



UNIVERSITA' POLITECNICA DELLE MARCHE
FACOLTA' DI INGEGNERIA

Corso di Laurea Triennale

In Ingegneria Informatica e dell'Automazione

**PROGETTO E SVILUPPO DI SISTEMI DI CONTROLLO
PER DRONI**

Project and development of drones control systems

Relatore:

Gianluca Ippoliti

Correlatore:

Giuseppe Orlando

Tesi di laurea di:

Lucia Paperi

A.A. 2018/2019

SOMMARIO

1	Introduzione	3
2	Caratteristiche fisiche del drone Mambo	5
3	Quadricottero	7
4	Coordinate del sistema di riferimento	11
5	Matrici di rotazione	13
6	Modello matematico	17
7	Forze e momenti	20
8	Modello linearizzato	24
9	Modello Simulink	26
9.1	Processo	27
9.2	Modello non lineare	28
9.3	Modello lineare	34
9.4	Controllo	35
10	Sistema di controllo per l'angolo di pitch	
10.1	Controllo	42
10.2	Sintesi in frequenza	42
10.3	Processo	44
10.4	Progettazione del controllore	51
11	PID	53
12	PID originale	55
13	Controllore in frequenza	57
14	Controllore PID originale nel caso lineare	59
15	Controllore PID originale nel caso non lineare	63
16	Controllore in frequenza nel caso lineare	67
17	Controllore in frequenza nel caso non lineare	71
18	Conclusioni	75

1 INTRODUZIONE

E' interessante sapere come sono nati e come si sono evoluti nel tempo gli APR (aeromobili a pilotaggio remoto), più comunemente conosciuti come droni.

L'impiego di un velivolo senza pilota, risale per la prima volta al 1849, utilizzato per scopi prettamente militari.

Grazie allo sviluppo tecnologico, nel periodo di tempo compreso tra le guerre mondiali, le aziende e le truppe militari pianificarono progetti che portarono alla conversione di alcuni modelli di aerei in APR (aeromobili a pilotaggio remoto) alla nascita dei primi sistemi senza pilota che potevano essere lanciati dalle navi da guerra e controllati mediante un autopilota.

La prima produzione in larga scala degli APR risale però al periodo della seconda guerra mondiale grazie a "Reginald Denny", che durante la prima guerra mondiale servì l'esercito britannico per poi trasferirsi negli Stati Uniti.

Negli anni '30, presentò all'Esercito americano uno dei primi esemplari di drone. Le autorità statunitensi inizialmente furono molto diffidenti, ma poi si sarebbero ricredute, spostandosi dall'impiego militare a quello civile.

L'origine della parola "drone" è ancora avvolta nel mistero. Attualmente vi sono numerose metafore che riconducono l'APR ad un insetto. In inglese moderno, infatti, il termine "drone" significa fuco, il maschio dell'ape. Si pensa che il nome derivi dal rumore simile al ronzio del fuco che i primi modelli producevano.

Se agli inizi la parola "drone" era stata assegnata una connotazione passiva, oggi possiamo dire che questi mezzi a pilotaggio remoto sono veri e propri protagonisti di una tecnologia molto sofisticata e in continua evoluzione.

A seguito del rapido progresso tecnologico registrato nel corso degli anni duemila, gli APR hanno cominciato ad essere utilizzati anche in ambito civile, dove sono impiegati in diversi settori, quali:

- settore cinematografico (riprese aeree, spot pubblicitari, etc);
- settore ambientale (monitoraggio ambientale, interventi tempestivi, analisi di rischio, etc);
- settore industriale (termorilevamenti, fotogrammetria, mapping, etc);
- settore agricolo (agricoltura di precisione, analisi stato del terreno, analisi delle colture, etc).

In questi anni lo sviluppo tecnologico ha permesso alle grandi case produttrici multinazionali (PARROT, DJI, WALKERA) di progettare droni con sistemi di sicurezza all'avanguardia che permettono di lavorare con estrema precisione.

Esistono varie tipologie dei droni, che possono essere ad elica come un elicottero, o ad ali come un aeroplano.

I droni ad elica si suddividono a loro volta in:

- monocottero o a singolo motore;
- tricotteri;
- quadricotteri;
- esacotteri;
- octacotteri.

Il mio progetto si è basato sullo sviluppo di un sistema di controllo per un quadricottero, più precisamente il mini-drone “Mambo” della casa Parrot.

2 CARATTERISTICHE FISICHE DEL DRONE MAMBO

In questo paragrafo farò una descrizione del mini-drone Mambo. Come detto prima, il mini-drone è un quadricottero, ovvero ha quattro eliche rotanti. Il suo peso è di soli 63gr.

