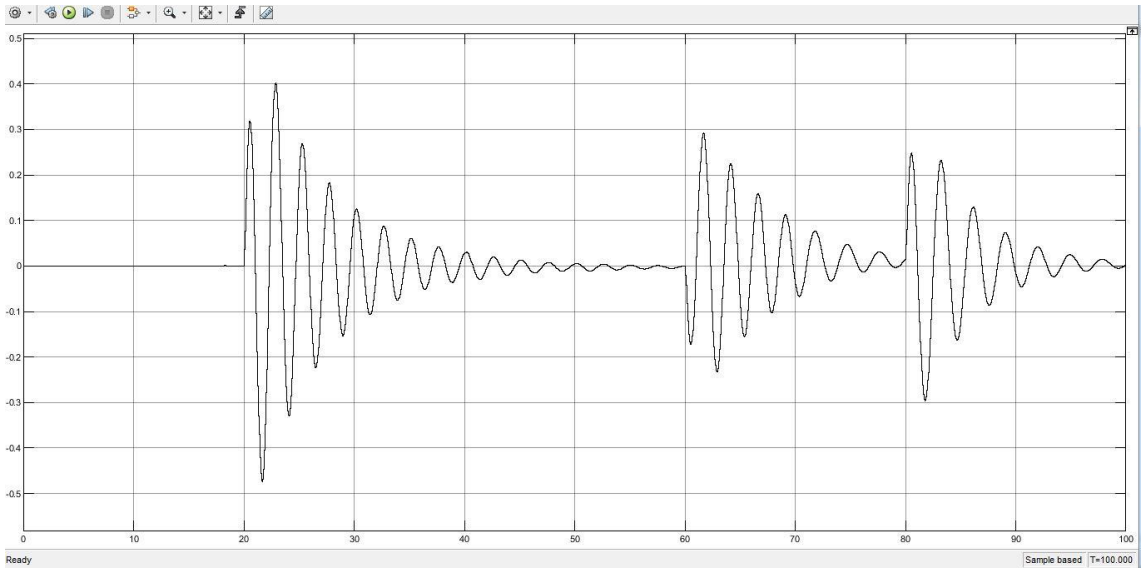


**Figura 66**

Posizione in Y con modello lineare

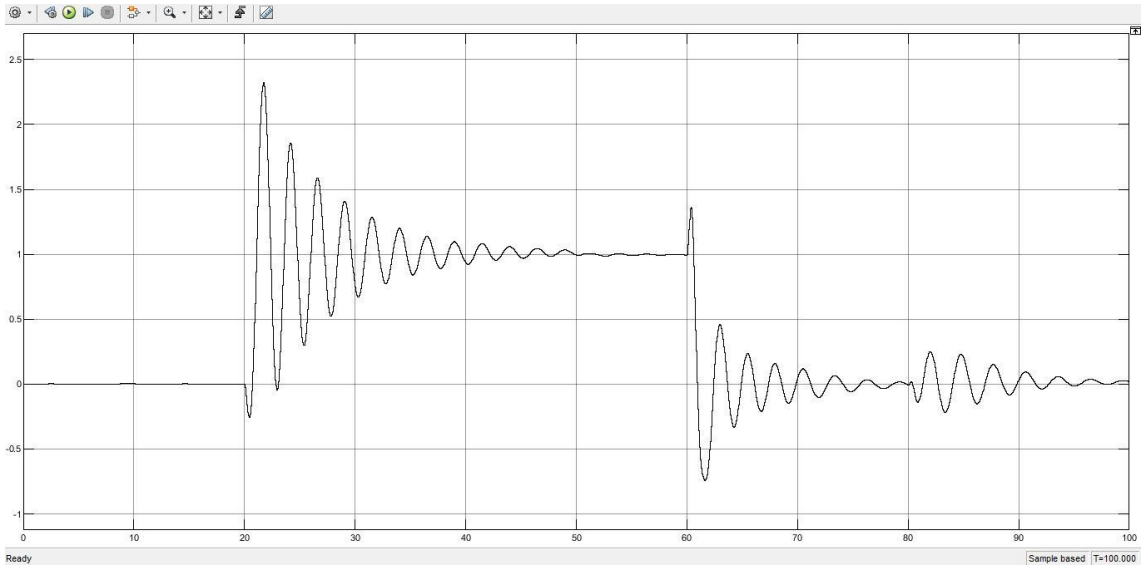


**Figura 67**

Rollio con modello non lineare

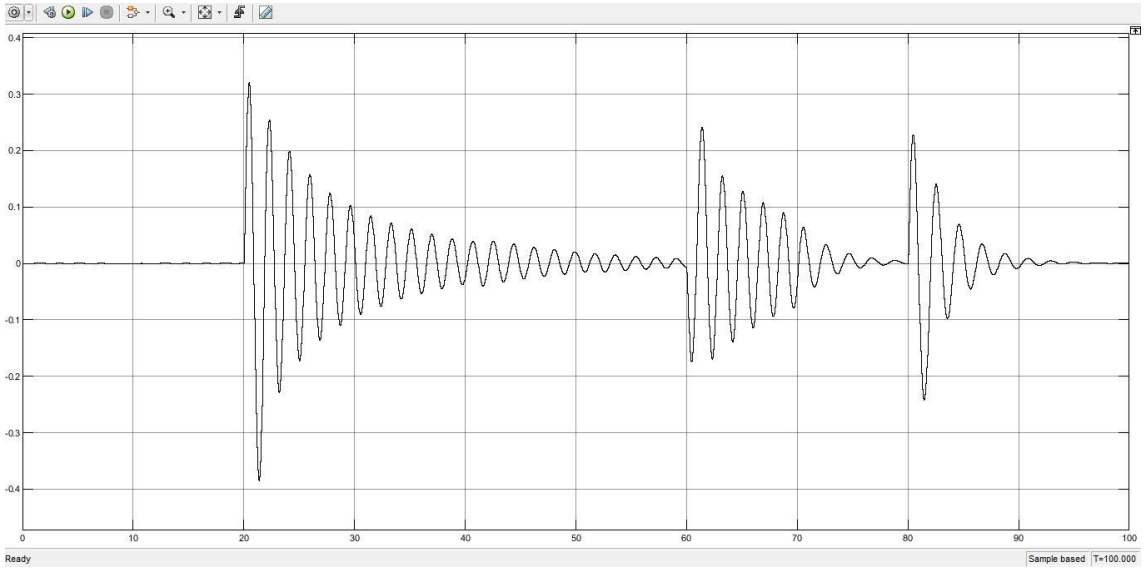


**Figura 68**

Posizione in Y con modello non lineare

