Università Politecnica delle Marche

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE
Corso di Laurea in Ingegneria Informatica e dell'Automazione

Tesi di laurea triennale

Studio e sviluppo di tecniche di controllo lineari per droni

Study and development of linear control techniques for drones

Candidato:
Davide Olivieri
Matricola 1083304

Relatore:
Prof. Ippoliti Gianluca

Correlatore:
Prof. Orlando Giuseppe

Anno Accademico 2019-2020

"La BBC li ritiene il futuro dell'agricoltura; Chris Anderson, direttore per undici anni di Wired e fondatore di 3D Robotics, li denisce i nuovi smartphone; Amazon li considera i nuovi corrieri, mentre gli analisti dell'Unione Europea li reputano il volano per una nuova rivoluzione industriale (e occupazionale) nel Vecchio Continente. Superando le considerazioni illustri, i droni sono più semplicemente l'occhio volante dei videomaker, uno strumento economico per le rilevazioni degli ingegneri e delle insostituibili vedette per il monitoraggio del dissesto idrogeologico. Insomma uno strumento di lavoro per molte categorie professionali."

05 maggio 2015 - Presentazione DRONEXPO

Indice

Elenco delle figure

Sommario

Lo svolgimento di questo progetto si è basato su un attento studio del modello matematico che descrive la dinamica di un quadricottero, in particolare del mini drone Mambo Parrot , il cui schema è fornito da Simulink.
Basterà digitare in Command Window di Matlab il seguente comando: *asbQuadcopterStart*, dopo aver installato le apposite librerie, per lavorare sullo schema a blocchi.
La fase realizzativa invece consiste nel sostituire il classico controllore PID, fornitoci dal modello di Simulink, su uno dei 6 gradi di libertà , con un controllore realizzato attraverso la sintesi per tentativi in frequenza.
Si è scelto come grado di libertà da controllare l'angolo di *Roll*.
Infine si è passati alla simulazione sul simulatore di volo *Simulink3D*, tracciando i grafici per le variabili da controllare.

1 Cenni Teorici sui Quadricotteri

Il quadricottero, è un aeromobile sollevato e spinto da quattro rotori, facente parte della categoria dei dispositivi *UAV (Unmanned Aerial Vehicle)* ovvero velivoli caratterizzati dall'assenza del pilota umano a bordo.
Il volo di tale mezzo è controllato dal computer a bordo del velivolo, sotto il controllo remoto di un navigatore o pilota, sul terreno o in un altro veicolo.
Le dimensioni di un quadricottero commerciale sono tipicamente non superiori al metro, il loro peso nell'ordine delle unità di kilogrammo e il loro costo spazia dalle centinaia alle migliaia di euro.
La piattaforma quadrirotore rappresenta un interesse rilevante nell'area dei controlli. È un sistema sottoattuato poiché ha sei gradi di libertà: **beccheggio**, **imbardata**, **rollio**, **x** (movimento nella direzione frontale del veicolo), **y** (movimento verso il lato sinistro del veicolo) e **z** (altitudine); ma è controllato utilizzando solo quattro attuatori.
Le piattaforme quadrirotore forniscono anche un'interessante prospettiva progettuale. La simmetria del design consente una centralizzazione dei sistemi di controllo e del carico utile.
Questa caratteristica agevola anche il sistema di controllo perché, anche con carichi diversi, è abbastanza semplice mantenere il baricentro nella stessa posizione, sull'asse verticale.

Entrando più specificatamente nell'argomento vediamo la costituzione di questo velivolo, mediante le sue parti fondamentali:

1) **Telaio:** rappresenta l'ossatura del drone, ossia la sua struttura portante, e varia a seconda del numero di motori.

2) **4 motori:** in cui ognuno è collegato a un'elica (propeller) che roteando permette al drone di alzarsi da terra. I motori installati sui droni sono motori elettrici, soprattutto di tipo "brushless", ossia "senza spazzole".

3) **Eliche**: svolgono la stessa funzione delle ali di un aereo e con la loro rotazione, creano in basso un campo di pressione che spinge verso l'alto il velivolo. Le eliche sono costituite da diversi materiali, che ne condizionano l'efficienza, ma i parametri più importanti sono il diametro ed il passo. Il diametro massimo delle eliche dipende dall'estensione del telaio, in modo che le eliche stesse non si sovrappongano. Il passo dell'elica è invece lo spostamento perpendicolare al loro piano, prodotto da una loro rotazione a 360°.

4) **ESC o Electronic Speed Controller:** è quella parte del drone che collega i motori al Flight Controller ed è costituita da un insieme di cavi nella cui parte centrale è posizionata una piccola scheda elettronica.

5) **Le batterie Li-Po** (Lithium-ion Polymer Batteries) : sono batterie ai Polimeri di Litio che vengono utilizzate sui droni in virtù del fatto che non possiedono alcun contenitore di metallo e risultano di conseguenza più leggere di quelle tradizionali.

6) **Flight controller :** È la centralina di bordo che rappresenta il sistema di autopilotaggio del drone. Questa unità processa i dati di volo e si occupa, tra le altre cose, di mantenere il multicottero "in equilibrio" durante il volo, agendo in automatico in base ad una lunga serie di informazioni ricavate da sensori e dall'hardware e dal software di cui esso dispone.

7) **IMU :** ovvero Inertial Measurement Unit è costituita da un insieme di componenti elettroniche che sono fondamentali per il funzionamento del drone. Questa parte può includere il GPS, giroscopi, accelerometri, barometri, magnetometri, optical flow e sonar, tutti strumenti di misurazioni inerziali, che permettono al Flight Controller di migliorare la risposta alle improvvise variazioni dei parametri fisico-elettromeccanici.

8) **Il sistema di controllo**: è solitamente costituito da un "radiocomando" trasmettitore (controller) a 2.4 GHz e da un ricevitore apposito che si interfaccia con il Flight Controller.

1.1 Caratteristiche Mambo Parrot S.A

Il minidrone Mambo Parrot (in Figura 1) è un quadricottero, dotato di quattro eliche rotanti azionate da motori coreless DC brusched, a differenza dei tradizionali DC brushed, non utilizzano un rotore di materiale ferromagnetico su cui sono collocati gli avvolgimenti rotorici ma dispongono gli avvolgimenti rotorici a formare una struttura cilindrica auto-supportante.

Figura 1: Mambo Parrot S.A

Vediamo le caratteristiche tecniche:

1. **Camera** : la telecamera integrata è in grado di girare video a 120° con risoluzione 720p con la possibilità di salvataggio su scheda SD. In alternativa, le immagini girate possono essere inviate in live streaming al visore Cockpitglasses 2.

2. **Batteria** : di tipo LiPo (Lithium Polymer) 660 mAh, Mambo ha un'autonomia di 8 minuti con accessori e di 10 minuti senza accessori. La durata della ricarica è di 30 minuti con caricatore.

3. **Stabilità**: Giroscopio a 3 assi e Accelerometro a 3 assi (IMU) per misurare l'accelerazione lineare e angolare; Barometro per misurare l'altitudine, Sensore a ultrasuoni per la misurazione della quota verticale; Sensore di telecamera per la stabilità orizzontale e verticale.

4. **Motore** : 4 motori coreless DC brusched e 4 eliche rotanti.

5. **Peso** : 63 g senza accessori e 73 g con fotocamera.

6. **Dimensioni** : $18x18cm$ (paraurti compresi).

7. **Velocità max** : $30km/h$

8. **Parrot Flypad** : Batteria LiPo 200 mAh con 6 ore di autonomia e 2 ore di carica. Il peso è 295 g con supporto per smartphone.

2 Modello Fisico

Il modello del quadricottero è schematizzabile secondo quanto mostrato in *Figura 2*, si osservi la terna trirettangolare solidale al velivolo con asse X_b verso il rotore frontale, Y_b orientato verso il rotore sinistro e l'asse Z_b ortogonale a X_b e Y_b verso la parte superiore del mezzo. Si definisce Ω_i la velocità angolare associata al rotore i-esimo: si osserva che la coppia di rotori 1 e 3 ha un senso di rotazione antiorario, mentre 2 e 4 ruotano in senso orario.

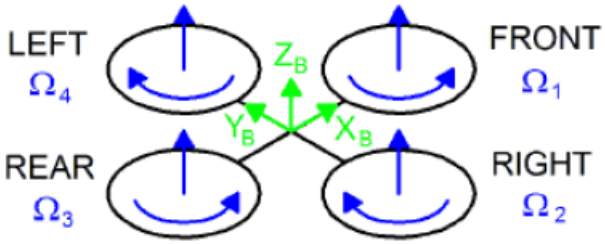
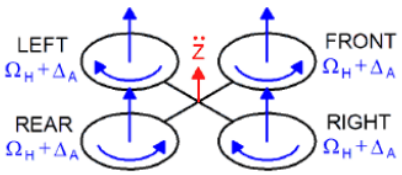
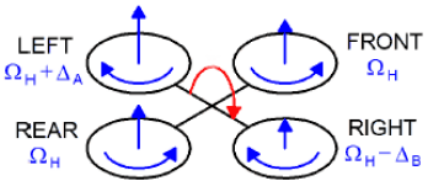


Figura 2: MOdello quadricottero

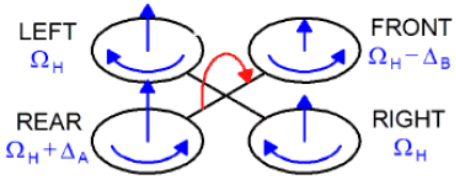
I movimenti concessi al quadricottero sono 4, riassumibili secondo quanto riportato in Figura 4: ***spinta, rollio, beccheggio e imbardata***



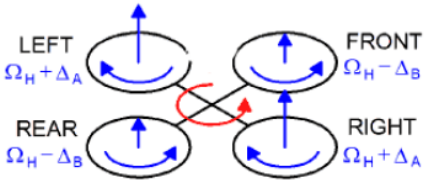
(a) Spinta



(b) Rollio



(c) Beccheggio



(d) Imbardata

Figura 3: Movimenti drone

Si definisce Ω_h *la velocità di rotazione* necessaria ad ogni rotore per indurre una forza al drone tale da contrastare la forza di gravità in condizione di hovering.

1) In *Figura 2(a)* è riportato il movimento di spinta lungo l'asse **Z** solidale al drone: la rotazione di tutti i rotori viene contemporaneamente incrementata di una quantità Δ_A, questo porta a una forza risultante lungo l'asse Z non nulla con conseguente accelerazione lungo Z.

2) In *Figura 2(b)* è riportato il movimento di **rollio(Roll)**, ovvero una rotazione attorno all'asse X solidale al drone: La rotazione associata ai rotori 1 e 3 rimane inalterata,mentre viene incrementata la rotazione del rotore 4 di un fattore Δ_A e decrementata la rotazione del rotore 4 di un fattore Δ_B. In tale modo la coppia indotta rispetto all'asse X non è più nulla, con conseguente accelerazione angolare attorno all'asse X.

3) In *Figura 2(c)* è riportato il movimento di **beccheggio(Pitch)**, ovvero una rotazione attorno all'asse Y solidale al drone: La rotazione associata ai rotori 2 e 4 rimane inalterata, mentre viene incrementata la rotazione del rotore 3 di un fattore Δ_A e decrementata la rotazione del rotore 1 di un fattore Δ_B. In tale modo la coppia indotta rispetto all'asse Y non è più nulla, con conseguente accelerazione angolare attorno all'asse Y.

4.) In *Figura 2(d)* è riportato il movimento di **imbardata(Yaw)**, ovvero una rotazione attorno all'asse Z solidale al drone: viene incrementata la rotazione dei rotori 2 e 4 di un fattore Δ_A e decrementata la rotazione dei rotori 1 e 3 di un fattore Δ_B. In tale modo la coppia indotta rispetto all'asse z non è più nulla, con conseguente accelerazione angolare attorno all'asse z.

2.1 Modello Matematico

Dinamica Quadrirotore

Prima di poter descrivere le equazioni che governano un velivolo, come il quadrirotore, è necessario introdurre le coordinate di riferimento che ci consentono di descrivere l'assetto e la posizione.
Nel caso del quadrirotore è possibile utilizzare due sistemi di riferimento, uno fisso, e uno mobile solidale al telaio le cui dinamiche possono essere espresse rispetto alla terna fissa.
La terna fissa, detta anche *inerziale*, è una terna dove si può considerare valida la prima legge di Newton; è stato scelto di usare come terna fissa una terna chiamata *NED* (North-East-Down), $\mathbf{O_{NED}}$ che come si evince dalla *Figura 4* che segue ha i versori gli assi rivolti verso Nord Est e verso il centro della terra.
La terna mobile a cui si è fatto riferimento prima è invece solidale con il baricentro del quadcopter, ed è nota come **terna $\mathbf{0_{ABC}}$** dove *ABC* sta per *Aircraft Body Center*.

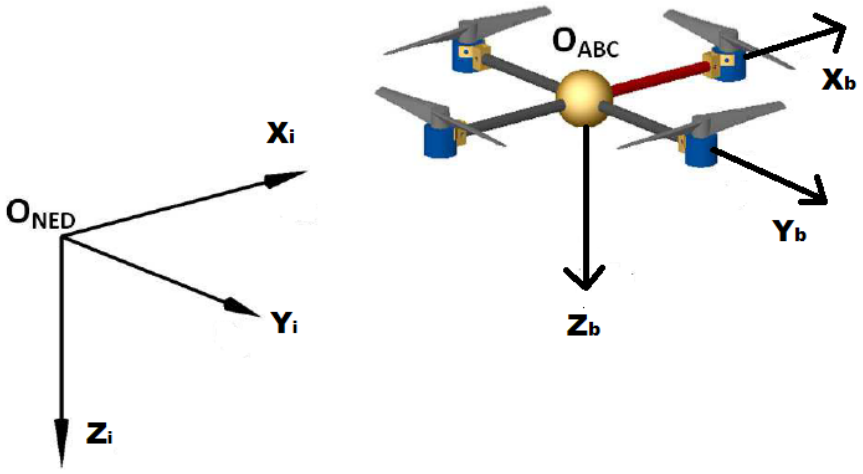


Figura 4: Terna inerziale (O_{NED}) e Terna mobile (O_{ABC})

Nella teoria del controllo le dinamiche del sistema modellato sono espresse tipicamente attraverso lo stato del sistema che nel caso del quadrirotore corrispondono a *12 equazioni* che ne descrivono l'assetto e la posizione a sei gradi di libertà.
Le variabili di interesse sono quindi 6 per il sistema *NED* ed altre 6 per il sistema *ABC*.
In particolare definiamo come $\boldsymbol{V_b}$ il vettore di velocità lineari e come $\boldsymbol{\Omega_b}$ il vettore velocità angolari del veivolo nella terna O_{ABC} con le seguenti variabili di stato:

$$\mathbf{v}_b = \begin{bmatrix} u \ v \ w \end{bmatrix}^T \tag{1}$$

$$\omega_b = \begin{bmatrix} p \ q \ r \end{bmatrix}^T \tag{2}$$

Per il sistema $\mathrm{O_{ABC}}$ definiamo le posizioni lineari:

$$\Lambda_{NED} = \begin{bmatrix} x \ y \ z \end{bmatrix}^T \tag{3}$$

e rispettivamente gli angoli di : Roll,Pitch,Yaw

$$\Theta_{NED} = \begin{bmatrix} \phi \ \theta \ \psi \end{bmatrix}^T \tag{4}$$

Per procedere alla stesura delle equazioni in forma di stato a riguardo della terna *NED* è necessario l'uso di una matrice di trasformazione che permette di descrivere, componente per componente l'orientamento del frame mobile rispetto al fisso.
Per legare i due tipi di riferimenti verrà applicata la teoria degli angoli di Eulero relativa alle rotazioni X-Y-Z, ovvero verrà ruotato l'asse $\mathrm{Z_i}$ di un angolo ψ , l'asse $\mathrm{X_i}$ di un angolo ϕ e infine l'asse $\mathrm{Y_i}$ di un angolo θ.
La matrice di trasformazione complessiva sarà composta dal prodotto di 3 matrici di rotazione rispetto agli assi di rotazione:

Matrice di rotazione attorno all'asse X: R(X,ϕ)

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\phi & sen\phi \\ 0 & -sen\phi & cos\phi \end{bmatrix} \tag{5}$$

Matrice di rotazione attorno all'asse Y: R(Y,θ)

$$\begin{bmatrix} cos\theta & 0 & -sen\theta \\ 0 & 1 & 0 \\ sen\theta & 0 & cos\theta \end{bmatrix} \tag{6}$$

Matrice di rotazione attorno all'asse Z: R(Z,ψ)

$$\begin{bmatrix} cos\psi & sen\psi & 0 \\ -sen\psi & cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{7}$$

La matrice risultante è il prodotto delle tre matrici ortogonali R^3 :

$$R_i^b = \begin{bmatrix} cos\theta cos\phi & cos\theta sen\phi & -sen\theta \\ -cos\phi sen\psi + sen\phi sen\theta cos\psi & cos\phi cos\psi + sen\phi sen\theta sen\psi & sen\phi cos\theta \\ sen\phi sen\psi + cos\phi sen\theta cos\psi & -sen\phi cos\psi + cos\phi sen\theta sen\psi & cos\phi cos\theta \end{bmatrix} \tag{8}$$

Dunque si introduce il legame:

$$\begin{bmatrix} x_b \\ y_b \\ z_b \end{bmatrix} = R_i^b * \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} \tag{9}$$

Allo stesso modo sfruttando il fatto che la matrice R è ortonormale è possibile ricavare le coordinate inerziali utilizzando:

$$R^T = R^{-1} \tag{10}$$

Una volta ottenute le matrici che consentono il passaggio dal sistema di riferimento solidale al veicolo a quello inerziale, si possono ricavare le altre grandezze fisiche di interesse, ad iniziare dalle velocità.
Per ottenere le velocità traslazionali rispetto alle coordinate di navigazione è sufficiente eseguire la trasformazione:

$$\mathbf{v}_i = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = R_i^b * V_b \tag{11}$$

Mentre per ottenere le velocità rotazionali rispetto alle coordinate inerziali è sufficiente esegueri la seguente trasformazione:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & sen\phi\frac{sen\theta}{cos\theta} & cos\phi\frac{sen\theta}{cos\theta} \\ 0 & cos\phi & -sen\phi \\ 0 & \frac{sen\phi}{cos\theta} & \frac{cos\phi}{cos\theta} \end{bmatrix} * \begin{bmatrix} p \\ q \\ r \end{bmatrix} = Q_b^i * \omega_b \tag{12}$$

In maniera analoga alla precedente è possibile ricavare le velocità angolari del riferimento del veivolo sfruttando la relazione di ortonormalità di Q.

2.2 Equazioni della dinamica del corpo rigido

Ipotizzando che gli assi principali del sistema di riferimento solidale al veicolo O_{ABC} coincidano con gli assi principali di inerzia O_{NED}, in tali condizioni la seconda legge di Newton è direttamente applicabile:

$$m \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \sum F_i \tag{13}$$

$$I \begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = \sum \tau_i \tag{14}$$

Riscrivendo le equazioni utilizzando le espressioni (11) e (12) descritte nella sezione precedente:

$$m \frac{\mathrm{d}}{\mathrm{dt}} \left(R_b^i \begin{bmatrix} u \\ v \\ w \end{bmatrix} \right) = \sum R_i^b * F_b \tag{15}$$

$$I \frac{\mathrm{d}}{\mathrm{dt}} \left(Q_b^i \begin{bmatrix} p \\ q \\ r \end{bmatrix} \right) = \sum R_i^b \tau_b \tag{16}$$

Esplicitando le derivate utilizzando la regola di derivazione del prodotto e ricordando due ugualianze fondamentali : $\dot{R_b^i} = \omega_b \times R \quad e \quad \dot{Q_b^i} = \omega_b \times Q$ otteniamo il seguente sistema :

$$\sum F_b = m \begin{bmatrix} \dot{u} + qw - rv \\ \dot{v} + ru - pw \\ \dot{w} + pv - qu \end{bmatrix} \tag{17}$$

$$\sum \tau_b = \begin{bmatrix} \dot{p} I_x + qr(I_z - I_y) \\ \dot{q} I_y + pr(I_x - I_z) \\ \dot{r} I_z + pq(I_y - I_z) \end{bmatrix} \tag{18}$$

Osserviamo che quanto finora mostrato vale per un qualsiasi corpo rigido soggetto a delle forze ed a delle coppie.