In figura 1 è mostrata la parte inferiore del mini-drone. Si vedono due sensori: uno è un sensore ad ultrasuoni e l'altro è una fotocamera.



Figura 1 - Parte inferiore del mini-drone

Il sensore ad ultrasuoni serve per misurare la distanza verticale, ovvero la distanza dal suolo.

La fotocamera scatta foto a 60 fotogrammi al secondo e utilizza una tecnica di elaborazione dell'immagine chiamata "optical flow" che tiene traccia del movimento di oggetti sul terreno, grazie ad una videocamera, puntata verso il basso.

All'interno del mini-drone si ha un sensore di pressione che misura indirettamente l'altitudine.



Figura 2 – Parte superiore del mini-drone

Il terzo sensore è l'IMU è composto da un accelerometro e da un giroscopio a tre assi che misurano rispettivamente l'accelerazione lineare e la velocità angolare ed è un sensore di misura inerziale. Quindi vengono utilizzati il sensore ad ultrasuoni e quello di pressione per determinare l'altitudine, mentre l'IMU e la fotocamera per determinare il movimento rotazionale e traslazionale.

3 QUADRICOTTERO

Il quadricottero è un multirotores dotato di quattro motori, posti alle estremità delle braccia che vanno a formare una croce. I motori adiacenti ruotano nella direzione opposta e si tratta di un sistema robotico senza pilota a decollo e atterraggio verticale. Utilizzano un sistema di controllo che si basa sulla lettura dei dati provenienti dalla IMU (Inertial Measurement Unit) grazie al quale è possibile un volo stabilizzato.

L'unità IMU, ovvero "Inertial Measurement Unit", è costituita da un insieme di componenti elettroniche che sono fondamentali per il funzionamento del drone.

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.

Vediamo ora una breve descrizione delle componenti dell'IMU:

- *Giroscopio*: rivela la propria posizione (e quindi quella del drone), rispetto ad ognuno dei tre assi;
- *Accelerometro*: rileva l'accelerazione sui tre assi coordinati;
- *Barometro*: misura l'altitudine, rilevando le variazioni di pressione atmosferica e permette alla centralina di mantenere una quota costante;
- *Magnetometro*: è essenzialmente una bussola, che permette al multicottero di mantenere l'orientamento, grazie al magnetismo terrestre;
- *Sonar e Optical Flow*: il sonar è un dispositivo di localizzazione acustica, che può misurare la distanza del drone da terra ad un'altezza di pochi metri. L'Optical Flow, invece segue traccia del movimento di oggetti del terreno, grazie ad una videocamera, puntata verso il basso.

In figura 3 sono rappresentate le configurazioni possibili per un quadricottero, cioè a 'X' o a '+'.

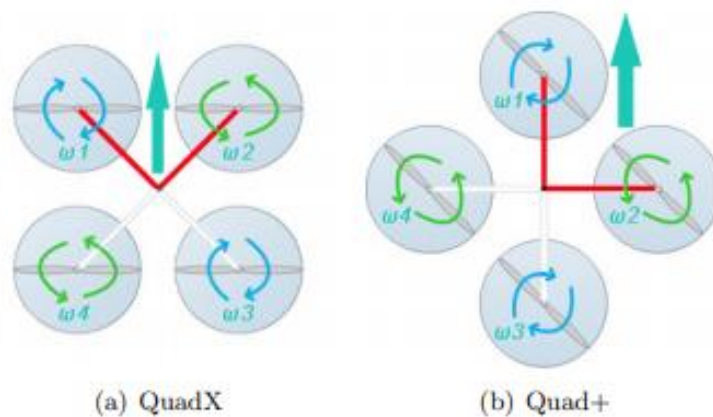


Figura 3 – Configurazione quadricottero

Il quadricottero è un sistema sottoattuato poiché ha sei gradi di libertà: beccheggio (pitch), imbardata (yaw), rollio (roll), x (movimento nella direzione frontale del velivolo), y (movimento verso il lato sinistro del velivolo) e z (altitudine), ma è controllato utilizzando solo quattro attuatori.

Il modello del quadricottero è schematizzabile secondo quanto mostrato in Figura 4: si osservi la terna trirettangola solidale a velivolo con asse X_B orientato 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.