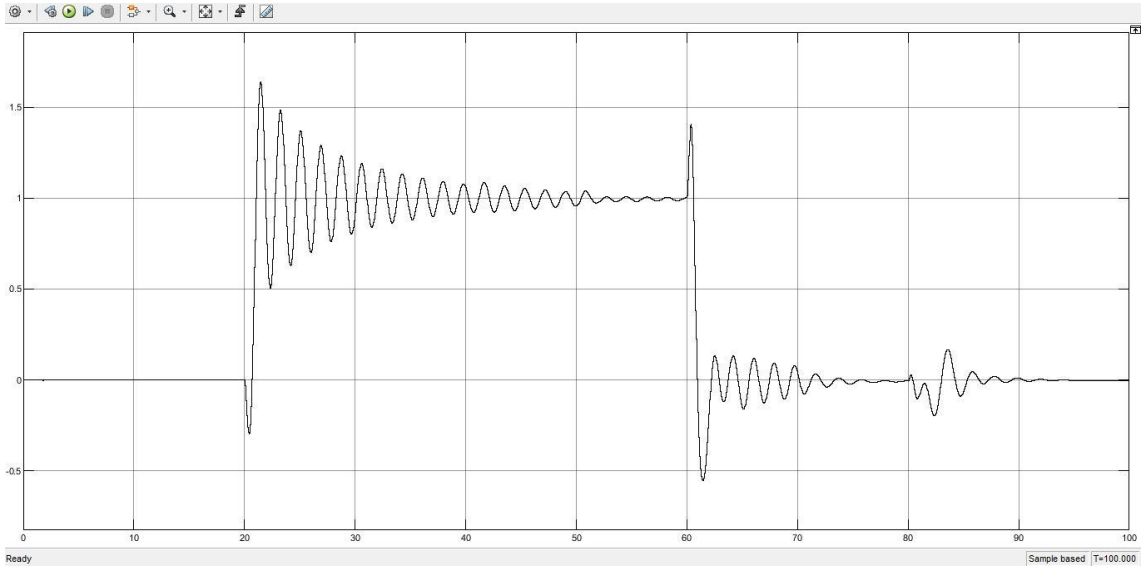


**Figura 69**

## 5.6. Miglioramento del controllore

Per migliorare il controllore è necessario ridurre l'elevato tempo di assestamento visibile in Figura 64, per farlo si ricorre a SISOTool.