2.3 Equazioni della dinamica non lineari del quadricottero

Ora specifichiamo meglio cosa accade nel caso del quadrirotore andando a vedere quali sono le forze e le coppie agenti sul frame.
Osserviamo che F_b può scomporsi in due pezzi il primo che è dato dalla forza gravitazionale che chiamiamo : $F_g = \begin{bmatrix} 0 & 0 & mg \end{bmatrix}^T$ ed il secondo è chiamato con terminologia aeronautica *spinta* ed è già espresso in termini del frame ABC,a differenza del primo che dovrà essere trasformato attraverso la matrice di rotazione R.
Otteniamo dunque:

$$R_i^b \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \end{bmatrix} \tag{19}$$

Discorso analogo va fatto per il momento di inerzia,dovremmo dunque aggiungere all'equazione (18) anche l'azione dei momenti meccanici esterni dovuti all'azione dei motori,cioè :

$$\begin{bmatrix} bl_y(-\Omega_1^2 - \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \\ bl_x(\Omega_1^2 - \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \\ d(\Omega_1^2 - \Omega_2^2 + \Omega_3^2 - \Omega_4^2) \end{bmatrix} \tag{20}$$

b = coefficiente spinta dei motori
d = coefficiente resistenza dell'aria
l = proiezioni braccio del drone

A questo punto sommando l'espressione (19) nell'equazione del sistema relativa alla forza (17),e riordinando i termini otteniamo:

$$\begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = \begin{bmatrix} m\dot{u} \\ m\dot{v} \\ m\dot{w} \end{bmatrix} = \begin{bmatrix} mqv - mrv - sen\theta \cdot mg \\ mru - mpw + sen\phi cos\theta \cdot mg \\ mpv - mqu + cos\phi cos\theta \cdot mg + b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \end{bmatrix} \tag{21}$$

In maniera analoga sommando l'espressione (20) nell'equazione del sistema relativa ai momenti (18),e riordinando i termini otteniamo:

$$\begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \begin{bmatrix} \dot{p}I_x \\ \dot{q}I_y \\ \dot{r}I_z \end{bmatrix} = \begin{bmatrix} bl_y(-\Omega_1^2 - \Omega_2^2 + \Omega_3^2 + \Omega_4^2) - qr(I_z - I_y) \\ bl_x(\Omega_1^2 - \Omega_2^2 - \Omega_3^2 + \Omega_4^2) - pr(I_x - I_z) \\ d(\Omega_1^2 - \Omega_2^2 + \Omega_3^2 - \Omega_4^2) - pq(I_y - I_z) \end{bmatrix} \tag{22}$$

In questa trattazione non è stata modellato l'effetto giroscopico, certamente presente, ma viste le ridotte velocità in gioco è stato ritenuto trascurabile.
Le equazioni sopra enunciate rappresentano la dinamica del quadricottero.

3 Modellazione in Simulink

Attraverso il comando *asbQuadcopterStart* apriamo lo schema completo fornitoci da *Simulink*.

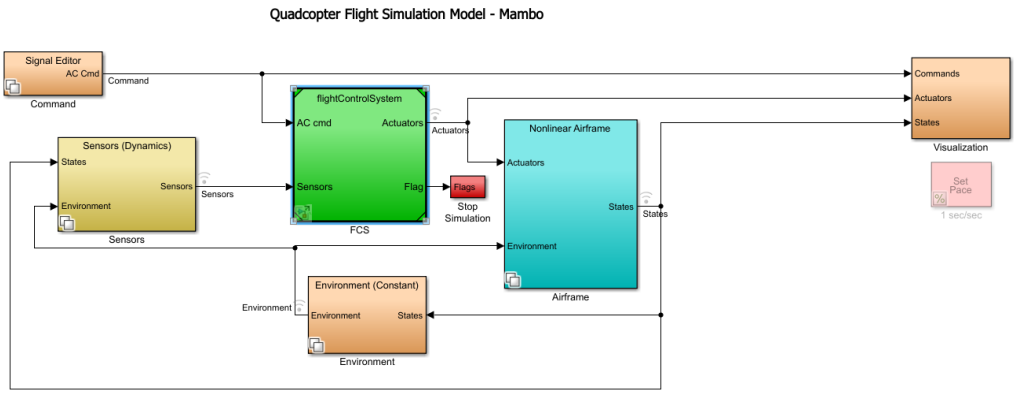


Figura 5: Modello Simulink del quadcopter

Come si può notare il classico schema in controreazione, in Figura 5, è costituito da 5 blocchi fondamentali :

1. **Signal Editor** è il blocco che genera il segnale di riferimento;
2. **Sensor** è il blocco che contiene le relazioni tra i diversi sensori a bordo del drone;
3. **Airframe** è il blocco che rappresenta il processo;
4. **FCS** (*Flight controll system*) rappresenta il controllore del processo;
5. **Enviroment** è il blocco che contiene le variabili legate all'ambiente;
6. **Visualization** comprende un blocco ausiliaro per la visualizzazione delle variabili;

3.1 Signal Editor (Generatore)

Questo blocco rappresenta il generatore di segnali di riferimento, al suo interno troveremo quattro modalità per generare segnali:

1. **Signal Builder**: selezionabile con il comando *VSS_ Command = 0*, è il costruttore di segnali di dafault da noi utilizzato. Si può notare come i segnali *X, Y, Z* vadano a fornire il riferimento di posizione (*posRef*) mentre i segnali *Yaw, Pitch, Roll* costituiscano il riferimento in orientamento (*orientRef*).

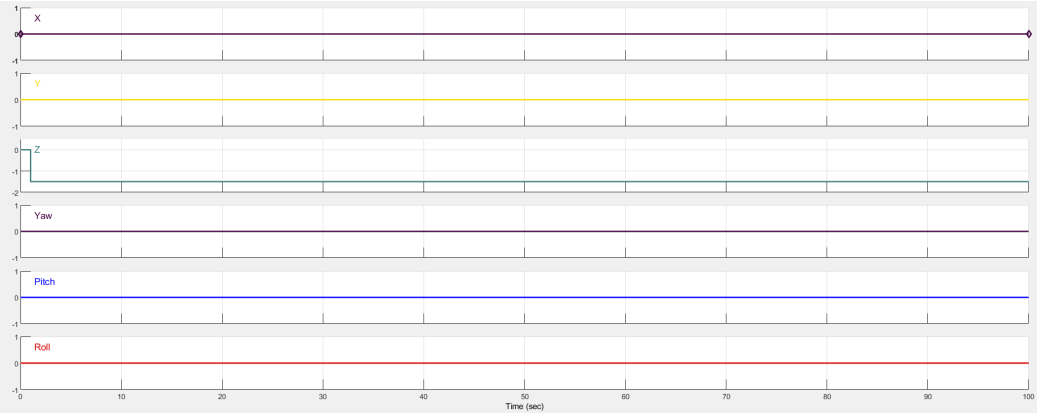


Figura 6: Segnali di riferimento

   Si è scelto dunque di generare un segnale di rifermento a 1,5 *m* lungo l'asse *Z*. Il segno negativo è dovuto alla scelta del sistema di riferimento citato nella sezione 2.1 .

2. **Joistick**: selezionabile con il comando *VSS_ Command = 1*, mi permette di gestire manualmente il drone attraverso un joistick.

3. **Data**: selezionabile con il comando *VSS_ Command = 2*, mi consente di gestire i segnali di riferimento selezionabili sul file *cmData.mat*, presente nella cartella del progetto.

4. **Spreadsheet Data**: selezionabile con il comando *VSS_ Command = 3*, mi consente invece si selezionare i segnali di riferimento dal file *cmData.xlsx* .

3.2 Sensor

Questo blocco rappresenta e gestisce tutta la sensoristica a bordo del drone vista nella sezione 1.1 . All'interno di questo troveremo due possibili blocchi da utilizzare:

Sensors Feedthrough : selezionabile con il comando *VSS_SENSORE = 0*, il quale rappresenta il blocco dei sensori senza disturbi in retroazione.

Sensors Dynamics : selezionabile con il comando *VSS_SENSORE = 1*, è il blocco di default in cui vengono considerati gli errori sui sensori, che dovranno essere poi corretti dai controllori del sistema.

Entrambi i blocchi ricevono in ingresso gli stati in uscita dal processo e le variabili legate all'ambiente,uscenti dal blocco *Enviroment*. Come si può notare aprendo il blocco *Sensor System* è diviso in due sotto-blocchi indipendenti tra loro,come mostra la *Figura 7*:

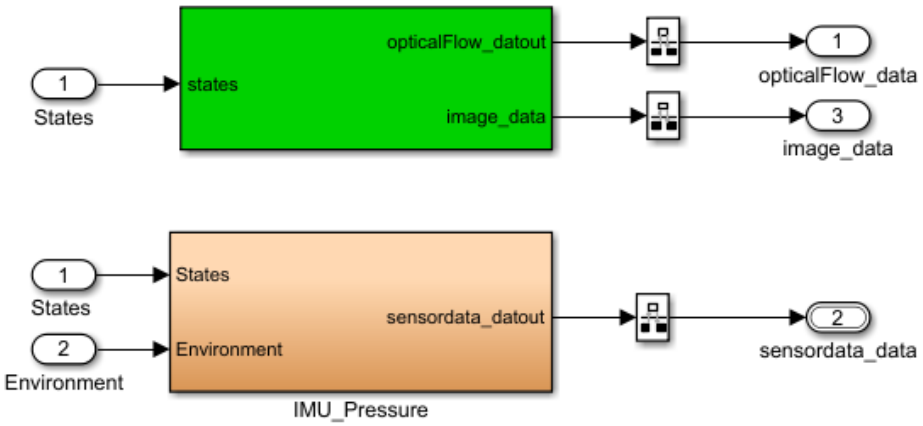


Figura 7: Sensor System

In *verde* è rappresentato il sottosistema che si occupa dell'elaborazione delle immagini provenienti dalla fotocamera.

In *arancione* il sottosistema in cui avviene l'elaborazione di tutti i dati provenienti dagli altri sensori compresi nell'IMU, cioè giroscopio, accelerometro, sonar e sensore di pressione. La sua funzione è quella di trasformare i dati ricevuti da questi sensori, relativi al Body Frame, in valori relativi al sistema di riferimento inerziale.

3.3 Airframe (Processo)

Rappresenta il processo del sistema, all'interno del blocco, *Simulink* ci fornisce due modelli di processo selezionabili:

NonLinearAirframe : modello non lineare selezionato di default.

LinearAirFrame : modello lineare, fornito da *Simulink* selezionabile attraverso il comando *VSS_ VEHICLE = 0* nella Command Window.

Il modello non lineare (**NonLinearAirframe**) garantisce una maggiore fedeltà di risposta rispetto al caso lineare in quanto si basa sul modello matematico sviluppato nella sezione 2.3, il quale contiene diversi contributi non lineari.

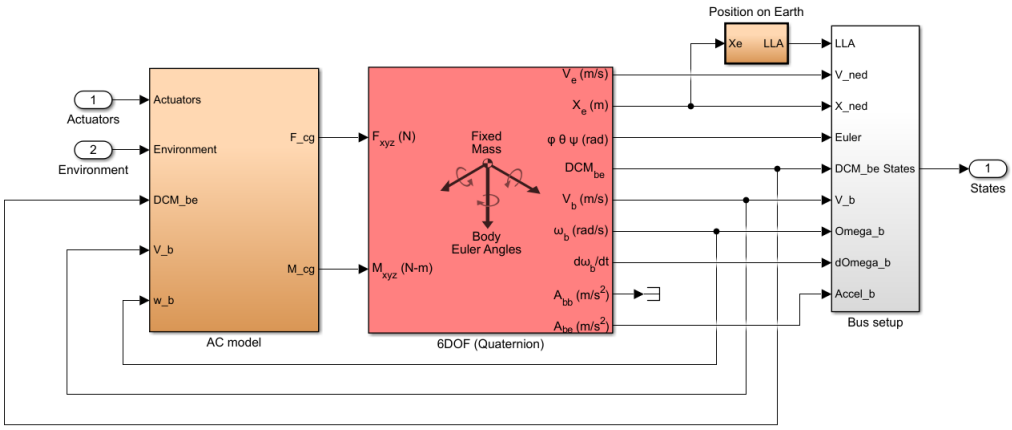


Figura 8: Modello del processo non lineare

Come si nota in *Figura 8*, è costituito da quattro sotto blocchi :

1. **AC Model**: è il blocco che contiene la parte della dinamica, in ingresso troviamo le velocità angolari w_b e le velocità lineari v_b espresse rispetto al sistema solidale con il body del drone ed uscenti dal blocco 6DOF. In uscita si hanno le forze e i momenti su tutti e tre gli assi.

   Entrando nel dettaglio di *AC Model* troviamo i seguenti blocchi mostrati in figura:

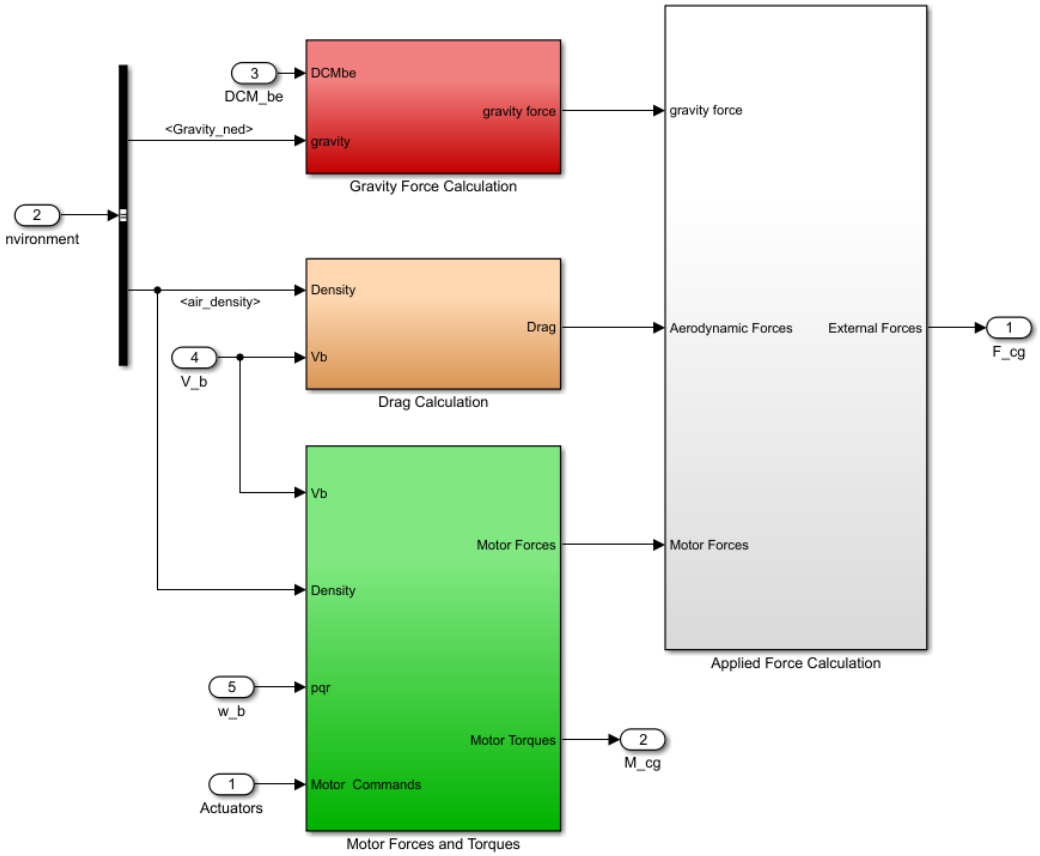


Figura 9: AC Model Non Lineare

AC Model / Gravity Force Calculation (*Figura 9* in *rosso*) calcola la forza di gravità rispetto al sistema solidale con il drone. Ha come ingresso la matrice di rotazione DCM, che contiene i coefficienti della matrice *Direct Cosine Matrix* necessari per il passaggio dal sistema di riferimento del corpo rigido a quello inerziale; e la costante di gravità.

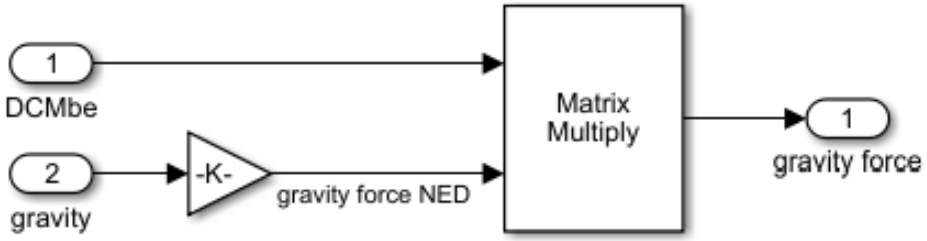


Figura 10: Calcolo forza di gravità

AC Model / Drag Calculation (*Figura 9* in *arancione*) calcola la forza di resistenza che l'aria imprime al drone in movimento.

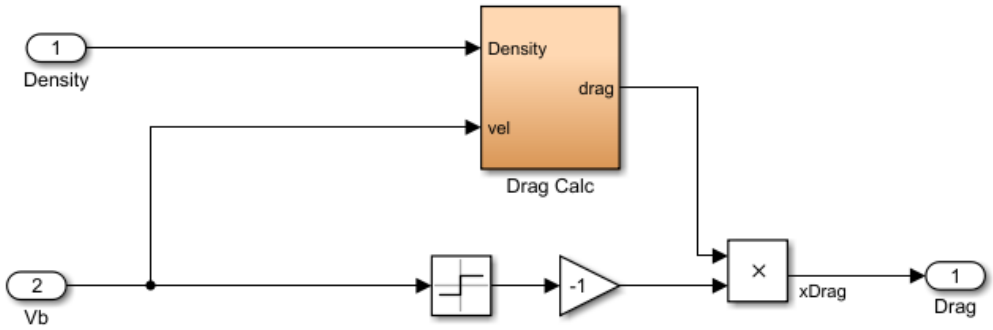


Figura 11: Calcolo forza di resistenza aria

Tale forza è dovuta allo spostamento del drone ed è esprimibile come:

$$D = \frac{1}{2}\rho V^2 S C_D \tag{23}$$

Dove:

ρ : la densità dell'aria;

V : la velocità di traslazione del drone;

S : la superficie che impatta l'aria (il fluido) durante la traslazione;

C_D : il coefficiente di drag.

AC Model / Motor Forces and Torques (*Figura 9* in *verde*) contiene il vero e proprio modello descritto dalle equazioni viste nelle sezioni (2.2) e (2.3).

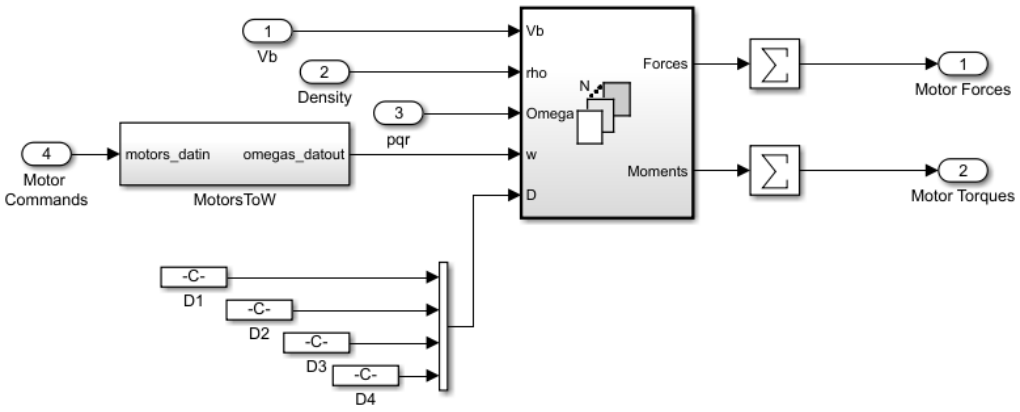


Figura 12: Calcolo delle forze e coppie del sistema

In *Figura 10* sopra si nota un blocco centrale il quale ha come ingressi le velocità lineari (V_b) e angolari (pqr) rispetto al body frame, il vettore Motor Commands (w) il quale contiene lo sforzo di controllo che arriva dal controllore ed infine il vettore (D) che descrive il posizionamento rispetto al centro di massa dei rotori.