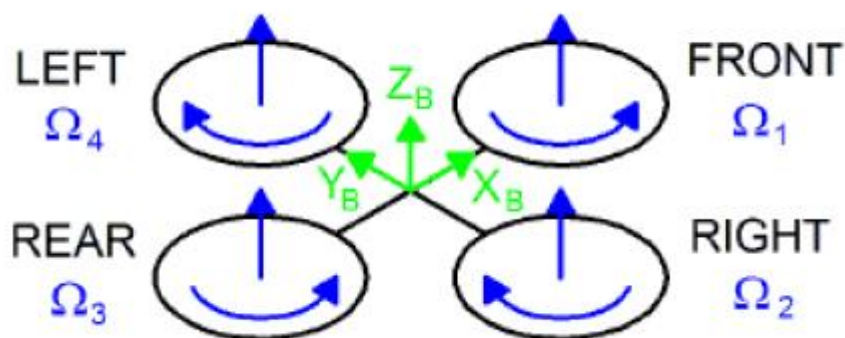


Figura 4 - Modello del quadricottero

I movimenti concessi al quadricottero sono 4, riassumibili secondo quanto riportato in Figura 5:

- a) spinta (thrust);
- b) rollio (roll);
- c) beccheggio (pitch);
- d) imbardata (yaw).

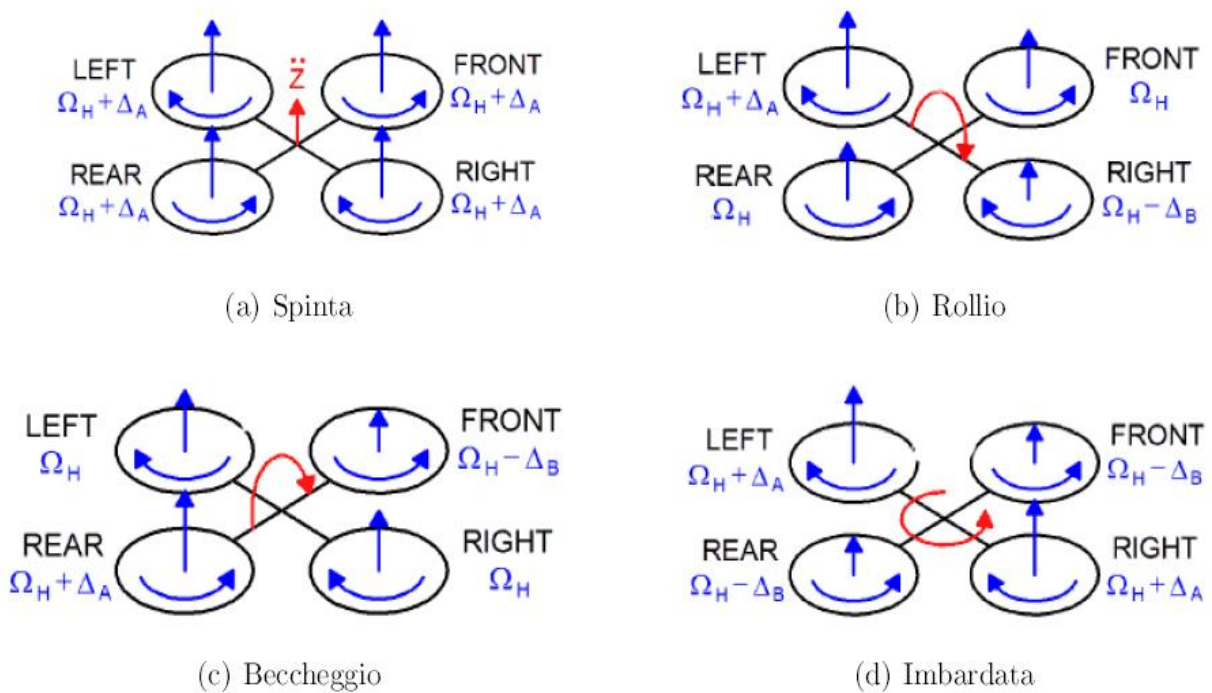


Figura 5 – Movimenti concessi al drone

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.

La condizione di hovering (volo a punto fisso) è una manovra che consiste nello stazionamento in volo a velocità nulla e quota costante. Questa condizione corrisponde matematicamente ad imporre i riferimenti di posizione nello spazio, e fissare l'angolo di imbardata, ovvero $(X_{ref}, Y_{ref}, Z_{ref}, \Psi_{ref})$. Solo in condizioni di perfetta simmetria della distribuzione della massa sul drone si può assumere che in condizione di hovering si abbia $\Omega_i = \Omega_H$.