### 5.6.1. Modifiche da SISOTool

Dall'applicazione SISOTool è possibile traslare i poli e gli zeri del diagramma di Bode manualmente. Effettuando dunque delle modifiche, con lo scopo di ridurre il tempo di assestamento, senza alterare però di molto le altre caratteristiche, si ottiene il controllore C4(s), del quale è stato elevato il guadagno fino a 0,043 per migliorare ulteriormente le prestazioni. Aumentare il guadagno non va contro le specifiche, infatti all'equazione **(41)** è richiesto che il guadagno sia almeno si 0,0274, non che sia pari ad esso. Il controllore C4(s) vale:

$$C4(s) = 0.043 * \frac{(s+0.8)(s+1)}{(s+0.1)(s+100)} \tag{47}$$

Le prestazioni mostrate in SISOTool sono riportate in Figura 70, dalla quale si osserva che rispetto alla Figura 64, il tempo di assestamento è notevolmente ridotto, si è mantenuto un buon tempo di salita, le oscillazioni sono velocemente smorzate e la sovraelongazione, benché sia leggermente aumentata non rappresenta una problematica.

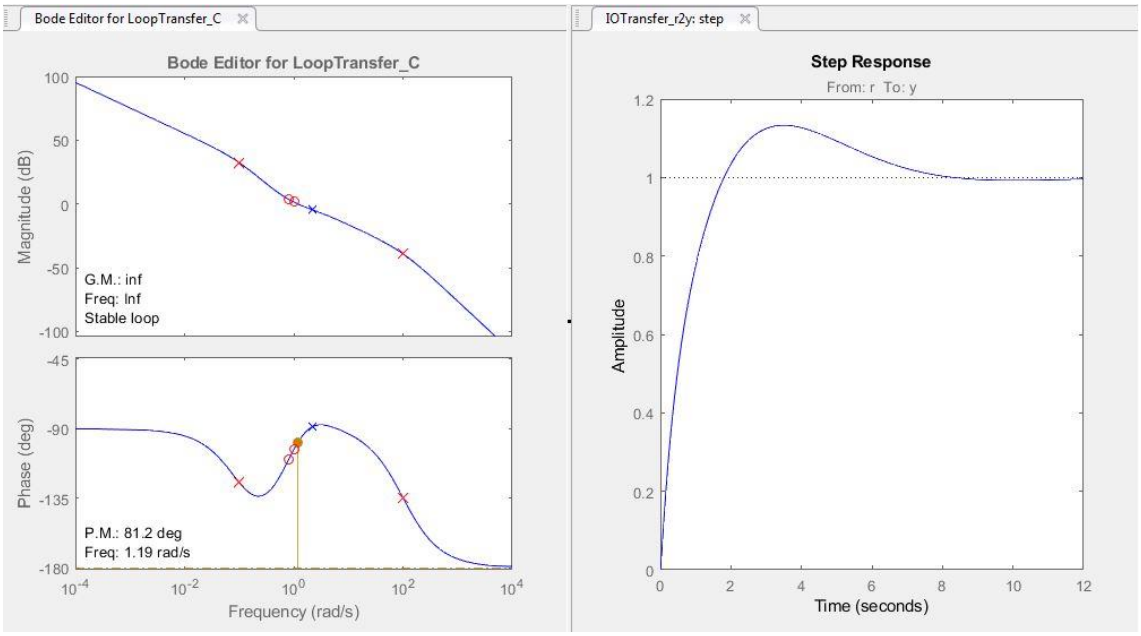


**Figura 70**

## 5.6.2. Simulazioni

I vettore dei poli, quello degli zeri e il guadagno statico di C4(s) vengono caricati quindi nel blocco zero-poli e le simulazioni forniscono i seguenti risultati: Figura 71 e Figura 72 per il modello matematico lineare, mentre Figura 73 e Figura 74 per quello non lineare.

Si osserva come il controllore agisca molto meglio del precedente e nel caso di modello matematico non lineare si avvicini molto al comportamento assunto dal PID originale visibile a Figura 48 e a Figura 49.

Nelle simulazioni effettuate adoperando il modello matematico lineare si notano più oscillazioni e delle sovraelongazioni leggermente maggiori, queste caratteristiche potrebbero essere migliorate (5.6.3.).