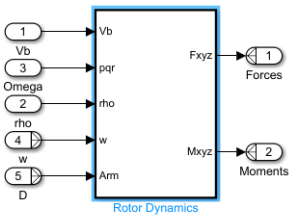


Figura 13: Dinamica del sistema

La *Figura 13* ci mostra come ricavare la dinamica del mio sistema,rappresentato nel modello matematico, sezione (2.1), all'interno di questo blocco centrale troveremo la dinamica di ogni motore diviso in *Motori DX* (di destra) e *Motori SX* (di sinistra).
In particolare dalla *Figura 12* si nota un ulteriore blocco chiamato *MotorsToW* il quale trasforma lo sforzo di controllo, in un segnale interpretabile dai motori.

AC Model / Applied Fource Calculation (*Figura 9* in *bianco*) questo blocco è responsabile della creazione di un vettore *Exsternal Force* avente 3 componenti, visibili in *Figura 12*.

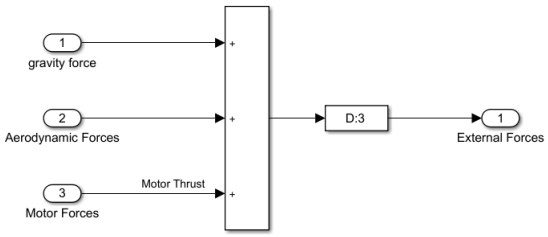


Figura 14: Forze esterne

2. **6DOF (Quaternion)**: Il secondo blocco fondamentale della *Figura 8*, prende in ingresso le forze e i momenti visti nel blocco *AC Model* che usando i quaternioni va a calcolare posizioni e velocità sia nel sistema di riferimento inerziale che in quello solidale col drone, fornendo cosìle variabili di stato ai successivi blocchi.

   Questo blocco descrive dunque le 6 equazioni del moto del drone presenti nella sezione (2.2), descritte dalle formule (21) e (22).

3. **Position on Earth**: il terzo è il blocco che prende in ingresso X_e , posizione rispetto al sistema inerziale, e ne calcola LLA, cioè latitudine, longitudine e altitudine in coordinate geodetiche.

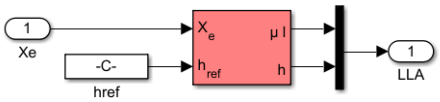


Figura 15: Position on Earth

4. **Bus Setup**: il quarto ed ultimo blocco compone un vettore di bus. Contiene tutte le velocità in entrambi i sistemi di riferimento oltre alla posizione espressa *North - East - Down*, accellerazioni e angoli di rotazione.

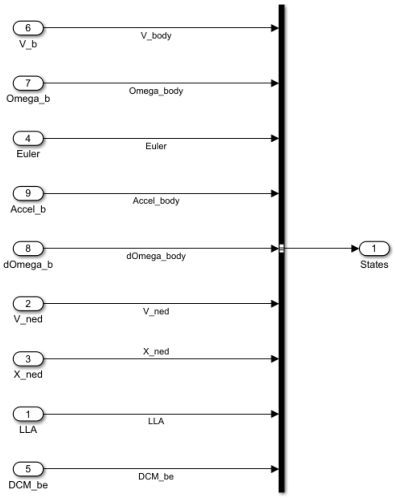


Figura 16: Variabili di stato

3.4 Flight Control System (Controllore)

Il modello *asbQuadcopter* implementa di base un controllore composto da sei PID: uno per il controllo dello *Yaw*, uno per la quota di volo e due PID a cascata per per l'angolo di *Roll* così come per il *Pitch*.
Il blocco *Simulink*, visibile in *Figura 17*, riceve come segnali di ingresso i valori dei riferimenti provenienti dal blocco *Signal Builder* e i valori delle variabili di stato provenienti dai sensori in retroazione. In uscita troviamo i segnali di riferimento degli attuatori che effettueranno il controllo sui motori del modello non lineare.
L'uscita *Flags* è un'uscita di sicurezza che, in caso di perdita di assetto da parte del drone, fornisce un segnale di controllo che interrompe la simulazione.

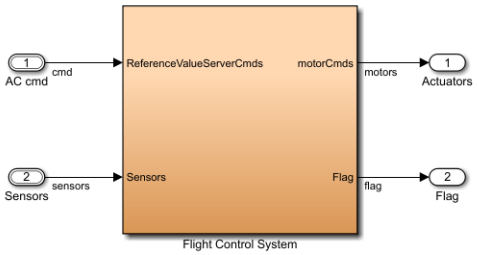


Figura 17: Flight Control System

Il blocco *FCS* risulta piuttosto complesso ed è il punto fondamentale su cui si è strutturato tutto il lavoro svolto.

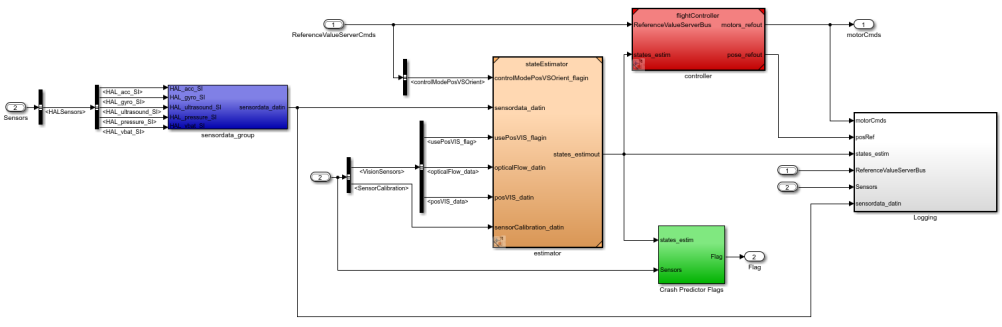


Figura 18: Flight Control System

Come è possibile vedere dalla *Figura 18*, il blocco è suddiviso in 4 sotto blocchi fondamentali, quali :

1. **SensorData Group**, evidenziato in *blu* nella *Figura 18*, il quale riceve in ingresso le letture grezze da parte dei sensori, giroscopio, accellerometro e sensore di pressione.

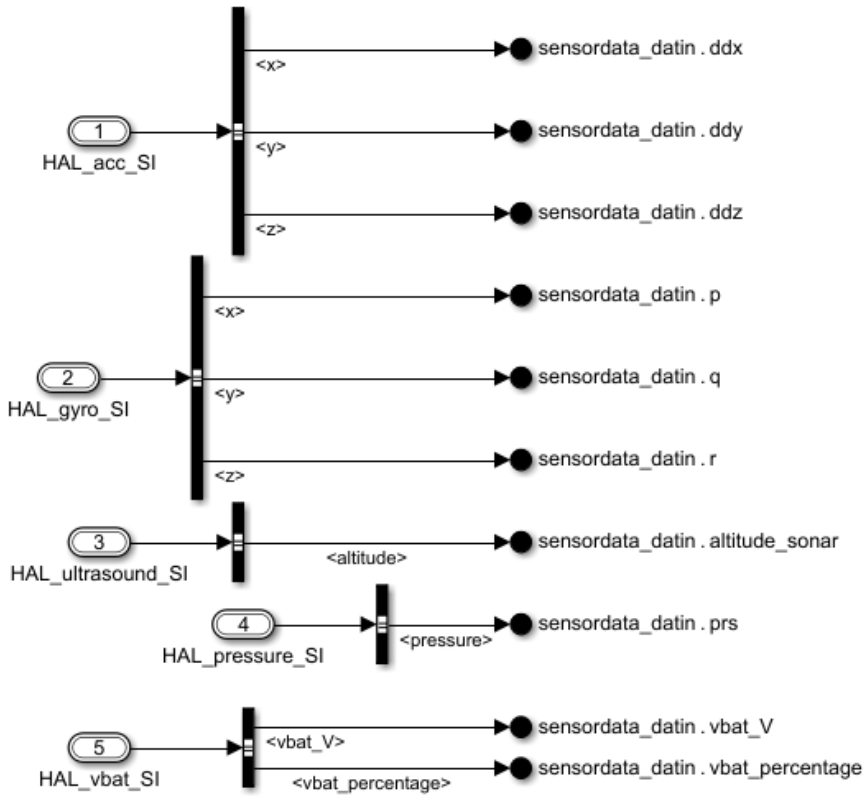


Figura 19: Flight Control System

Una volta valorizzati i dati presi sul bus centrale e forniti dai vari sensori a bordo del drone, trasferisce l'uscita al blocco *Estimator* e al blocco *Logging*.

2. **State Estimator**, evidenzianto in *arancione* nella *figura 18*, prende in ingresso i risultati letti dal blocco *SensorData Group* e da altri sensori che mi indicano la posizione XY, l'orientamento e l'altitudine, ne stima il risultato e lo trasferisce in ingresso al controllore e al *Flag Prediction*.

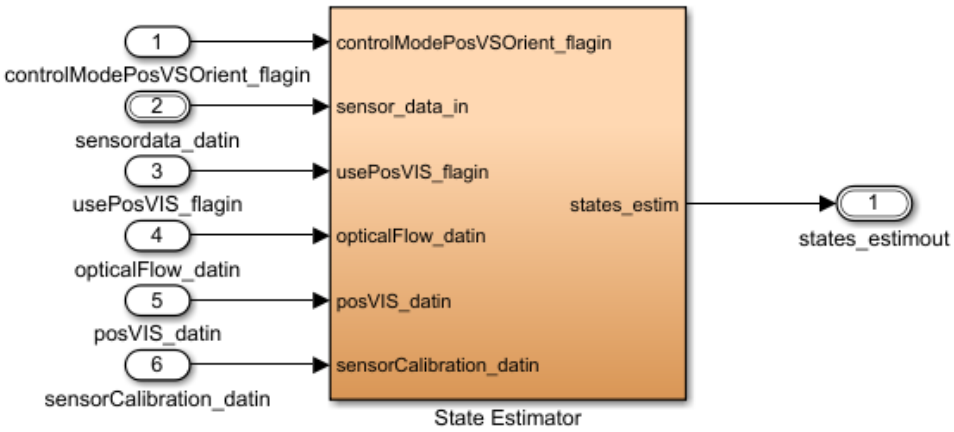


Figura 20: Blocco Estimator

3. **Crash Predictor Flags**, evidenziato in *verde* nella *Figura 18*, è il blocco di sicurezza il quale riceve in ingresso i valori letti dai sensori a bordo del drone e i valori stimati dal blocco *Estimator*, li confronta con quelli ideali in modo da verificare se ci sono le condizioni necessarie al volo, in caso contrario il drone farà un atterraggio di sicurezza.

4. **Controller**, evidenziato in *rosso* nella *figura 18*, è il blocco fondamentale, è il vero e proprio controllore, ed è costituito da sei PID: uno per il controllo dello *Yaw*, uno per la quota di volo e due PID a cascata per per l'angolo di *Roll* così come per il *Pitch*.

Il blocco *Controller* è composto da quattro sottosistemi principali e da tre blocchi ausiliari evidenziati in *verde* nella *figura 21*, dei quali *Switch Ref Att* consente il passaggio dal segnale di riferimento rispetto alla posizione, al segnale di riferimento rispetto all'orientamento del veicolo; mentre gli altri due blocchi ausiliari *Control Mixer* e *Thrurst to Motor Command* forniscono le costanti moltiplicative relative all'azione degli attuatori.

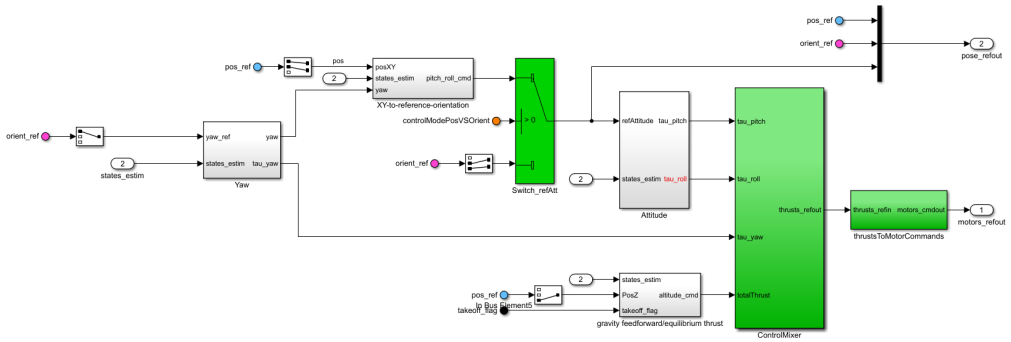


Figura 21: Controller

Vediamo ora i 4 sotto-blocchi principali, colorati in *bianco*, i quali contengono i sei PID del sistema:

Yaw : contiene il controllore PD per l'angolo di Yaw, come si nota dalla *figura 22* si ha un controllore Proporzionale - Derivativo e una struttura del tipo *Derivative Output of Controller* dove la parte derivativa non viene eseguita sull'errore ma sull'uscita, in questo caso r.

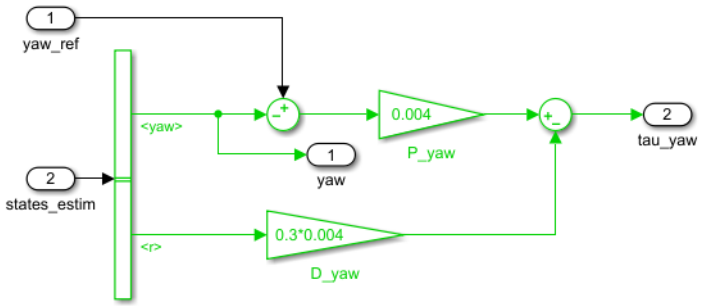


Figura 22: Controller Yaw

Riceve in ingresso il riferimento relativo all'angolo di *Yaw*, il valore corrente di tale angolo e la relativa velocità angolare. In uscita fornisce il segnale di controllo per l'angolo di Yaw.

XY to Reference Orientation : questo blocco adotta una struttura PD, proporzionale e derivativa, è responsabile del controllo delle grandezze sul piano XY, è intuibile che una rotazione attorno all'asse Z, cioè variando l'angolo di *Yaw*, genera una variazione sulla posizione XY.

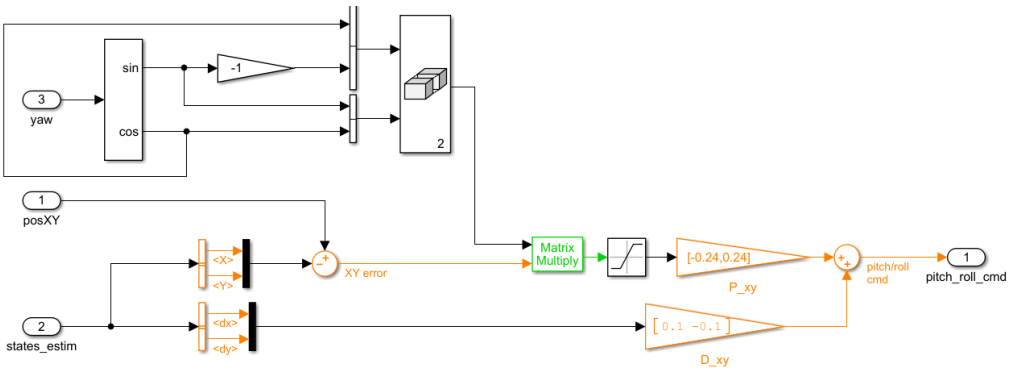


Figura 23: XY to Reference Orientation

Come si nota dalla *Figura 23*, utilizzando l'angolo di *Yaw*, si calcolano le componenti X, Y rispetto al sistema di riferimento inerziale dovute alla rotazione del veicolo sull'asse Z, in modo tale da sommarli agli errori sulla posizione longintudinale e latitudinale.

Attitude : questo blocco può ricevere in ingresso o il segnale di controllo generato dal blocco *XY to Reference Orientation* o il segnale di orientamento proveniente dal blocco *Signal Builder*, a secondo della posizione dello *Switch*.

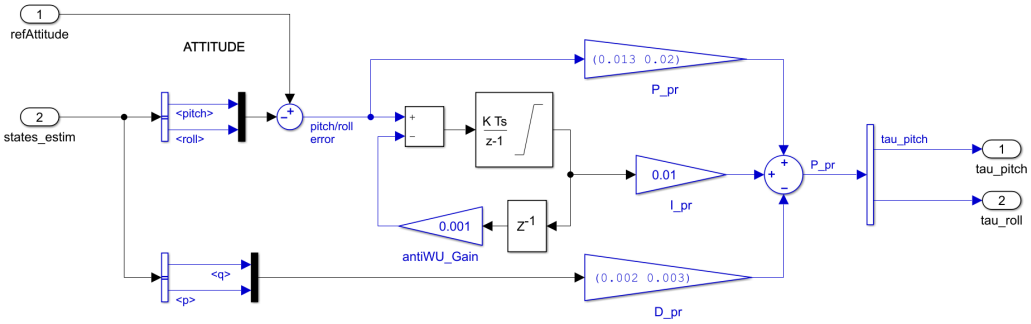


Figura 24: Attitude

Il controllore PID (*Figura 24*), proporzionale, integrativo e derivativo, riceve in ingresso gli angoli di *Pitch* e *Roll* e le rispettive velocità angolari p,q. Confronta tali valori con quelli stimati dal blocco *State Estimator*, in modo da ricavare i rispettivi errori, utilizzati dal controllore per avere in uscita i rispettivi valori corretti. Si nota anche l'aggiunta di un polo all'errore con lo scopo di evitare il fenomeno dell' Integral Wind Up dovuto all'azione integrale. Questo blocco verrà poi modificato in seguito isolando i due angoli e trovando un nuovo controllore per l'angolo di *Roll*.

Gravity feedforward / Equilibrium thrust : è il blocco che controlla l'altitudine sull'asse Z del drone, è composto da un controllore PD, che riceve in ingresso la posizione e la velocità sull'asse Z stimate dal blocco *State Estimator*, li confronta con quelli reali in modo da ricavarne l'errore.

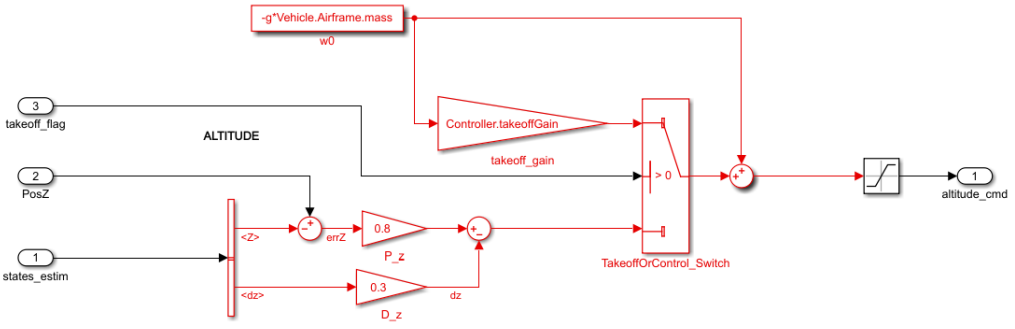


Figura 25: Gravity feedforward and Equilibrium thrust

L'uscita del controllore come si vede dalla *Figura 25*, va in blocco di switch (*TakeOfforControll*), il quale attraverso la variabile booleana *takeofflag* ci indica se il drone è in fase di decollo o meno, in modo tale da regolare la spinta verticale dei motori.

I valori in uscita da tutti i controllori giungono al blocco *Controll Mixer* mostrato in *Figura 21*.

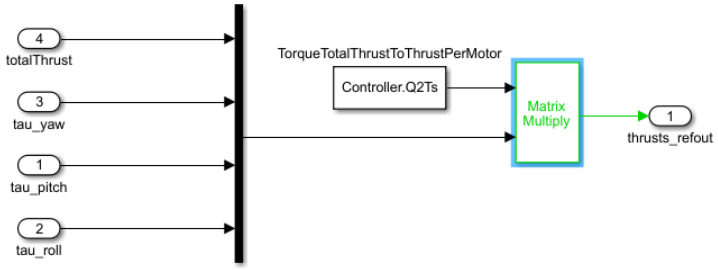


Figura 26: Matrice costanti di spinta motori

il quale li moltiplica per la matrice *ControllerQ2ts*, che descrive le costanti con cui, a seconda della variabile di stato da controllare, viene gestita la partizione del segnale di controllo.

```
>> Controller.Q2Ts

ans =
    thurst  tau yaw  tau pitch  tau roll
    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 27: Matrice costanti di spinta motori

4 Linearizzazione di un sistema

I sistemi dinamici reali come nel nostro caso il drone, non sono mai dei sistemi perfettamente lineari, ma possono essere approssimati nell'intorno di un movimento prefissato, come il punto di equilibrio. Su questo concetto è dunque possibile creare un modello linearizzato, il quale è molto più utili e facili da studiare durante l'analisi di quest'ultimo.

Sia un sistema dinamico, non lineare e MIMO, descritto nello spazio degli stati dalla equazione di stato e dalla trasformazione dell'uscita di seguito riportate:

$$\dot{x}(t) = f((x(t), u(t)) \tag{24}$$

$$y(t) = g(x(t)), u(t)) \tag{25}$$

Si calcolano le condizioni di equilibrio o punto di equilibrio imponendo che per l'ingresso $u(t)$ sia tale che $u(t) = \bar{u}$, sia verificata la condizione: $\dot{x}(t) = 0$, cioè in formule:

$$u(t) = \bar{u} \tag{26}$$

$$x(t) = \bar{x} \tag{27}$$

$$y(t) = \bar{y} \tag{28}$$

Dunque:

$$\dot{x} = 0 \tag{29}$$

Allora le equazioni della dinamica del sistema diventano:

$$0 = f(\bar{x}, \bar{u}) \tag{30}$$

$$\bar{y} = g(\bar{x}, \bar{u}) \tag{31}$$

Si considera l'equilibrio $(\bar{x}, \bar{y}, \bar{u})$ trovato al punto precedente e si linearizza il sistema nell'intorno del punto di equilibrio, come segue:

$$u(t) = \bar{u} + \delta u(t) \tag{32}$$

$$x(t) = \bar{x} + \delta x(t) \tag{33}$$

$$y(t) = \bar{y} + \delta y(t) \tag{34}$$

Da cui si ottiene il modello linearizzato sul punto di equilibrio:

$$\delta\dot{x}(t) = \boldsymbol{A}\delta x(t) + \boldsymbol{B}\delta u(t) \tag{35}$$

$$\delta y(t) = \boldsymbol{C}\delta x(t) + \boldsymbol{D}\delta u(t) \tag{36}$$

In cui le matrice della dinamica del sistema sono ottenute in questo modo:

$$A = \left.\frac{\partial f(x,u)}{\partial x}\right|_{(\bar{x},\bar{u})} \qquad B = \left.\frac{\partial f(x,u)}{\partial u}\right|_{(\bar{x},\bar{u})}$$

$$C = \left.\frac{\partial g(x,u))}{\partial x}\right|_{(\bar{x},\bar{u})} \qquad D = \left.\frac{\partial g(x,u)}{\partial u}\right|_{(\bar{x},\bar{u})}$$

4.1 Processo lineare in Simulink

Simulink, come accennato precedentemente, oltre al modello non lineare del processo, ci offre un modello linearizzato secondo le regole sopra riportate. Attraverso il comando $VSS\_VEHICLE = 0$ nella Command Window di *Matlab* è possibile visualizzare questo modello:

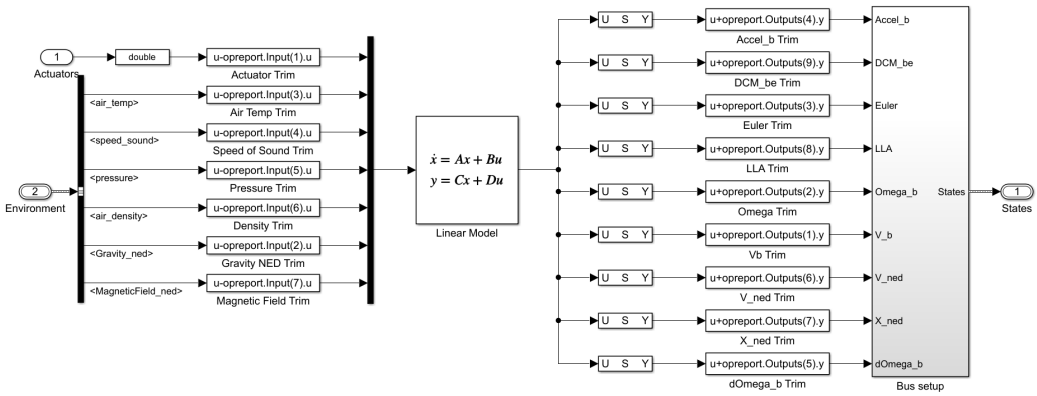


Figura 28: Modello del processo linearizzato

Gli ingressi e l'uscita sono gli stessi visti per il processo non lineare, ma ora abbiamo un modello lineare descritto dalle matrice della dinamica A, B, C, D . Da queste matrici è possibile ricavarci il sotto-sistema relativo all'angolo di *Roll*, in maniera tale da sintetizzarci poi un nuovo controllore. Attraverso il comando *load('linearizedAirframe.mat')* è possibile caricare il seguente file in *Matlab*, il quale contiene le matrici della dinamica del sistema:

phi — *Matrice A* — *p*

	1	2	3	4	5	6	7	8	9	10	11	12
1	0	3.6611e-17	0	0	0	0	0	0	0	1	3.0292e-23	-5.2296e-12
2	-3.6611e-17	0	0	0	0	0	0	0	0	0	1	5.2295e-12
3	-6.8347e-13	-2.0323e-28	0	0	0	0	0	0	0	0	-5.2295e-12	1.0000
4	0	-9.8100	0	-3.4402e-09	-3.4402e-09	6.8532e-13	0	0	0	-5.4616e-11	0.1380	-3.2848e-13
5	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
6	5.4456e-11	5.4456e-11	0	-6.8347e-13	6.8347e-13	0	0	0	0	3.2920e-13	3.5087e-13	0
7	1.7216e-24	1.3478e-19	3.2920e-13	1.0000	-1.4969e-18	-5.2296e-12	0	0	0	0	0	0
8	-1.3478e-19	2.0176e-37	3.5087e-13	1.4969e-18	1	5.2295e-12	0	0	0	0	0	0
9	-3.2920e-13	-3.5087e-13	0	5.2296e-12	-5.2295e-12	1.0000	0	0	0	0	0	0
10	0	0	0	-5.7590e-08	-5.7590e-08	-2.7287e-14	0	0	0	-2.1715	9.1429e-10	4.3301e-13
11	0	0	0	4.0369e-08	4.0369e-08	-2.1644e-14	0	0	0	6.4089e-10	-1.6192	-5.1363e-13
12	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 29: linsys.A

phi

matrice B

p

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0
4	-6.8995e-17	6.8995e-17	-6.8995e-17	6.8995e-17	-9.9331e-14	0	0	0	1.0000	1.4969e-18	5.2296e-12	0	0
5	6.7628e-17	-6.7628e-17	6.7628e-17	-6.7628e-17	9.8151e-14	0	0	0	-1.4969e-18	1.0000	-5.2295e-12	0	0
6	-0.0096	0.0096	-0.0096	0.0096	-8.2855	0	0	0	0	0	1.0000	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0.4202	0.4202	-0.4202	-0.4202	2.5288e-14	0	0	0	0	0	0	0	0
11	0.3133	-0.3133	-0.3133	0.3133	1.8856e-14	0	0	0	0	0	0	0	0
12	-0.0115	-0.0115	-0.0115	-0.0115	-7.9371e-16	0	0	0	0	0	0	0	0

Figura 30: linsys.B

phi

Matrice C

p

	1	2	3	4	5	6	7	8	9	10	11	12
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
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
3	5.4456e-11	5.4456e-11	0	0	0	0	0	0	0	0	0	0
4	0	5.5511e-12	0	0	0	0	0	0	0	0	0	0
5	-5.2296e-12	-5.2295e-12	-1.0000	0	0	0	0	0	0	0	0	0
6	1.4969e-18	1.0000	-5.2295e-12	0	0	0	0	0	0	0	0	0
7	0	8.3096e-30	1.0000	0	0	0	0	0	0	0	0	0
8	5.5511e-12	0	0	0	0	0	0	0	0	0	0	0
9	-1.0000	1.4969e-18	-5.2296e-12	0	0	0	0	0	0	0	0	0
10	0	-1.0000	0	0	0	0	0	0	0	0	0	0
11	1.0000	-2.9029e-23	0	0	0	0	0	0	0	0	0	0
12	5.5511e-12	5.5511e-12	0	0	0	0	0	0	0	0	0	0
13	1	0	0	0	0	0	0	0	0	0	0	0
14	0	1	0	0	0	0	0	0	0	0	0	0
15	0	0	1	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	9.0026e-06	0	0	0	0	0
17	0	0	0	0	0	0	0	1.2127e-05	0	0	0	0
18	0	0	0	0	0	0	0	0	-1.0000	0	0	0
19	0	0	0	0	0	0	0	0	0	1	0	0
20	0	0	0	0	0	0	0	0	0	0	1	0
21	0	0	0	0	0	0	0	0	0	0	0	1
22	0	0	0	1.0000	0	0	0	0	0	0	0	0
23	0	0	0	0	1	0	0	0	0	0	0	0
24	0	0	0	0	0	1.0000	0	0	0	0	0	0
25	1.7216e-24	1.3478e-19	3.2920e-13	1.0000	-1.4969e-18	-5.2296e-12	0	0	0	0	0	0
26	-1.3478e-19	2.0176e-37	3.5087e-13	1.4969e-18	1	5.2295e-12	0	0	0	0	0	0
27	-3.2920e-13	-3.5087e-13	0	5.2296e-12	-5.2295e-12	1.0000	0	0	0	0	0	0
28	0	0	0	0	0	0	1	0	0	0	0	0
29	0	0	0	0	0	0	0	1	0	0	0	0
30	0	0	0	0	0	0	0	0	1	0	0	0