Rollio con modello lineare

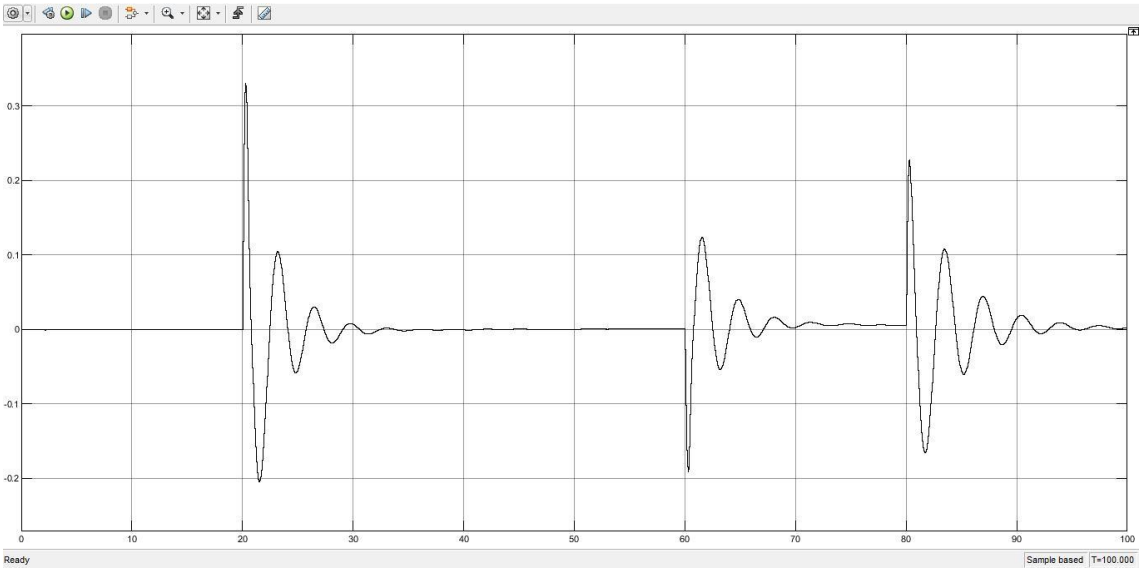


**Figura 71**

Posizione in Y con modello lineare

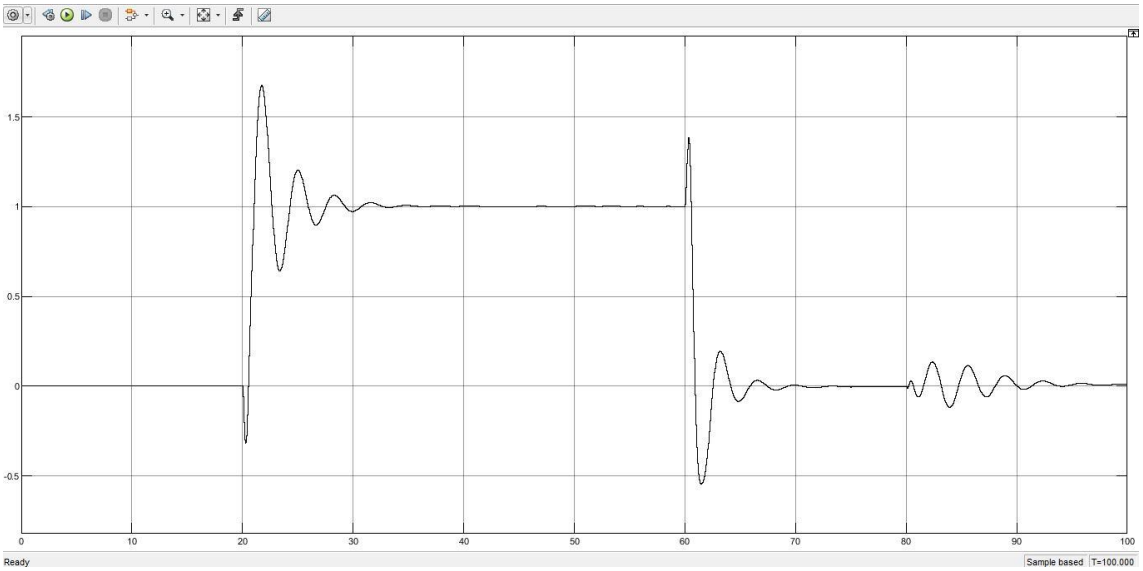


**Figura 72**

Rollio con modello lineare

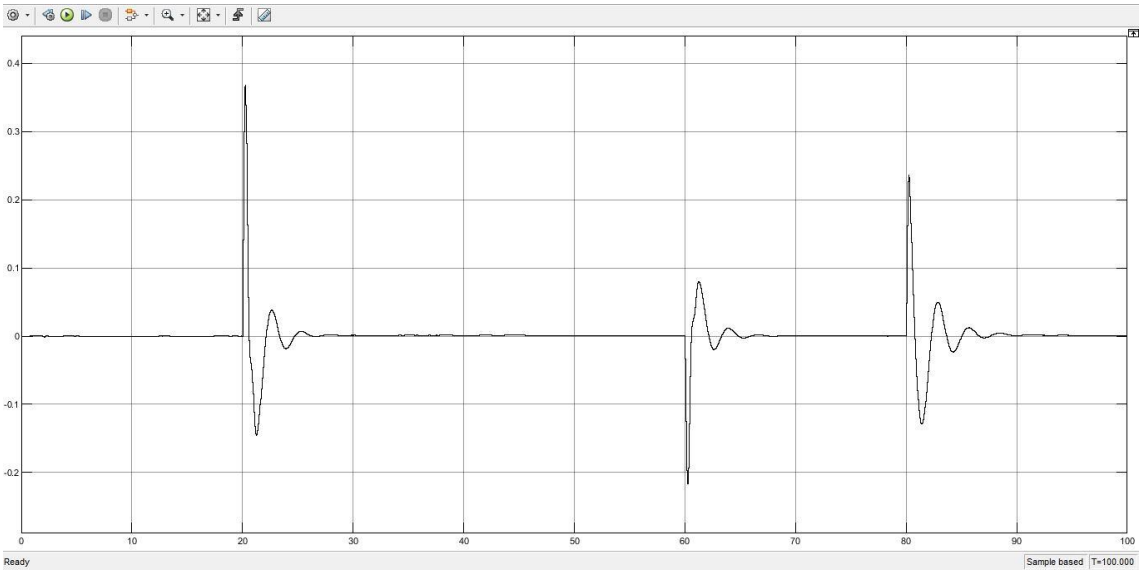


**Figura 73**

Posizione in Y con modello non lineare

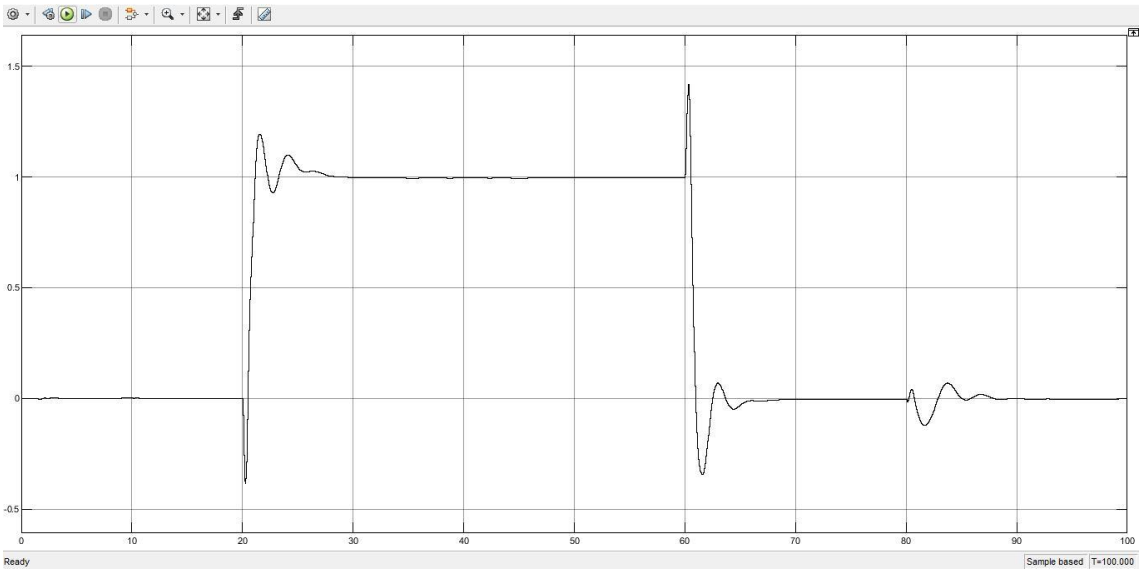


**Figura 74**

### 5.6.3. Eventuale correzione del controllore per il modello lineare

Per migliorare la risposta del controllore nel caso di modello matematico lineare, si potrebbe aumentare il valore del guadagno statico da 0,043 a 0,05 o 0,055. In questo modo si otterrebbero un rollio ed una posizione in Y visualizzabili nella Figura 75 e nella Figura 76. Da questi grafici si nota come siano presenti meno oscillazioni e anche la sovraelongazione sia diminuita. Aumentando il guadagno, diminuiscono il tempo di salita e di assestamento, questo potrebbe sembrare un fattore positivo, ma in realtà se si osservano i grafici di Figura 77 e di Figura 78, nel modello non lineare la risposta si sporca e si generano molte oscillazioni indesiderate. Questo succede dal momento che il controllore del rollio, diminuendo tempi di salita ed assestamento, assume una dinamica troppo veloce rispetto a quella del controllore a monte, che si occupa invece del controllo della posizione X-Y. Se quindi il controllore a valle ha una dinamica troppo veloce, il controllore a monte potrebbe non essere in grado di gestirlo, in quanto sarà meno sensibile alle variazioni di quanto non lo sia l'altro. In questo caso potrebbe generarsi uno scambio, fra i due controllori, di brusche variazioni, aventi lo scopo di compensarsi l'un l'altra. Questa reazione a catena, potrebbe diventare instabile con l'eccessivo aumento del guadagno. In Figura 79 sono riportate le risposte al gradino con un guadagno di 0,05 o di 0,055 con le varie caratteristiche che ne derivano.