31	0	0	0	-5.7590e-08	-5.7590e-08	-2.7287e-14	0	0	0	-2.1715	9.1429e-10	4.3301e-13
32	0	0	0	4.0369e-08	4.0369e-08	-2.1644e-14	0	0	0	6.4089e-10	-1.6192	-5.1363e-13
33	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 31: linsys.C

linsys.D

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	-6.8995e-17	6.8995e-17	-6.8995e-17	6.8995e-17	-9.9331e-14	0	0	0	1.0000	1.4969e-18	5.2296e-12	0	0	0
2	6.7628e-17	-6.7628e-17	6.7628e-17	-6.7628e-17	9.8151e-14	0	0	0	-1.4969e-18	1.0000	-5.2295e-12	0	0	0
3	-0.0096	0.0096	-0.0096	0.0096	-8.2855	0	0	0	0	0	1.0000	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0
29	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0
31	0.4202	0.4202	-0.4202	-0.4202	0	0	0	0	0	0	0	0	0	0
32	0.3133	-0.3133	-0.3133	0.3133	0	0	0	0	0	0	0	0	0	0
33	-0.0115	-0.0115	-0.0115	-0.0115	-1.3323e-15	0	0	0	0	0	0	0	0	0

Figura 32: linsys.D

Dunque effettuando il disaccoppiamento ingresso-uscita è stato possibile ricavarci il nostro sotto-sistema, tenendo in considerazioni le variabili ϕ, p, cioè l'angolo di *Roll*, e la velocità corrispondente. Prendendo le parti evidenziate in rosso nelle figure soprastanti, possiamo definire le seguenti matrici che caratterizzano il nostro sotto-sistema:

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 \\ 0 & -2.1715 \end{bmatrix} \qquad \boldsymbol{B} = \begin{bmatrix} 0 \\ 0.4202 \end{bmatrix} \qquad \boldsymbol{C} = \begin{bmatrix} 1 & 0 \end{bmatrix} \qquad \boldsymbol{D} = \begin{bmatrix} 0 \end{bmatrix} \quad (37)$$

Potremmo dunque ora scrivere la Funzione Di Trasferimento del sotto-sistema attraverso la seguente formula:

$$FdT = \boldsymbol{C} \cdot (s\boldsymbol{I} - \boldsymbol{A})^{-1} \cdot \boldsymbol{B} \quad (38)$$

Il sotto-processo ricavato dovrà essere moltiplicato per la costante $K_1 = 5.6659$, ricavata dalla matrice $Controller.Q2Ts$ relativa alla colonna $\tau Roll$ (sezione 3.4), la quale è responsabile della partizione del segnale di controllo; e per la costante $K_2 = 1530.7$, la quale rappresenta il coefficiente di spinta dei motori.

4.2 Linearizzazione di Olivieri Davide

Al fine di rendere lo schema del drone il più semplice possibile, è stato selezionato il processo lineare fornito da *Simulink* e il blocco *Enviroment* contenente tutte le variabile legate all'ambiente.
Sono state eliminate tutte i blocchi "secondari", riducendo al minimo indispensabile il sistema di controllo.

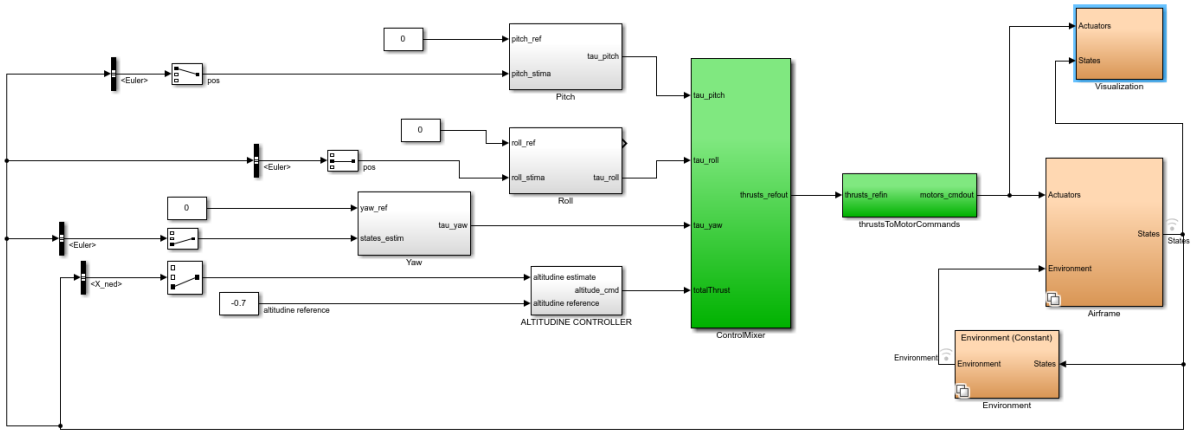


Figura 33: Schema di controllo semplificato e linearizzato

Il blocco *Airframe* rappresenta il processo linearizzato visto nella sezione 4.1, sono stati poi estrapolati dall'*FCS* i due blocchi :
ControloMixer, il quale prende i valori in uscita da tutti i controllori e li moltiplica per la matrice che gestisce la partizione del segnale di controllo;
ThurstsToMotorCommands che moltiplica il segnale di uscita per la costante di spinta dei motori in moda da regolare la direzione di rotazione, antioraria o oraria, delle eliche del drone.
Sono stati inseriti 4 controllori in spazio di stato, uno per l'angolo *Pitch*, uno per *Yaw*, uno per l'*Altitudine (Z)*, sintetizzati da colleghi negli anni precedenti, e uno per l'angolo di *Roll* che verrà trattato in dettaglio nelle sezioni successive.
E' possibile notare da blocco ausiliaro *Visualization*, come dando un segnale di riferimento costante a -0.7 [m] sull'asse *Z*, il drone si è stabilizza velocemente alla quota di riferimento.

5 Controllore PID

5.1 Cenni teorici sul PID

Lo schema mostrato in *Figura 34* è il classico schema in controreazione, di cui anche il nostro drone è dotato; consiste nel forzare la variabile controllata *y(t)* (l'uscita di un certo sistema da controllare) a seguire il più fedelmente possibile una variabile di riferimento *r(t)* generata dal *Signal Editor*. Il sistema di controllo acquisisce la misura della variabile da controllare tramite il blocco *Sensor* che fornisce una variabile di misura, proporzionale alla variabile misurata *y(t)*, quindi confronta tale valore con il riferimento *r(t)*, ottenendo una variabile errore *e(t)*. La variabile errore viene elaborata dal controllore per calcolare un opportuno valore per il segnale di controllo *v(t)*.

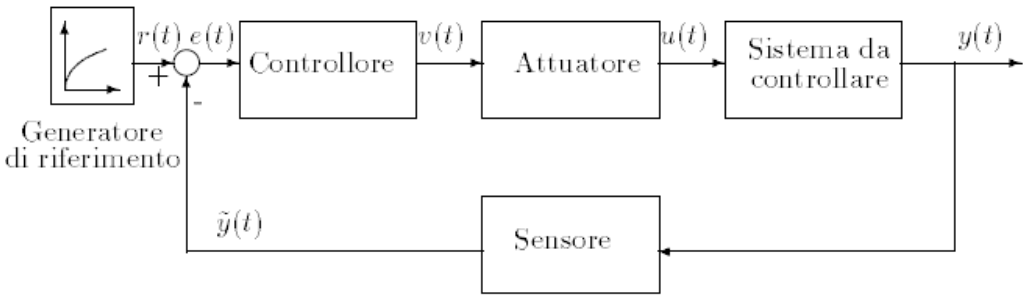


Figura 34: Sistema di controllo generale

Nelle applicazioni industriali sono spesso usati controllori con una struttura costituita da un termine proporzionale (P), un termine integrale (I) e uno derivativo (D). Tali controllori, detti regolatori standard o PID, sono particolarmente apprezzati per la loro semplicità ed efficacia.

Il **PID** regola l'uscita in base a:

- il valore del segnale di errore (azione proporzionale);
- i valori passati del segnale di errore (azione integrale);
- quanto velocemente il segnale di errore varia (azione derivativa).

Azione proporzionale (P) : ottenuta moltiplicando il segnale d'errore e con un'opportuna costante:

$$u_p = K_p \cdot e(t) \tag{39}$$

é perfettamente possibile regolare un processo con un simile controllore, che, in alcuni casi semplici, risulta anche in grado di stabilizzare processi instabili. Tuttavia, non è possibile garantire che il segnale d'errore "e" converga a zero: questo perchè un'azione di controllo "u" è possibile solo se "e" è diverso da zero.

Azione integrale (I) : è proporzionale all'integrale nel tempo del segnale di errore e, moltiplicato per la costante K_i:

$$u_i = K_i \cdot \int e(t)dt \tag{40}$$

Questa definizione dell'azione integrale fa sì che il controllore abbia memoria dei valori passati del segnale d'errore. Questa proprietà da al PID la capacità di portare il processo esattamente al punto di riferimento richiesto, dove la sola azione proporzionale risulterebbe nulla. E' bene fare attenzione a prevenire il fenomeno di WindUp, cioè continuare ad integrare l'errore quando la variabile di controllo è satura. Se l'azione integrale si sovraccarica per tornare ai valori normali ci vorranno lunghi transitori; è consigliato dunque adottare schemi anti WindUp come Questo il *tracking*, assicurandosi che l'integrale sia mantenuto ad un valore corretto quando l'attuatore si satura, in modo che il controller sia pronto a riprendere l'azione, non appena l'errore di controllo cambia.

Azione derivativa (I) : per migliorare le prestazioni del controllore si aggiunge l'azione derivativa:

$$u_d = K_d \cdot \frac{de(t)}{dt} \tag{41}$$

Quando "e" sta aumentando, l'azione derivativa cerca di compensare questa deviazione in termini della sua velocità di cambiamento, senza aspettare che l'errore diventi significativo (azione proporzionale) o che persista per un certo tempo (azione integrale). L'azione derivativa viene spesso tralasciata nelle implementazioni perchè subisce una brusca variazione nel momento in cui il riferimento venisse cambiato quasi istantaneamente da un valore a un altro, facendola tendere all'infinito. Una soluzione è quella di inserire il blocco derivativo in controreazione in modo da non derivare direttamente "e".

5.2 Test del controllore PID sul processo NON lineare

Nel blocco *FCS* è possibile trovare un sotto blocco chiamato *Logging*, inserendo dei dispositivi di *Scope*, come mostrato in *Figura 35* uno per ogni variabile, è stato possibile studiare il loro andamento per tutta la durata della simulazione.

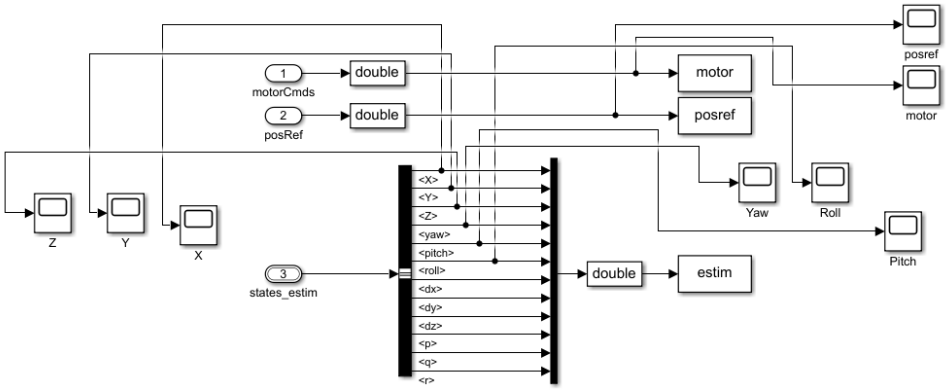


Figura 35: Blocco Logging

I risultati ottenuti sono i seguenti:

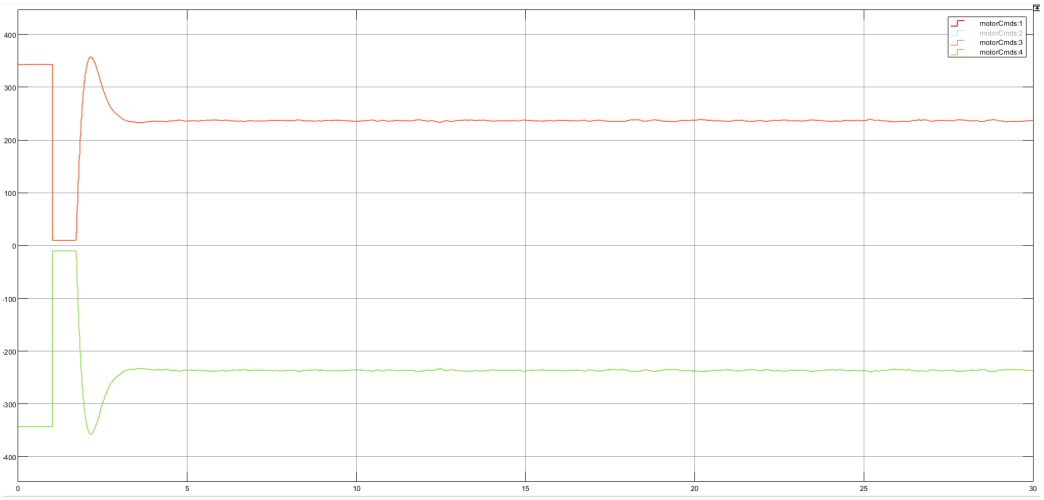


Figura 36: Andamento dei motori sul modello non lineare controllato da PID

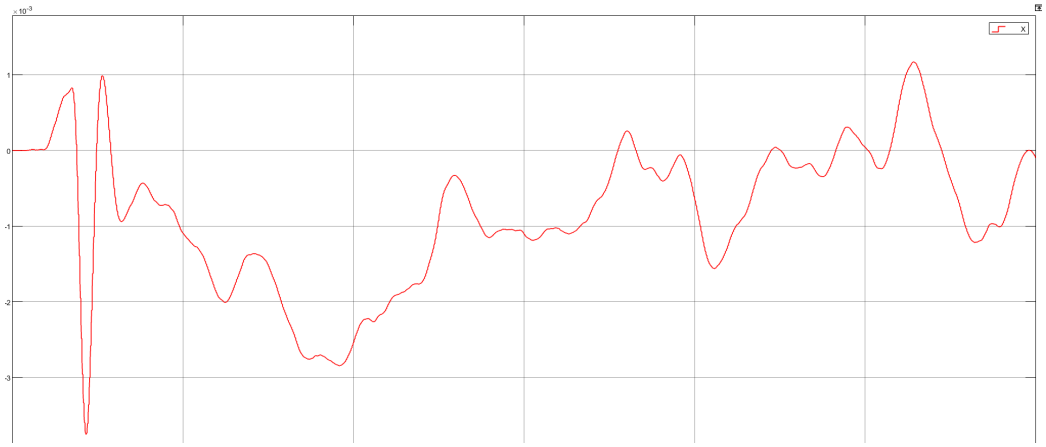


Figura 37: Andamento della variabile X sul modello non lineare controllato da PID

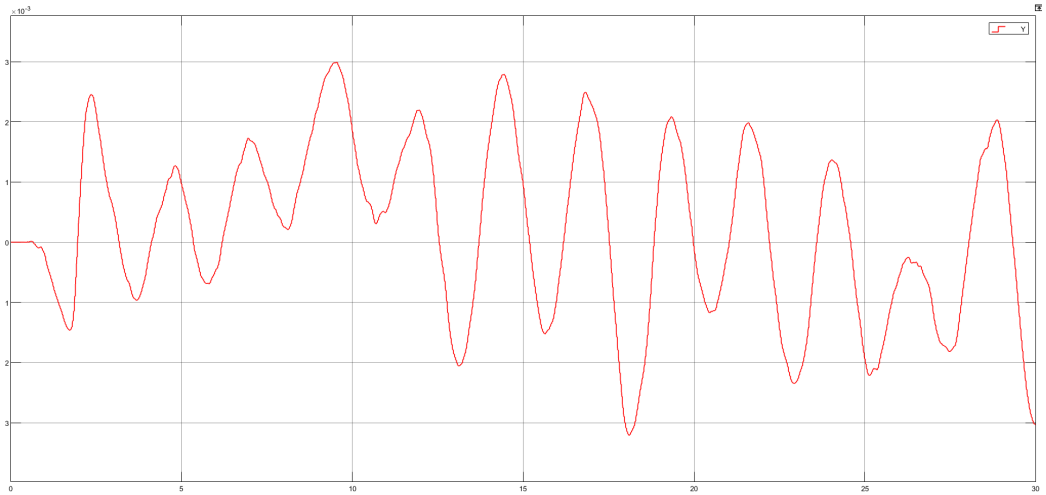


Figura 38: Andamento della variabile Y sul modello non lineare controllato da PID

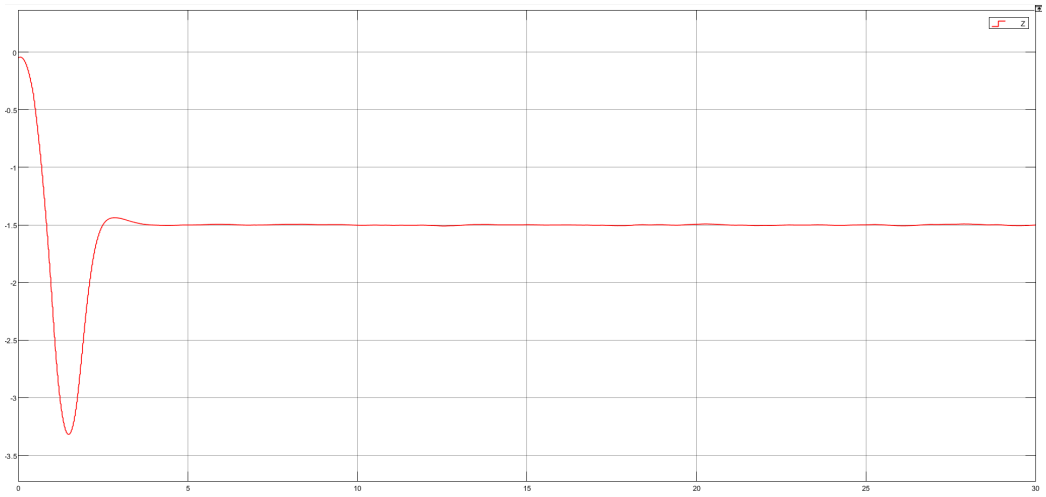


Figura 39: Andamento della variabile Z sul modello non lineare controllato da PID

Le variabili sulla posizioni X, Y risultano tutte controllate nell'intorno di 10^{-3}, mentre Z come mostra la *Figura 39*, segue il riferimento costante a -1.5 m dopo una sovraelongazione del tutto accettabile.

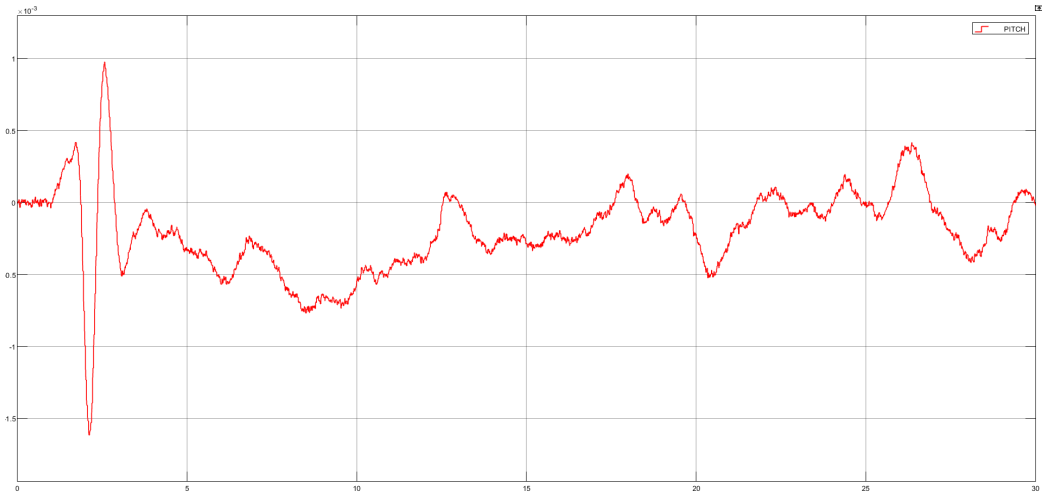


Figura 40: Andamento della variabile *Pitch* sul modello non lineare controllato da PID

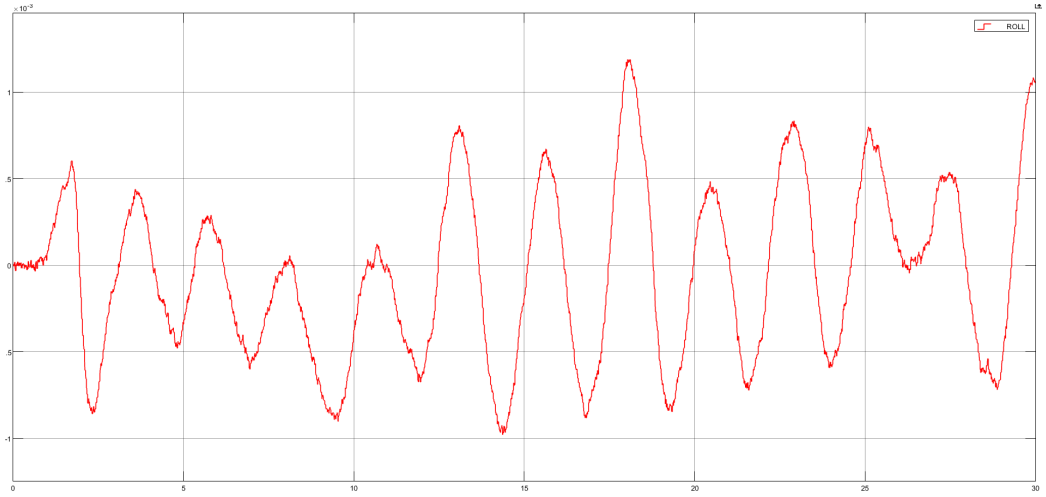


Figura 41: Andamento della variabile *Roll* sul modello non lineare controllato da PID

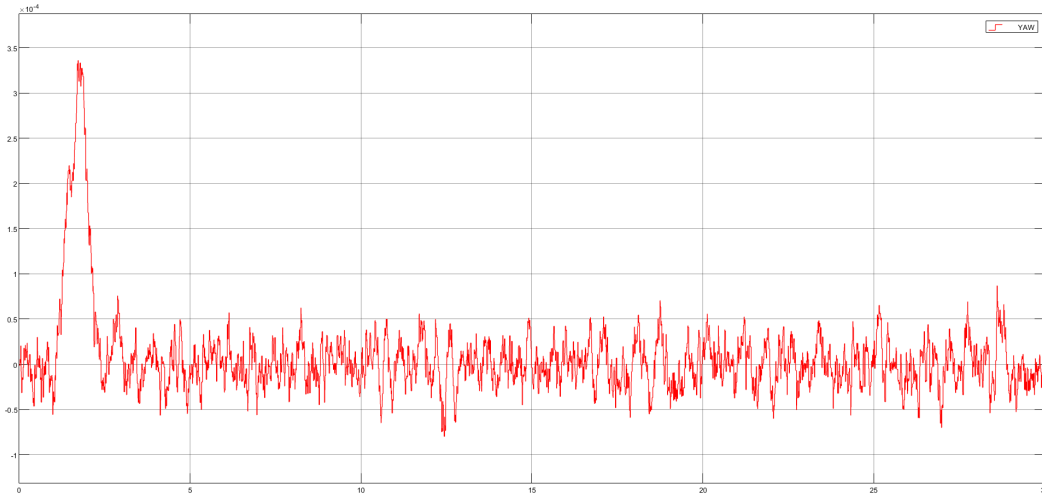


Figura 42: Andamento della variabile *Yaw* sul modello non lineare controllato da PID

Anche per quanti riguarda le variabili sull'orientamento, e quindi gli angoli di *Pitch, Roll*, risultano controllate nell'intorno di 10^{-3} mentre l'angolo di *Yaw* risulta ancora più preciso oscillando nell'intorno di 10^{-4}.

I grafici ci testimoniano come i PID realizzati dalla *Simulink* riescano a controllare tutte le variabili di stato in maniera molto precisa.

5.3 Test del controllore PID sul processo lineare

Digitando il comando $VSS\_VEHICLE = 0$, selezioneremo il processo linearizzato in maniera da testate il controllore PID anche su questo processo.

I risultati ottenuti sono i seguenti:

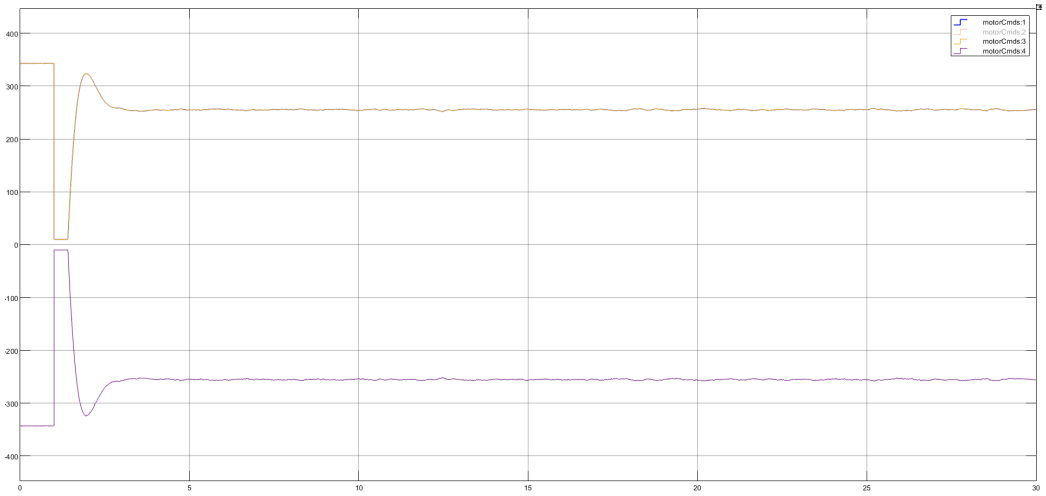


Figura 43: Andamento dei motori sul modello lineare controllato da PID

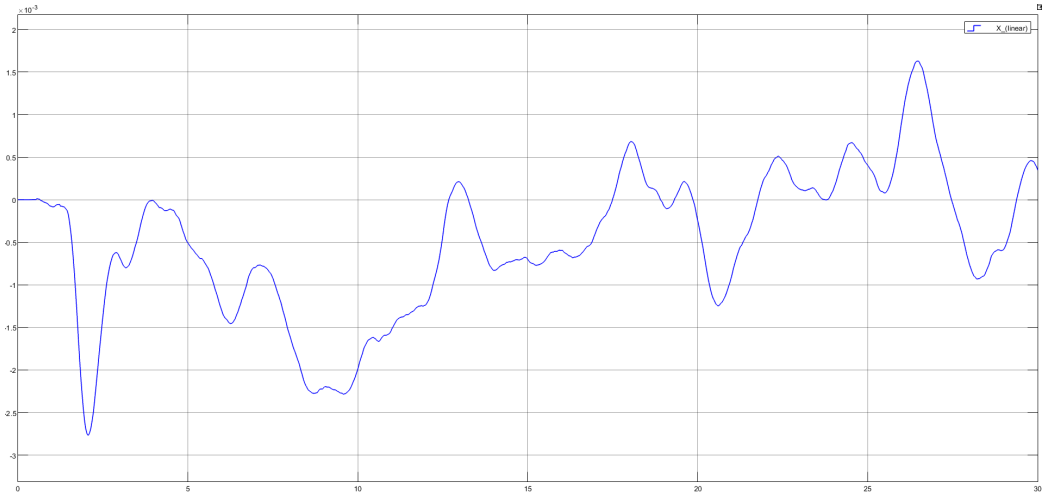


Figura 44: Andamento della variabile X sul modello lineare controllato da PID

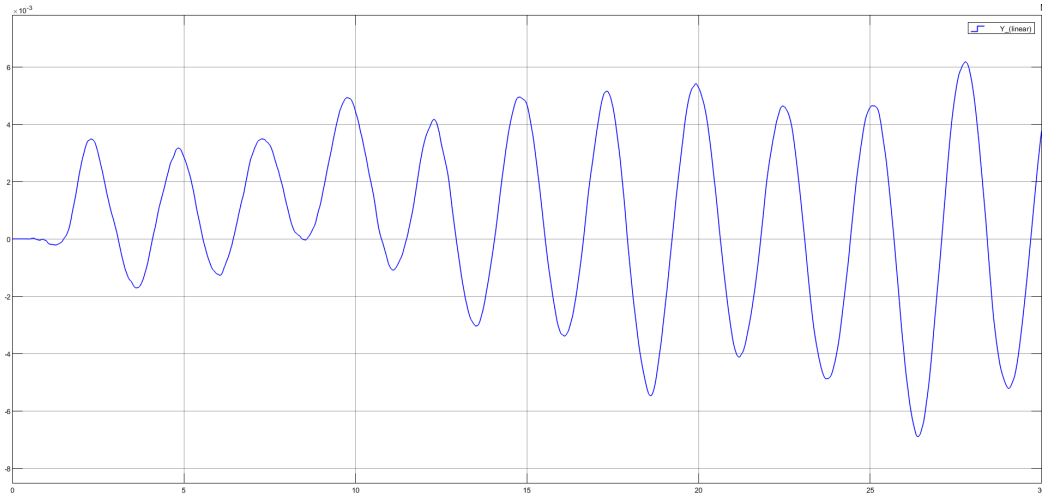


Figura 45: Andamento della variabile Y sul modello lineare controllato da PID

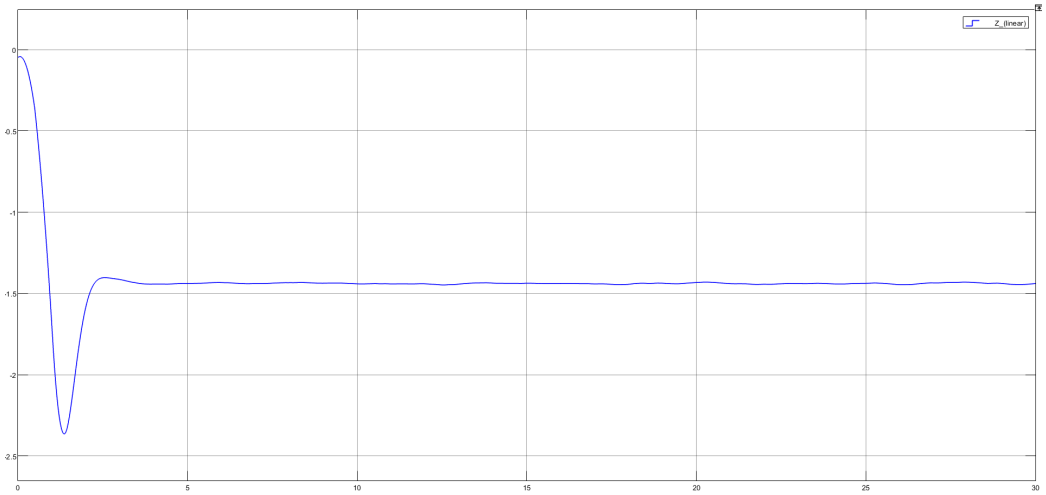


Figura 46: Andamento della variabile Z sul modello lineare controllato da PID

Le variabili sulla posizioni X, Y risultano tutte controllate nell'intorno di 10^{-3}, mentre Z come mostra la *Figura 46* segue il riferimento costante a -1.5 m, con un piccolo discostamento dal riferimento per tutta la simulazione, ciò è dovuto agli errori sulla linearizzazione del sistema. In definitiva i risultati sono del tutto comparabile al caso del processo non lineare (sezione 5.2).

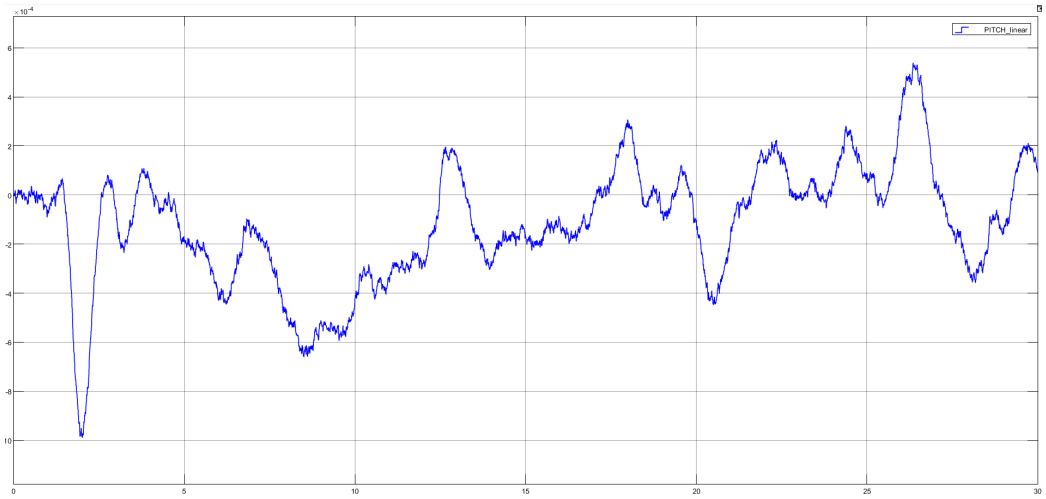


Figura 47: Andamento della variabile *Pitch* sul modello lineare controllato da PID

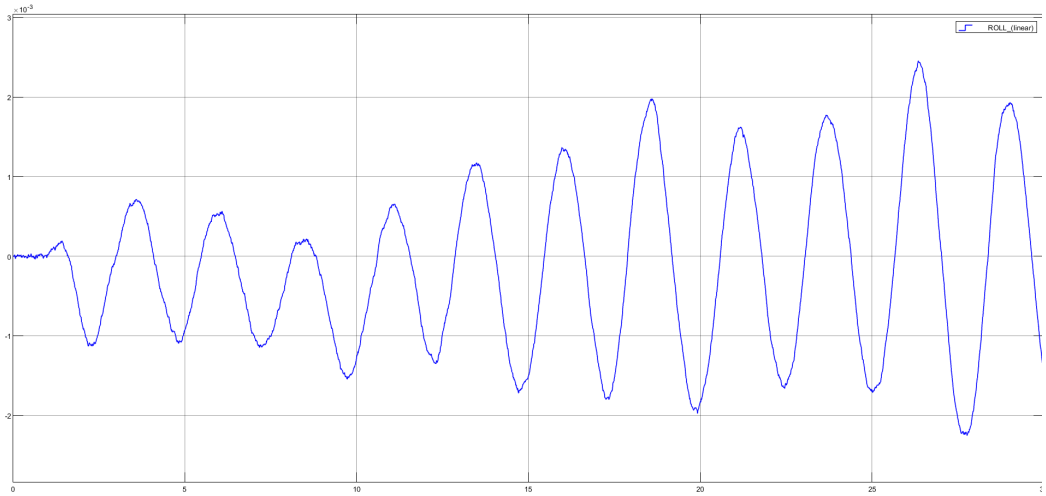


Figura 48: Andamento della variabile *Roll* sul modello lineare controllato da PID

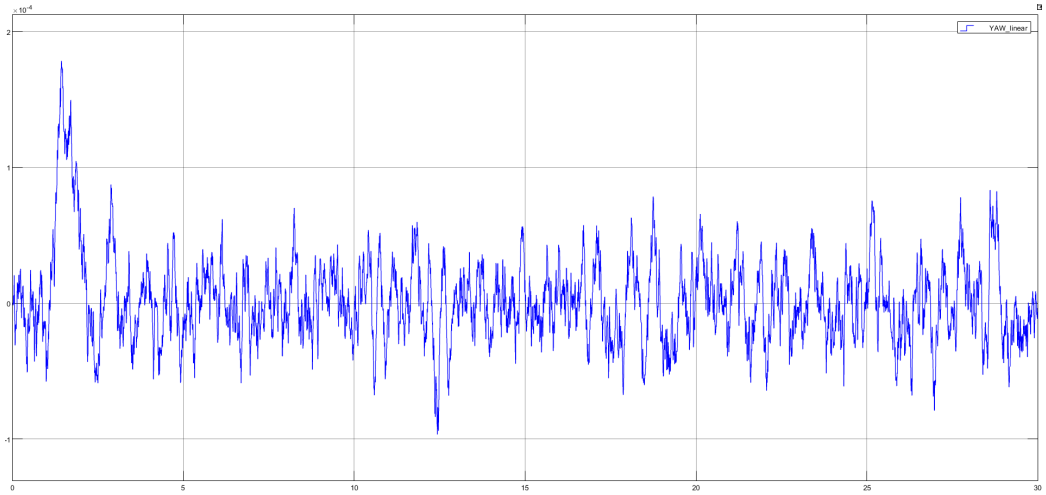


Figura 49: Andamento della variabile *Yaw* sul modello lineare controllato da PID

Anche per quanti riguarda le variabili sull'orientamento, e quindi gli angoli di *Pitch, Roll*, risultano controllate nell'intorno di 10^{-3} mentre l'angolo di *Yaw* risulta ancora più preciso oscillando nell'intorno di 10^{-4}. Anche in questa situazione i risultati sono analoghi a quelli del caso non lineare (sezione 5.2).

6 Sintesi per tentativi nel dominio della frequenza

Attraverso il file *sintesi-controllore.mat* presente nella cartella del progetto, è possibile vedere i passi da seguire per trovare la funzione di trasferimento in spazio di stato partendo dalle matrici della dinamica relative alla variabile di *Roll*. Il sotto-processo per il quale è stato sintetizzato il controllore è il seguente:

$$P(s) = \frac{3643}{s(s + 2.715)} \tag{42}$$

Nelle sintesi per tentativi si cerca di soddisfare le specifiche del sistema a ciclo chiuso assegnando la struttura della funzione di trasferimento in catena aperta con un'oppurtuna scelta del controllore.
Le specifiche a ciclo chiuso vengono impostate indirettamente e devono successivamente essere controllate con strumenti opportuni quali i diagrammi di Bode e la carta di Nicholos.
In particolare nella sintesi nel dominio della frequenza si mettono in relazione proprietà della *FDT* in catena aperta *F(s)* con quella in catena chiusa *W(s)* e si sceglie il controllore *G(s)* che modifica le caratteristiche in frequenza della funzione *W(jw)*.
Si definiscono le caratteristiche desiderate dalla W(s) che si divino in :

specifiche univoche :

1. *sistema tipo 1*: il processo P(s) ha già un polo in s = 0 allora il nostro sistema è già di tipo 1 non è dunque necessario aggiungere alcuno integratore al controllore G(s);

2. $\tilde{e} \leq 0.01$: l'errore in regime permanente di un sistema di tipo 1 è dato dalla seguente formula: $\tilde{e} = \frac{k_d^2}{k_g * k_p} \leq 0.01$, dove k_p è il guadagno statico del processo e k_g quello del controllore.