Rollio con modello lineare e guadagno 0,055

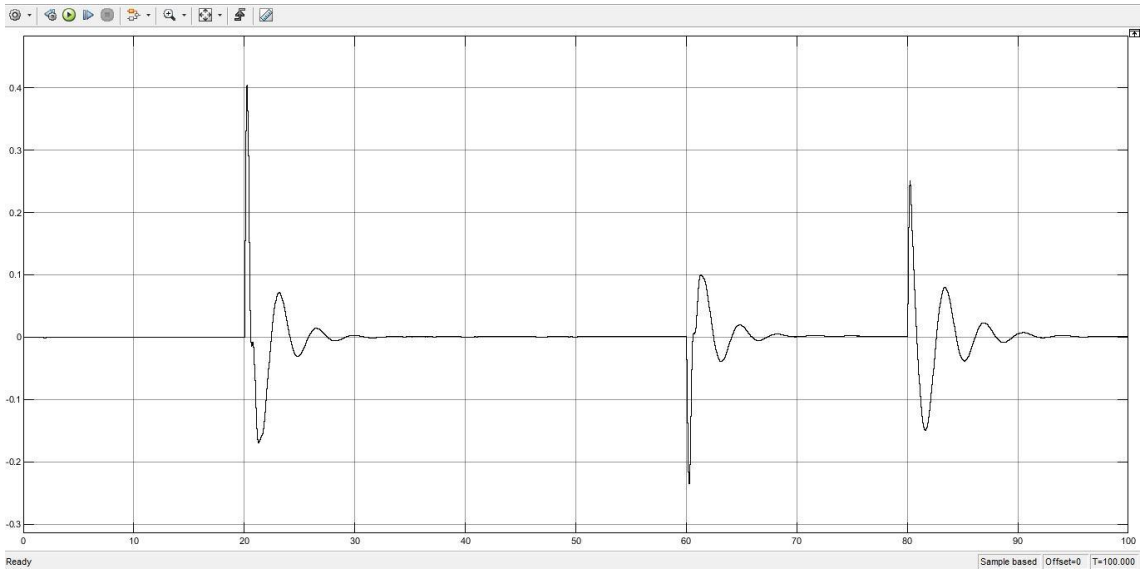


**Figura 75**

Posizione in Y con modello lineare e guadagno 0,055

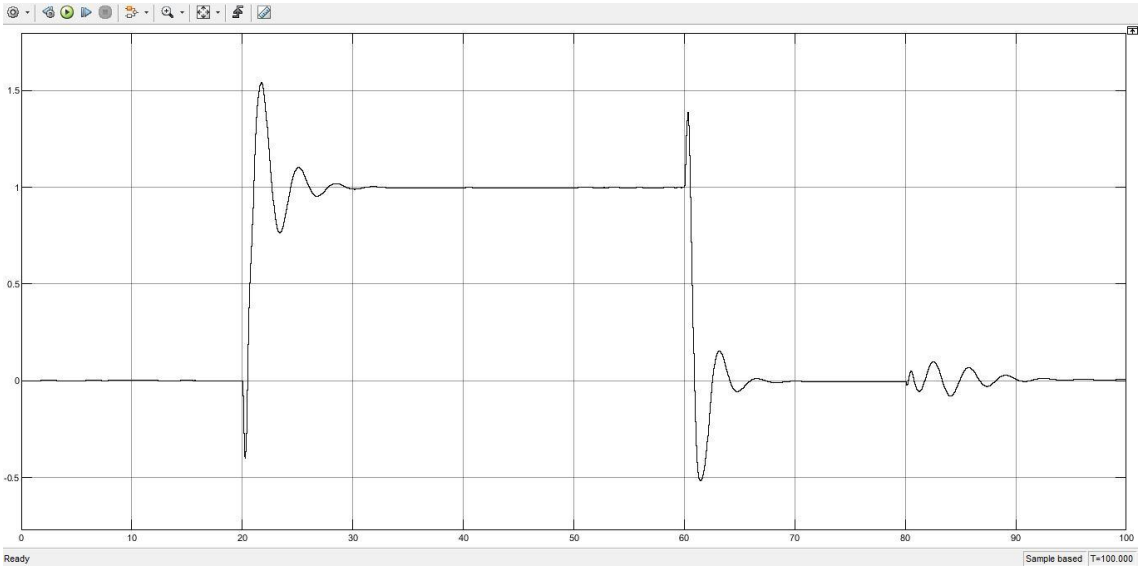


**Figura 76**

Rollio con modello non lineare e guadagno 0,055

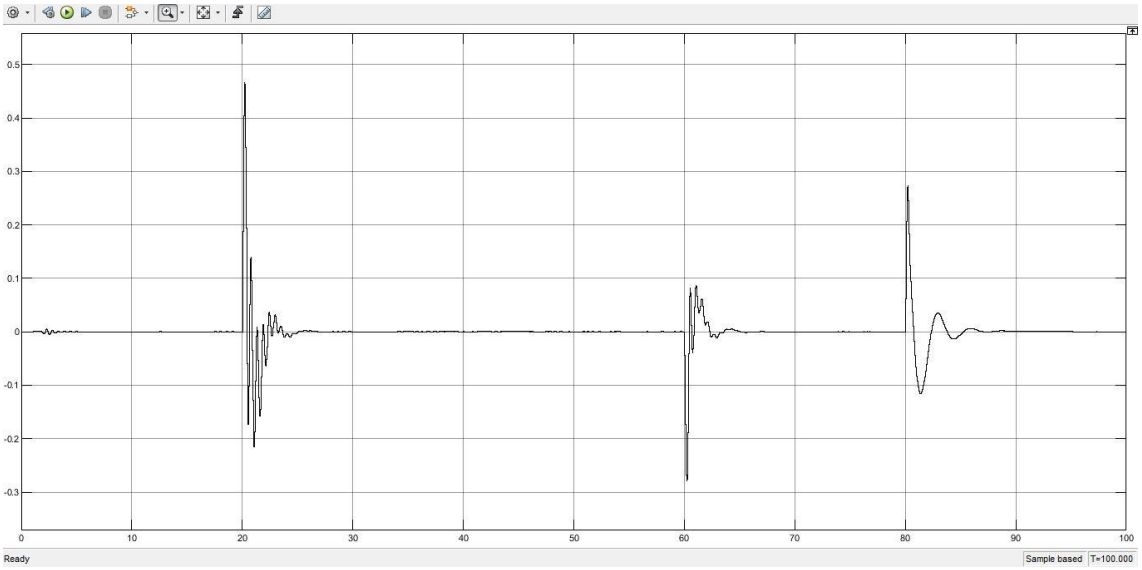


**Figura 77**

Posizione in Y con modello non lineare e guadagno 0,055

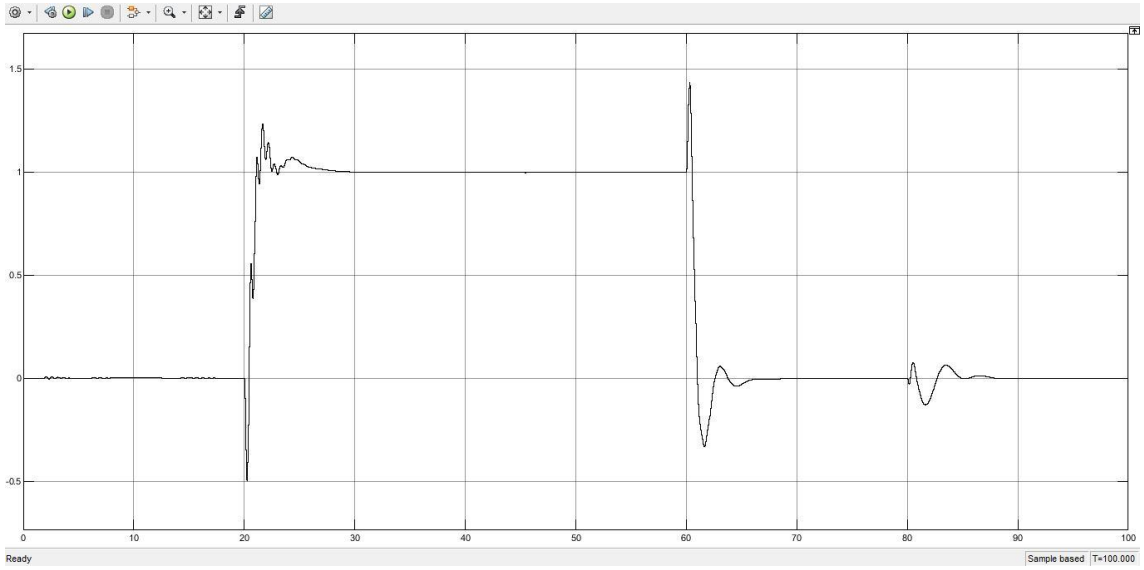


**Figura 78**

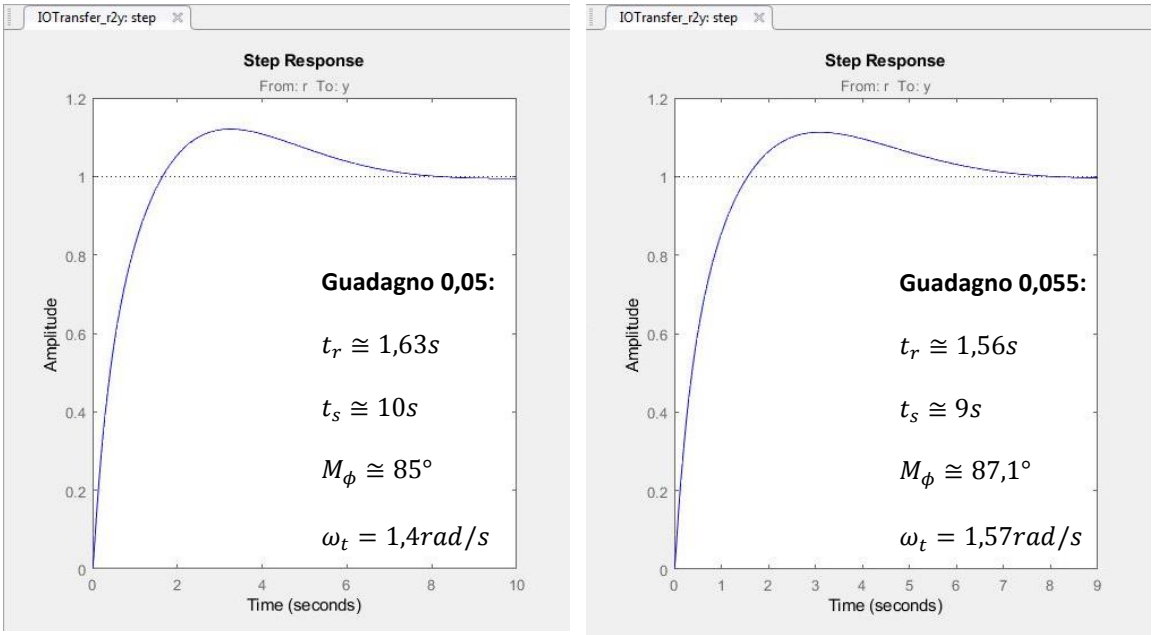


**Figura 79**

# 6. Conclusioni

C4(s) è in grado di offrire prestazioni simili al PID originale nel caso in cui venga utilizzato un modello matematico non lineare, ossia nel caso più simile alla realtà. Se invece fosse selezionato un modello matematico lineare, sarebbe comunque in grado di regolare il rollio, ma presentando delle oscillazioni in più.