specifiche lasche :

1. $B_3 = 2 \quad Hz$: la *banda passante* (B_3) è un paramentro a ciclo chiuso corrispondente all'intersezione del luogo a costate a -3dB con il diagramma di Nichols, da cui si ricava il rispettivo parametro a ciclo aperto attraverso la legge empirica $W_t = (3 \div 5) * B_3 \rightarrow W_t = 10 \quad r/s$.

2. $M_r \leq 2 \quad dB$: il *modulo di risonanza* (M_r) anch'esso è un parametro a ciclo chiuso, dunque sarà trasformato in parametro a ciclo aperto,

(m_φ = margine di fase), attraverso la relazione:
$m_\varphi \geq 180 - fase(F(j10)) \geq 47°$

Iniziamo a progettare un controllore di primo tentativo che rispetti innanzitutto le specifiche univoche calcolando il guadagno statico del controllore : $k_g = \frac{1}{0.01 \cdot 3643} = 0.0274$
dunque il nostro controllore di primo tentativo sarà:
$G_1(s) = 0.0274$
Utilizzando il comando di Matlab "*sisotool*" è possibile verificare attraverso i diagrammi di Bode il rispetto delle specifiche in catena aperta della funzione: $F(s) = G_1(s)P(s)$.

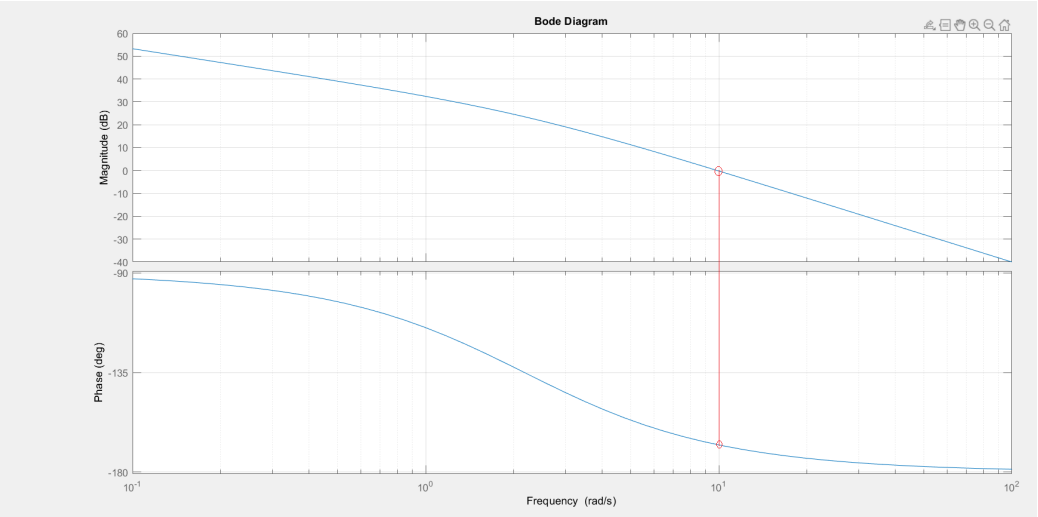


Figura 50: Diagrammi di bode F(s)

E' possibile notare dalla figura che in corrispondenza della pulsazione di attraversamento $w_t = 10 \quad r/s$, il modulo risulta circa 0 dB mentre la fase risulta essere -168°, ciò corrisponde ad un $m_\varphi = 12$, dovremmo dunque progettare una rete anticipatrice $R_a(s)$ che fornisca un aumento di fase di almeno 35° cercando di mantenere il modulo pressochè inalterato.
E' buona norma aumentare la fase di una quantità superiore a quella strettamente necessaria perché si deve tener conto della rete attenuatrice $R_i(s)$ che inevitabilmente introdurrà una diminuzione della fase.
La rete anticipatrice ha la seguente funzione di trasferimento:

$$R_a(s) = \frac{1 + \tau_a \cdot s}{1 + \frac{\tau_a}{m_a} \cdot s} \tag{43}$$

si è deciso dunque di scegliere:
$(w_t * \tau_a) = 2 \qquad w_a = 5 \qquad m_a = 16$
In modo tale da avere un aumento di fase di 57° per garantire la specifica sul margine di fase ma avremo un corrispondente aumento di modulo di 7dB,che verrà successivamente attenuato dalla rete $R_i(s)$.

$$R_a(s) = \frac{1 + 0.2s}{1 + 0.0125s} = \frac{s + 5}{s + 80} \tag{44}$$

A tal proposito si deve progettare una rete attenuatrice $R_i(s)$ che mi faccia diminuire il modulo di 7dB in corrispondenza della $w_t = 10 \quad r/s$, senza che la fase scenda al di sotto del valore minimo consentito dalla specifica sul margine di fase.
Quando si progetta un controllore che comprende sia la rete anticipatrice che quella attenuatrice staremo utilizzando una rete a sella, generalmente è buona norma posizionare $R_a(s)$ una decade prima della pulsazione di attraversamento e $R_i(s)$ una decade dopo w_t.
La rete attenuatrice ha la seguente funzione di trasferimento :

$$R_i(s) = \frac{1 + \frac{\tau_i}{m_i} \cdot s}{1 + \tau_i \cdot s} \tag{45}$$

Per i motivi sopra citati si è deciso di scegliere
$(w_t * \tau_i) = 100 \qquad w_i = 0.1 \qquad m_i = 2$
In maniera tale da avere una diminuzione del modulo di -7dB e una corrispondente diminuzione di fase di circa 1°.

$$R_i(s) = \frac{1 + 0.2s}{1 + 0.0125s} = \frac{s + 0.2}{s + 0.1} \tag{46}$$

Il controllore così ottenuto $G(s) = G_1(s)R_a(s)R_i(s)$ è stato valutato con gli strumenti disponibili su Matlab,grazie al comando *sisotool*, in catena aperta con il processo $P(s)$.

$$G(s) = \frac{0.0271(s + 0.2)(s + 5)}{s(s + 0.1)(s + 100)} \tag{47}$$

$$F(s) = G(s)P(s) = \frac{3643(s + 0.2)(s + 5)}{s(s + 0.1)(s + 100)} \tag{48}$$

Come si nota dalla figura sottostante il modulo di $F(j10)$ è pari a 0 dB mentre la fase risulta essere -112°.

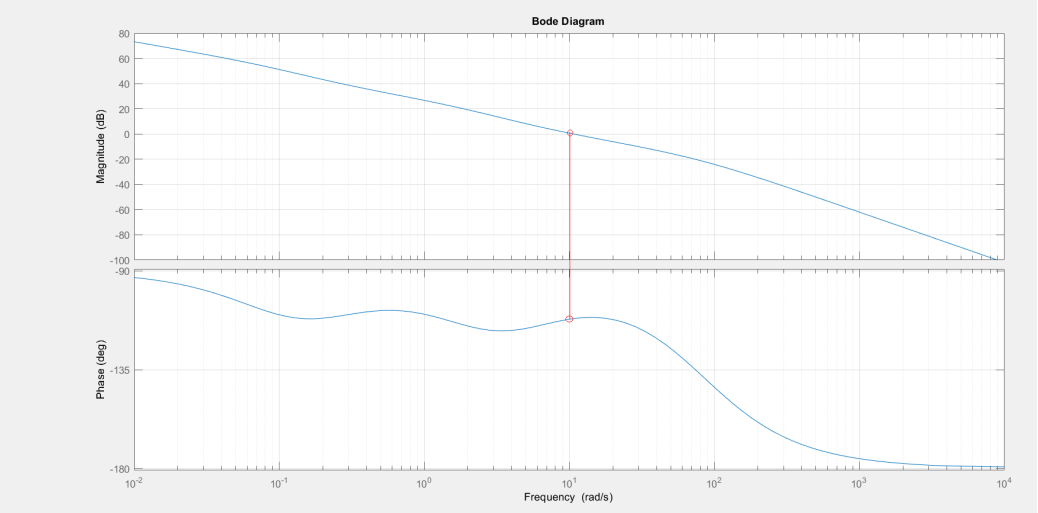


Figura 51: Diagrammi di bode $F(s) = G(s)P(s)$

In questa maniera avremo un $m_\varphi = 180 - 112 = 68°$ in corrispondenza della pulsazione di attraversamento $w_t = 10$.
Si è scelto un margine di fase più grande rispetto a quello descritto nella specifica in modo tale che il controllore sia più pronto a rispondere alle sollecitazioni esterne.

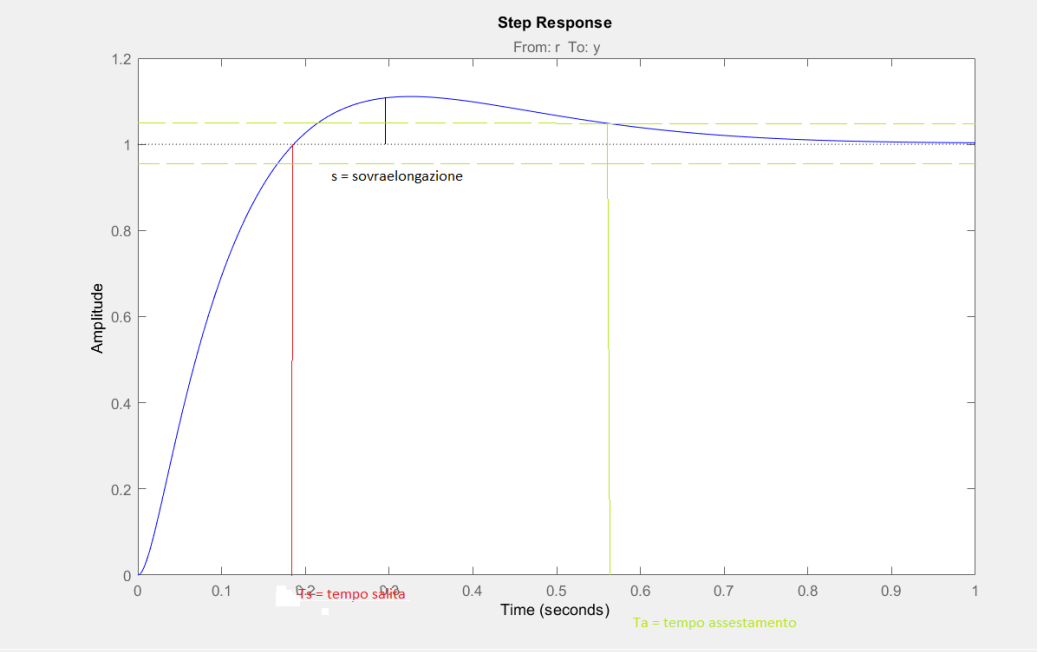


Figura 52: Risposta nel dominio del tempo

Per quanto riguarda la risposta nel dominio del tempo, presente in figura *52*, da notare la banda passante B_3 alla quale è legato il tempo si salita T_s dalla relazione empirica:
$B_3 T_s = 3$, come notiamo dal grafico $T_s = 0.18s \rightarrow B_3 = \frac{3}{0.17} = 2.8Hz$; e anche la prima specifica lasche è rispettata e il nostro sistema è dotato di prontezza.
Oltre al tempo di salita definito come, il tempo necessario affinchè la risposta raggiunga il valore di regime, altre caratteristiche presenti sono il tempo di assestamento, $T_a = 0.57s$, definito come il tempo necessario perché la risposta entri in una fascia di ampiezza ϵ attorno al valore di regime, corrisponde ad un comportamento poco oscillatorio; infine troviamo la sovraelongazione S, cioè il massimo valore assunto dalla risposta, normalizzato rispetto al valore di regime.

Un ulteriore analisi è stata fatta sul *criterio ridotto di Nyquist* il quale afferma che : nel caso in cui il sistema in catena aperta sia stabile,condizione necessaria e sufficiente affinchè il sistema a ciclo chiuso sia stabile è che il diagramma polare completo della funzione $F(jw)$, mostrato in *Figura 53*, non compia alcun giro attorno al punto critico $(-1, j0)$.

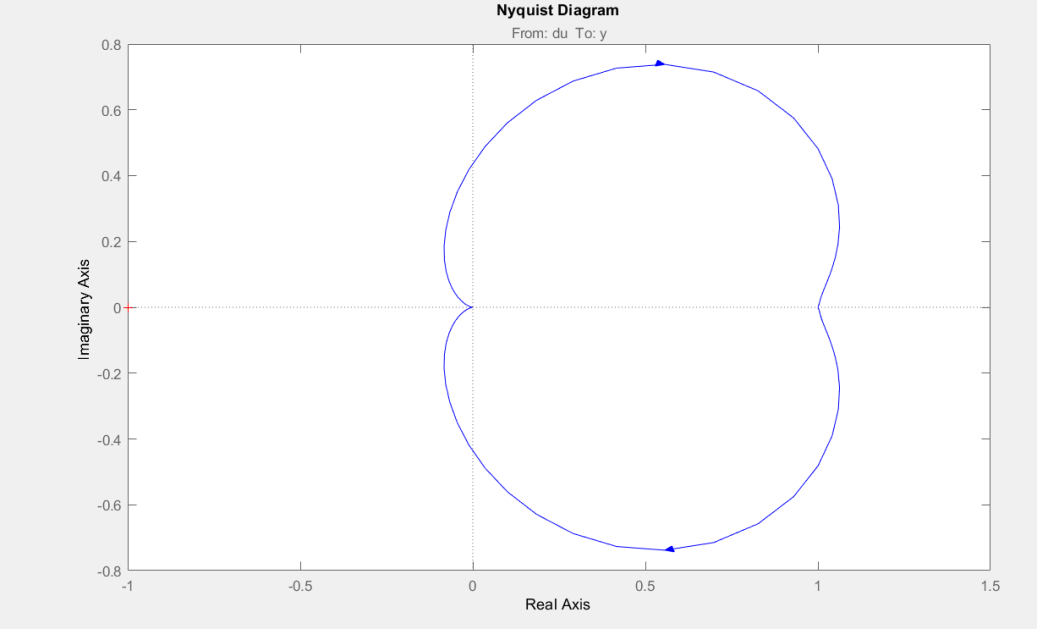


Figura 53: Driagramma di Nyquist F(s)

Una volta verificato il soddisfacimento delle specifiche sia a ciclo aperto che a ciclo chiuso, si è passati all'implementazione del controllore relativo alla variabile *Roll*.
Seguendo il percorso *FCS -> Controller -> Attitude* è stato sufficiente effettuare un disaccoppiamento delle variabili di *Pitch* e *Roll* attraverso un demultiplexer, in maniera tale che l'angolo di *Pitch* continui ad essere controllato dal PID fornito dal modello Simulink, mentre l'angolo di *Roll* sarà gestito dal controllore $G(s)$ precedentemente sviluppato attraverso la sintesi in frequenza.

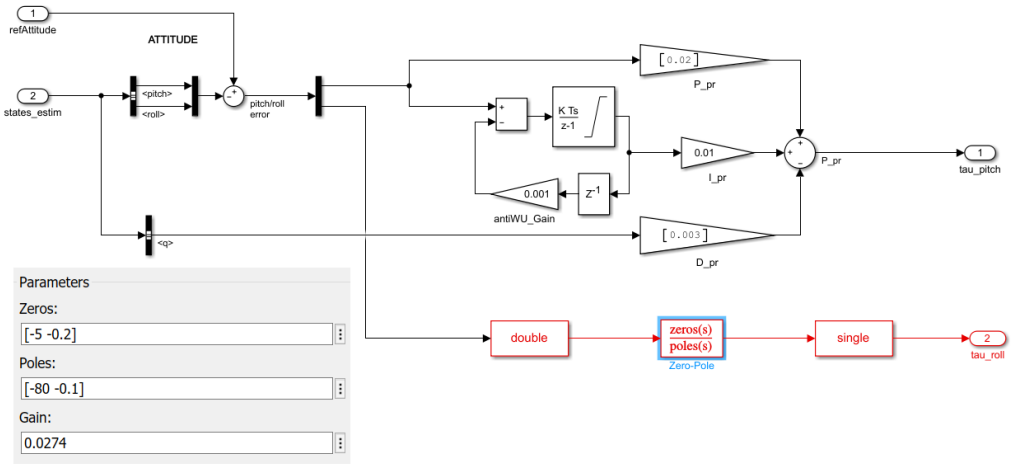


Figura 54: Controller Attitude : *pitch* e *roll*

Il nuovo modello del controller, mostrato in *Figura 54*, è stato compilato e attraverso il blocco ausiliario *VisualizationSimulink3D* è possibile controllare l'andamento del drone per tutta la durata della simulazione ($T = 30s$), attraverso una fotocamera isometrica.

6.1 Test del controllore G(s) sul processo non lineare

Attraverso gli strumenti di Scope inseriti precedentemente nel blocco *Logging* presente nel *FCS* sono stati effettuati i test sul nuovo controllore per poi confrontarli con quelli ottenuti attraverso il controllo PID.
I risultati ottenuti sono i seguenti:

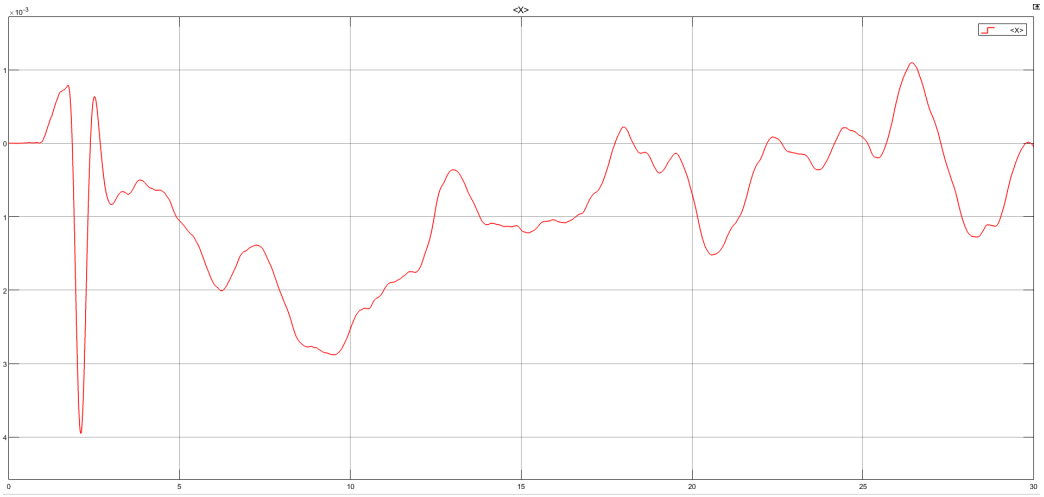


Figura 55: Andamento variabile X sul modello non lineare controllato da G(s)

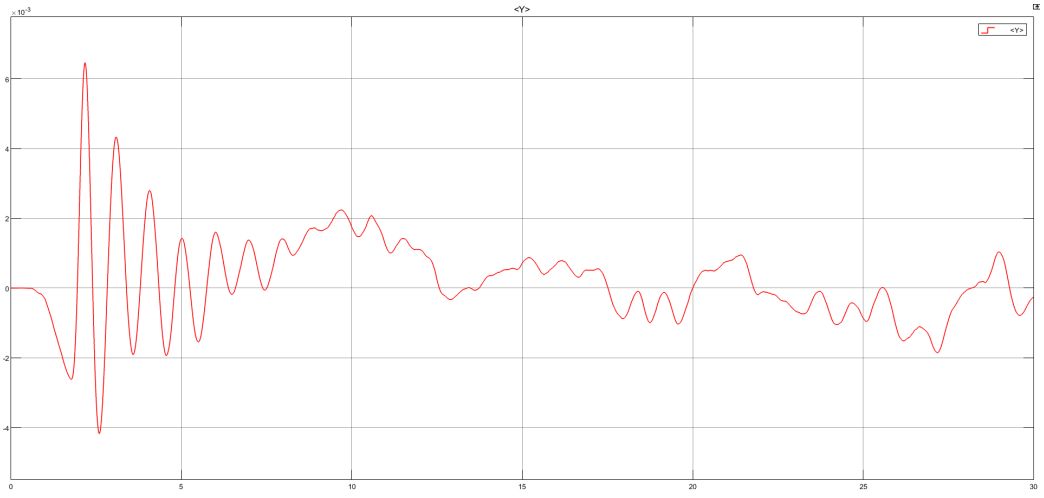


Figura 56: Andamento variabile Y sul modello non lineare controllato da G(s)

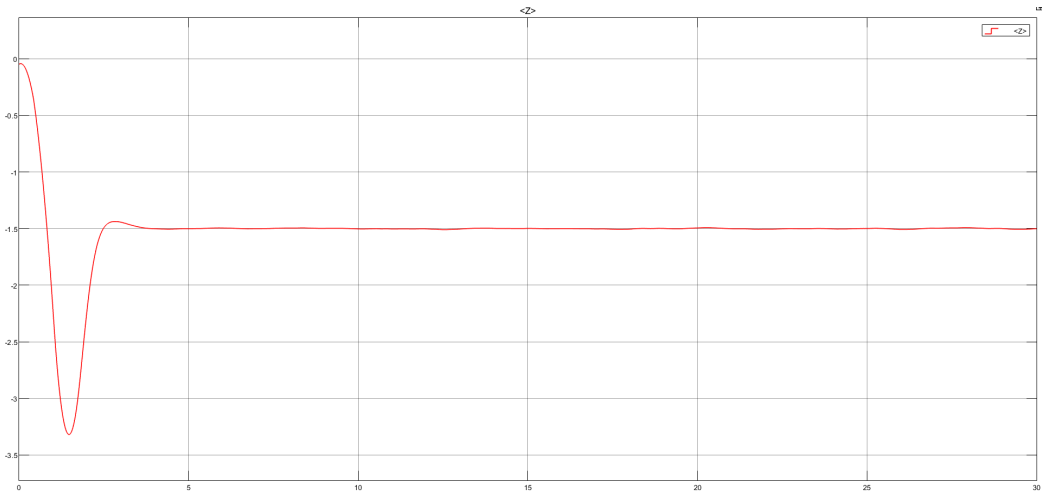


Figura 57: Andamento variabile Z sul modello non lineare controllato da G(s)

Per quanta riguarda i tracciati delle tre variabili *(X,Y,Z)* relative alla posizione del quadricottero i loro andamenti sono del tutto comparabili a quelli simulati nella sezione 5.2 ,in cui il controllore per l'angolo di Roll era un PID.
L'unica che risulta essere leggermente diversa è la variabile Y, in *Figura 56*, in cui nei primi $5s$ della simulazione risulta avere delle oscillazioni più ampie, sempre nell'ordine di 10^{-3}, che si attenuano subito dopo.

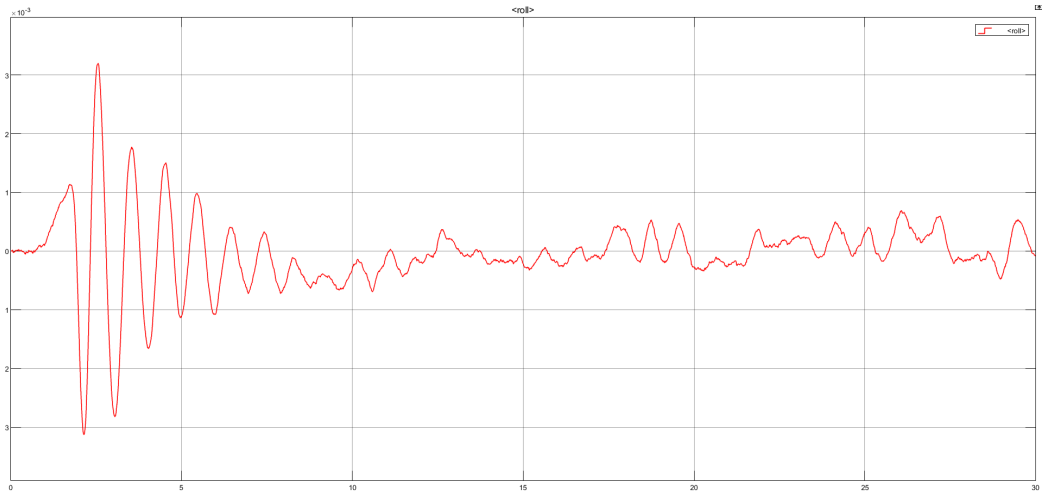


Figura 58: Andamento variabile *ROLL*, sul modello non lineare, con riferimento variabile

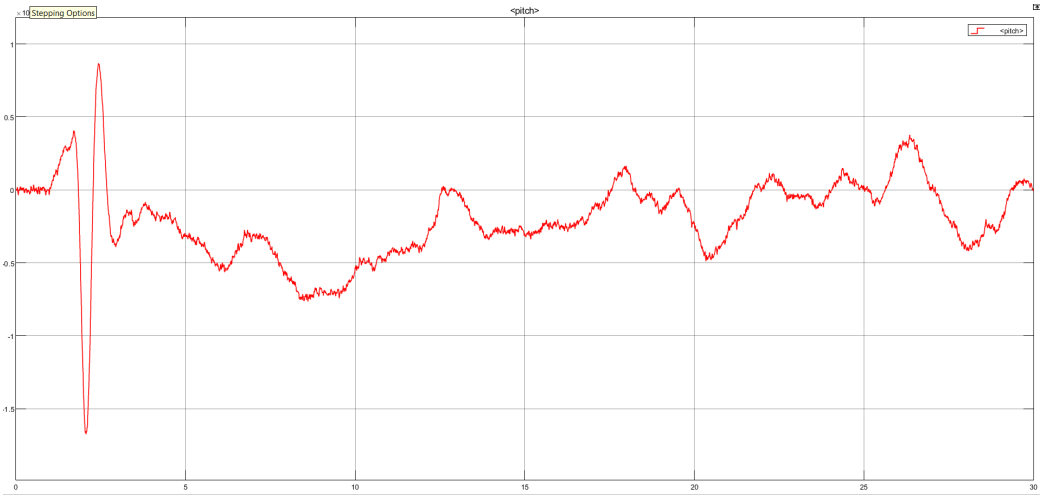


Figura 59: Andamento variabile *PITCH*, sul modello non lineare, con riferimento variabile

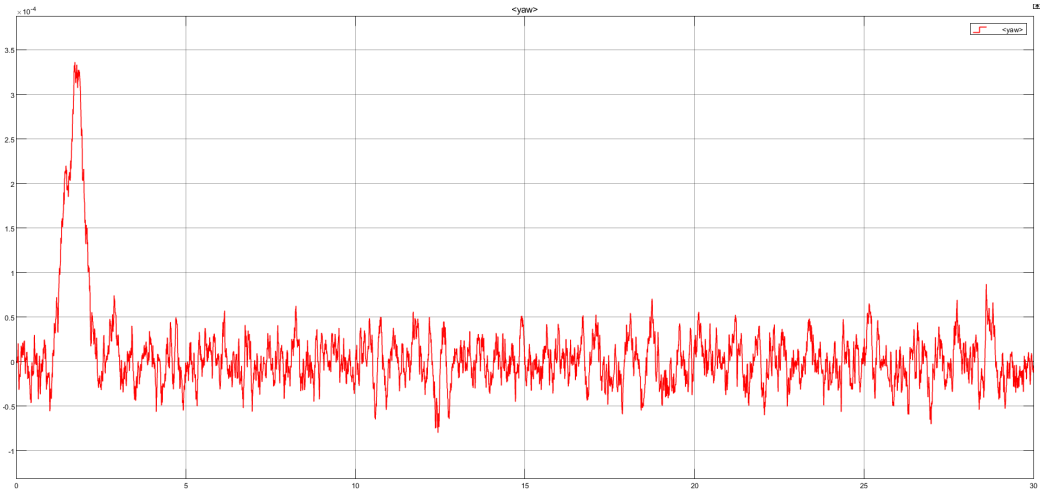


Figura 60: Andamento variabile *YAW*, sul modello non lineare, con riferimento variabile

Anche i tracciati delle variabili sull'orientamento del drone *Roll, Pitch, Yaw*, sono del tutto analoghe a quelle viste nella sezione 5.3.
Si può notare, nella *Figura 58*, come l'angolo di *Roll* inizialmente presenti delle oscillazioni maggiori rispetto al caso in cui veniva utilizzato il PID, ma che poi verranno attenuate,e la variabile risulti controllata in maniera molto più efficiente per tutta la durata della simulazione.

6.2 Test del controllore G(s) sul modello lineare del processo

E' stato valutato il controllore *G(s)* anche sul modello lineare del processo fornito da *Simulink*.
I risultati sono mostrati nelle figure sottostanti:

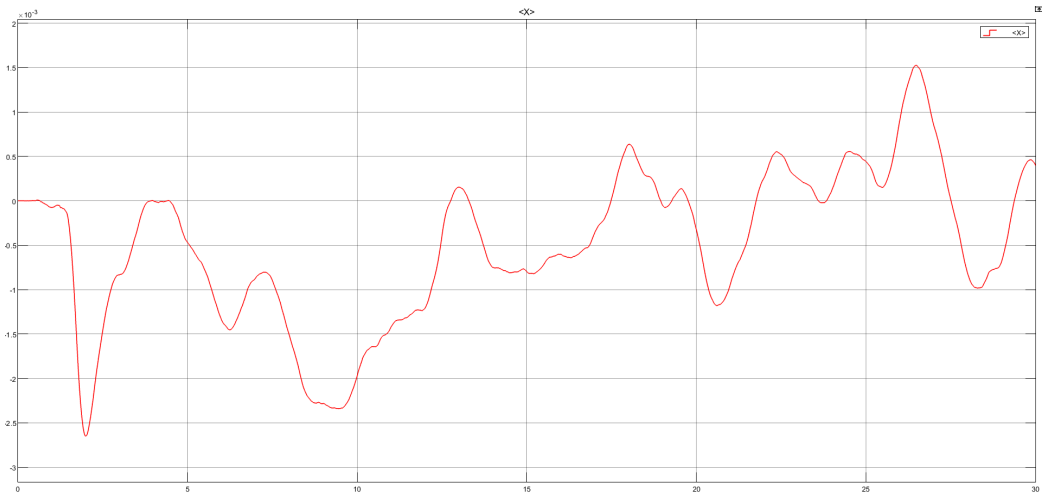


Figura 61: Andamento variabile X, sul modello lineare, con riferimento variabile

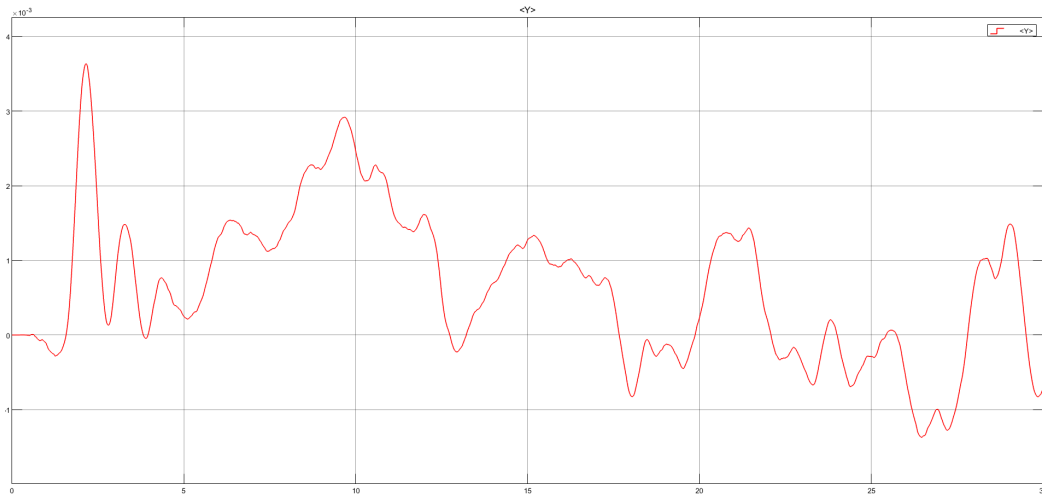


Figura 62: Andamento variabile Y, sul modello lineare, con riferimento variabile

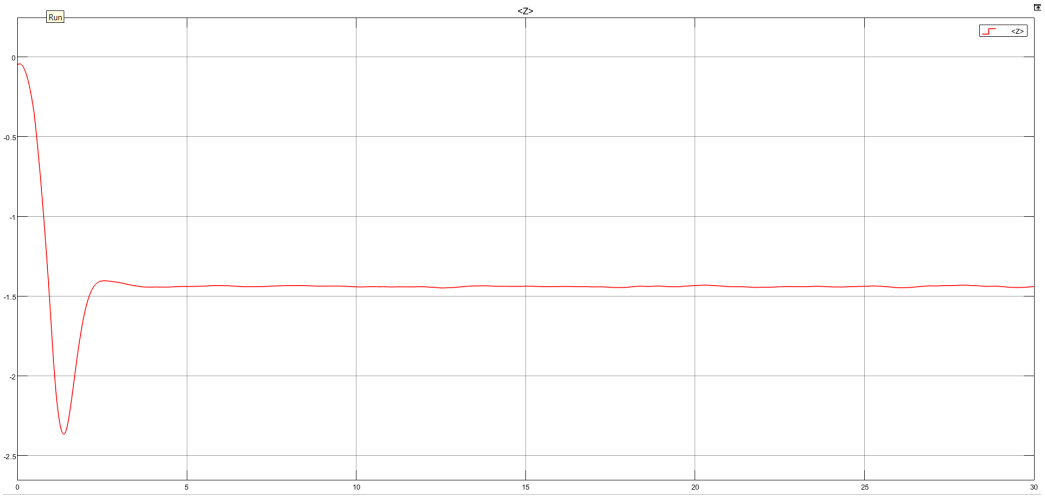


Figura 63: Andamento variabile Z, sul modello lineare, con riferimento variabile

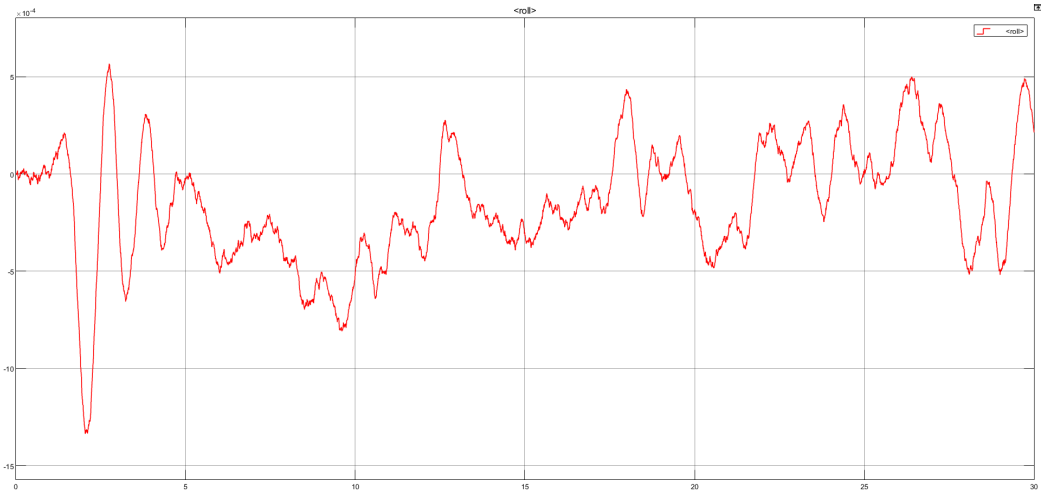


Figura 64: Andamento variabile $ROLL$, sul modello lineare, con riferimento variabile

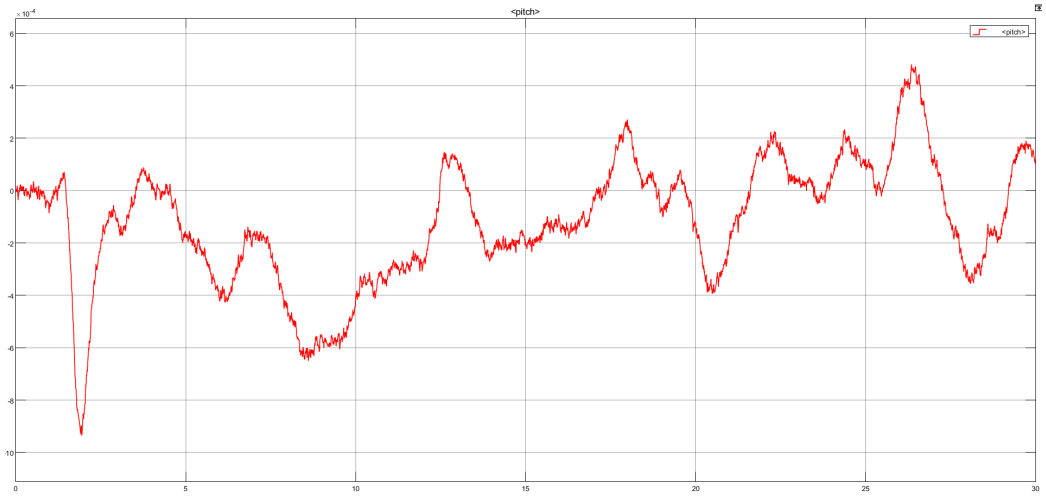


Figura 65: Andamento variabile *PITCH*, sul modello lineare, con riferimento variabile

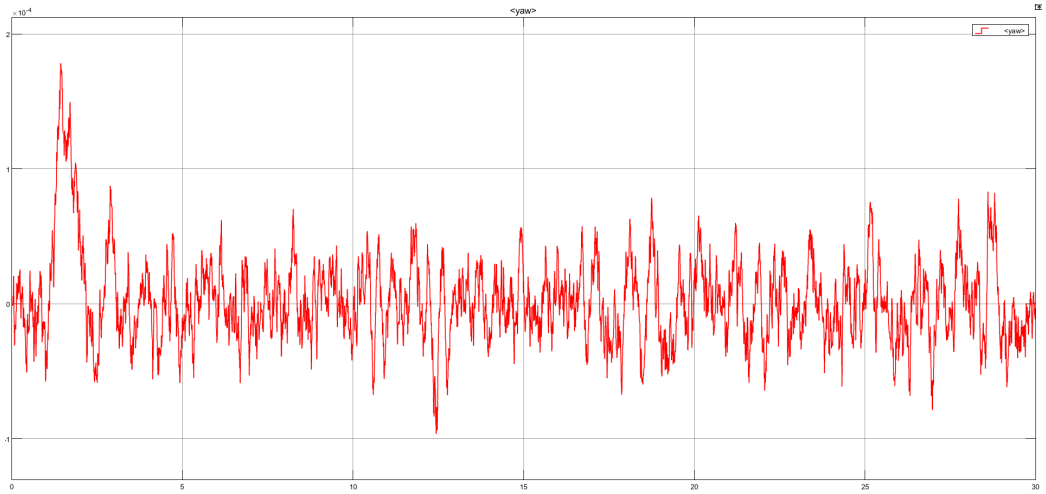


Figura 66: Andamento variabile *YAW*, sul modello lineare, con riferimento variabile

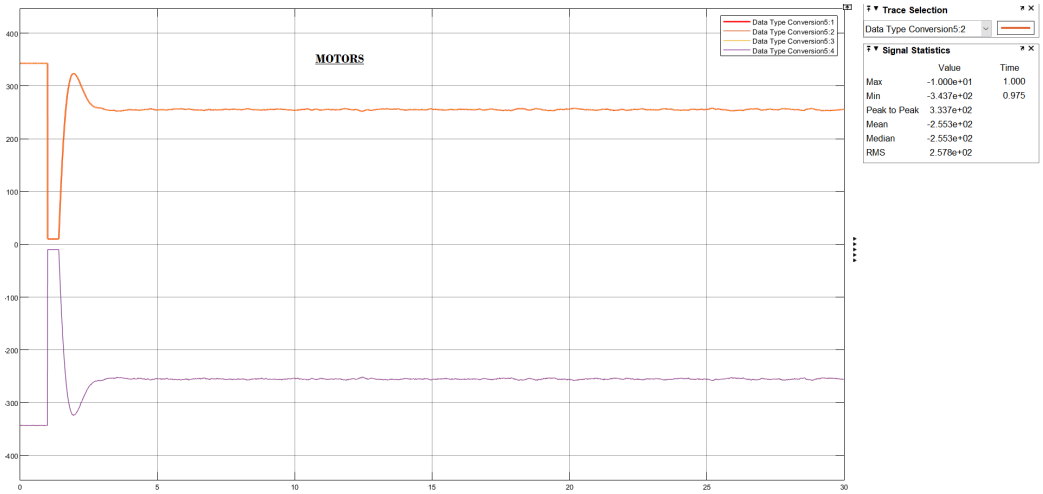


Figura 67: Tracciato dei motori, sul modello lineare, con riferimento variabile

Anche per quanto riguarda il modello lineare, tutte le variabili controllate hanno l'andamento analogo ai casi precedenti.
Si può notare anche questo caso, nella *Figura 64*, come l'angolo di *Roll* risulti controllato in modo più efficiente con oscillazioni nell'ordine di 10^{-4}.

7 Conclusioni

Il nostro obiettivo era quello di sintetizzare un controllore per il sotto-processo relativo all'angolo di *Roll* che migliorava le prestazioni del drone rispetto ai PID utilizzati dalla *Simulink*.
E' stato analizzato nelle sezioni precedenti che i test effettuati sul controllore PID e sul controllore *G(s)* sono del tutto analoghi.
Concentrandoci ora sui risultati ottenuti riguardo la variabile di *Roll*, e confrontando i risultati è possibile notare alcuni miglioramenti che ci permettono di affermare il successo del progetto.
Comprovata l'affidabilità del controllore attraverso la simulazione, si può generato il codice *C++* relativo al modello *Simulink* e lo si può è caricare sul drone attraverso il cavo USB.

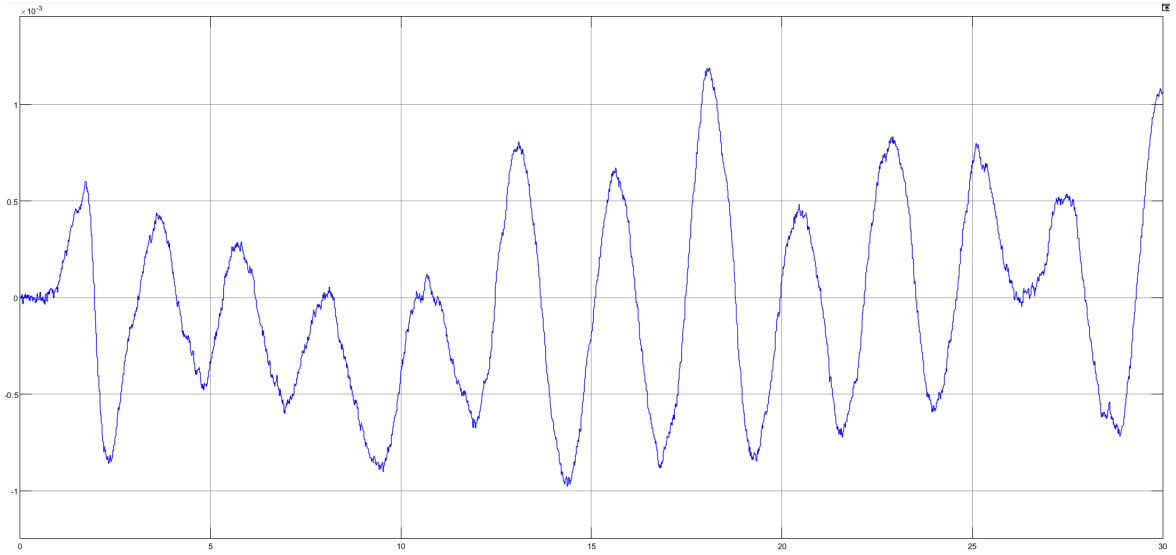


Figura 68: Tracciato *Roll* controllato da PID

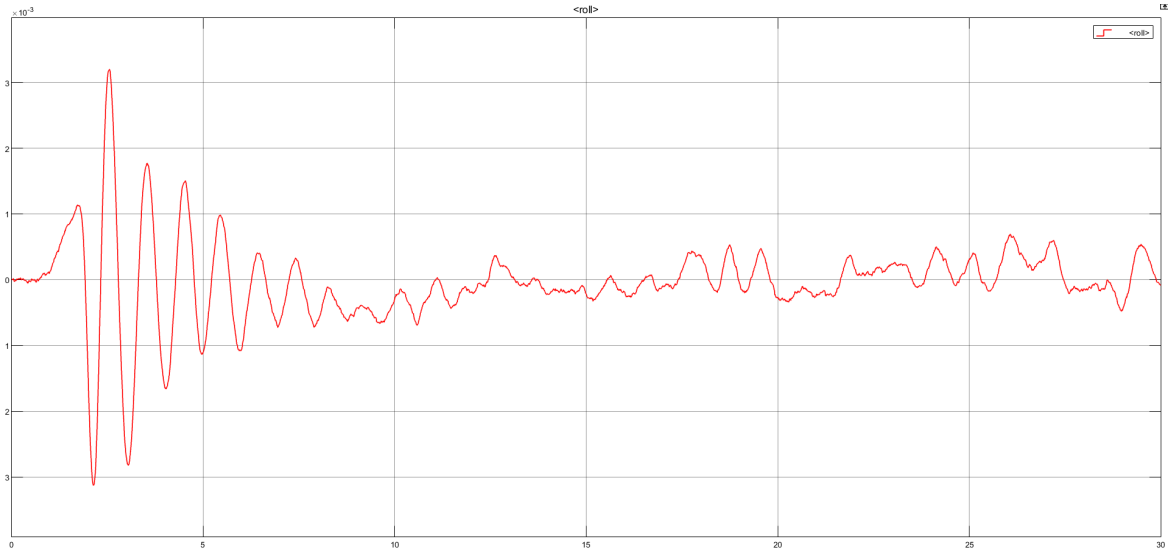


Figura 69: Tracciato *Roll* controllato da *G(s)*

E' possibile notare dalla figura come l'andamento della variabile *Roll* controllata dal controllore *G(s)* (in *rosso*), sintetizzato in precedenza, risulti avere oscillazioni più ampie nei primi 5 s, ma che si attenuano in maniera molto più efficacie per tutta la durata della simulazione, rispetto al controllo PID (in *blu*).

Tecniche a confronto nel caso lineare

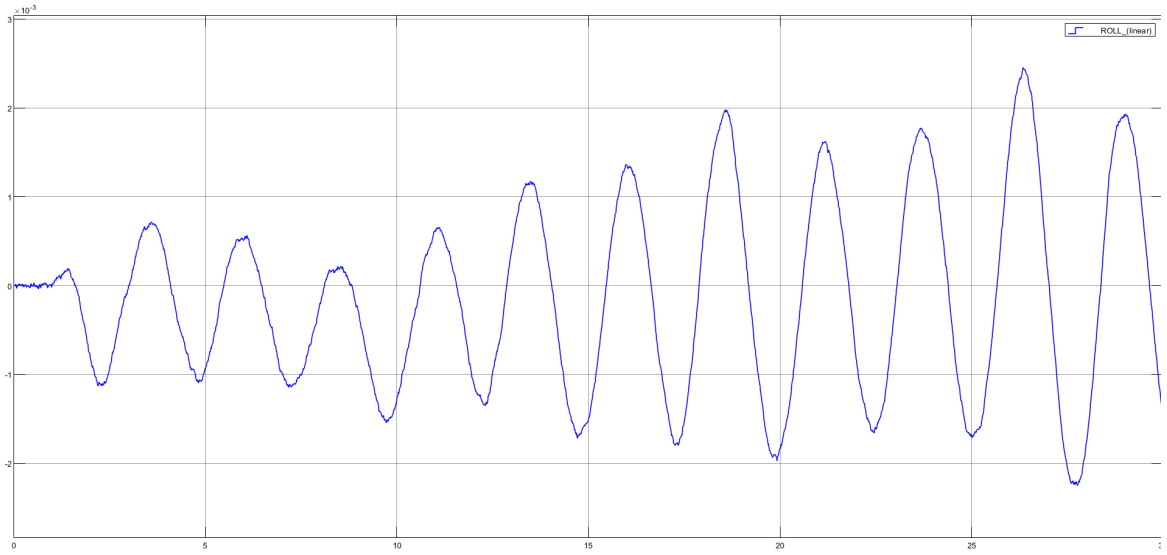


Figura 70: Tracciato *Roll* controllato da PID

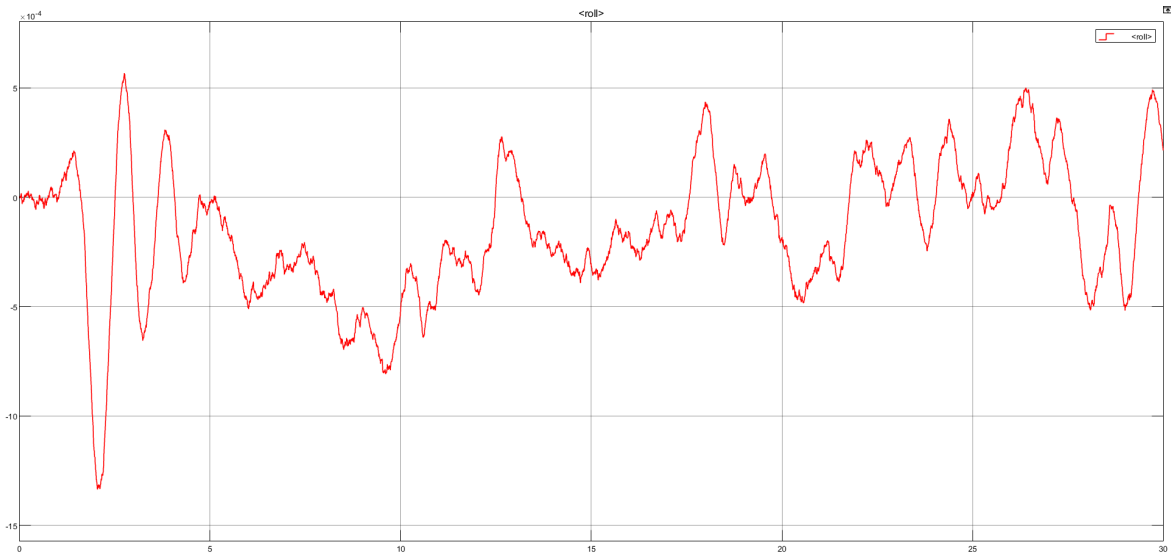


Figura 71: Tracciato *Roll* controllato da *G(s)*

Dato che il controllore è stato sintetizzato a partire dal processo linearizzato, è possibile notare dalle figure un notevole miglioramento nel tracciato controllato da *G(s)* (in *rosso*), dove le oscillazioni sono contenute nell'intorno di $10^{-}4$[m] attorno al valore di riferimento, a differenza di quelle ottenute con il controllo PID (in *blu*) comprese in $10^{-}3$[m].

8 Bibliografia

1. *"Techniques for Quadcopter Modelling and Design"*, Article May 2018.
2. Ing.Cesare Fantuzzi, *"Controllori standard PID"*, 1997
3. MATLAB, https://es.mathworks.com/help/aeroblks/quadcopter-project.html.
4. MATLAB, https://www.mathworks.com/videos/drone-simulation-and-control-part-1-setting-up-the-control-problem-1539323440930.html
5. Ing.Andrea Coglievina *"Controllo di un quadricottero per trasporto di carichi ignoti"*, 2014.
6. M. Letizia Corradini, Giuseppe Orlando *"Fondamenti di automatica"*, 2002.