La differenza fra i due casi potrebbe risiedere nel fatto che nell'ottenimento del modello lineare, delle dinamiche di natura non lineare siano state trascurate. Queste dinamiche trascurate però, forniscono nel modello non lineare dei contributi, che una volta accentuati dal maggiore guadagno, vanno poi a generare la reazione a catena fra i due controllori vista in precedenza.

Si può concludere affermando che essendo C4(s) un buon controllore nel caso del modello matematico non lineare, lo sarà con alta probabilità anche nel caso reale. Se però si volesse effettuare una simulazione utilizzando un modello matematico lineare è considerabile l'idea di aumentare il guadagno statico fino a 0,05 o 0,055 per delle migliori prestazioni.

# Bibliografia


- Mücahid Rıdvan Kaplan, Abdullah Eraslan, Aykut Beke and Tufan Kumbasar, *"Altitude and Position Control of Parrot Mambo Minidrone with PID and Fuzzy PID Controllers"*

- Teppo Luukkonen, *"Modelling and control of quadcopter",* August 22, 2011

- Agostino De Marco e Domenico P. Coiro, *"Orientamento del velivolo e trasformazione di assi",* marzo 2017

- Robert W. Deters, Or D. Dantsker, Stefan Kleinke, Narcrisha Norman and Michael S. Selig,*"Static Performance Results of Propellers Used on Nano, Micro, and Mini Quadrotors",* June 25-29, 2018

- MathWorks website: *https://www.mathworks.com/*