Vediamo la spiegazione dei quattro riquadri in figura 5:

- In Figura 5(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;
- In Figura 5(b) è riportato il movimento di rollio (roll), ovvero una rotazione attorno all'asse x solidale al drone: viene incrementata la rotazione del rotore 4 di un fattore Δ_A e decrementata la rotazione del rotore 2 di un fattore Δ_B , mentre la rotazione associata ai rotori 1 e 3 rimane inalterata.
In tale modo la coppia indotta rispetto all'asse x non è più nulla, con conseguente accelerazione angolare attorno all'asse x.
- In Figura 5(c) è riportato il movimento di beccheggio (pitch), ovvero una rotazione attorno all'asse y solidale al drone: viene incrementata la rotazione del rotore 3 di un fattore Δ_A e decrementata la rotazione del rotore 1 di un fattore Δ_B , mentre la rotazione associata ai rotori 2 e 4 rimane inalterata.
In tale modo la coppia indotta rispetto all'asse y non è più nulla, con conseguente accelerazione angolare attorno all'asse y.
- In Figura 5(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.

4 COORDINATE DEL SISTEMA DI RIFERIMENTO

Per costruire il modello matematico del quadricottero, è importante definire le coordinate dei sistemi di riferimento che descrivono il moto di traslazione e di rotazione. Nel nostro modello matematico ne vedremo 3:

- **Inertial frame $\{i\}$:** è un sistema di riferimento fisso rappresentato dal vettore unitario $\{i\} = [x_i, y_i, z_i]^T$. L'asse x punta verso Nord, l'asse y verso est e l'asse z verso il centro della Terra;
- **Vehicle frame $\{v\}$:** ha la sua origine fissata nel centro di gravità del quadricottero ed è rappresentato dal vettore unitario $\{v\} = [x_v, y_v, z_v]^T$.
Gli assi sono allineati a quelli del sistema inerziale e non cambiano durante il movimento rotatorio del quadricottero. Le coordinate del “vehicle frame” descrivono il movimento traslazionale del quadricottero rispetto alle coordinate dell’ “inertial frame” sul piano x-y;
- **Body frame $\{b\}$:** in modo simile al sistema velivolo, ha la sua origine nel baricentro del quadricottero ed è rappresentato dal vettore unitario, $\{b\}=[x_b, y_b, z_b]^T$, che è rigidamente attaccato al corpo e quindi ruota con il corpo.
Il body frame descrive il movimento rotatorio del quadricottero rispetto a $\{v\}$. L'asse x punta sempre verso la parte anteriore del quadricottero, l'asse y a destra e l'asse z è rivolto verso il basso. $\{v\}$ può essere ruotato lungo l'asse z dall'angolo di imbardata (yaw, Ψ), lungo l'asse y dall'angolo di inclinazione (pitch, θ) e lungo l'asse x dall'angolo di rollio (roll, ϕ).

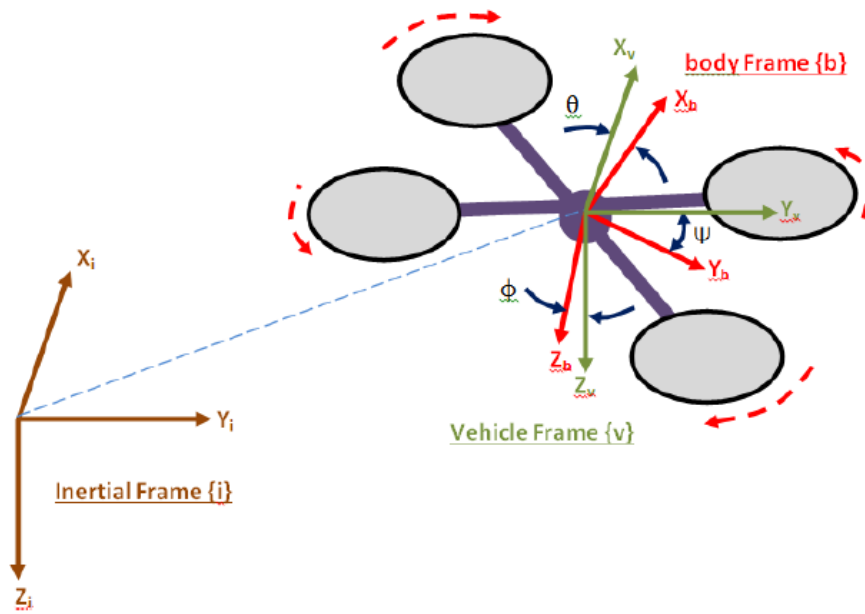


Fig. 6 – Coordinate del sistema di riferimento inerziale, veicolo e corpo

Una volta visti i tre sistemi di riferimento, ci spostiamo sullo studio delle matrici di rotazione.

5 MATRICI DI ROTAZIONE

La rotazione di un corpo rigido nello spazio può essere parametrizzata in più modi come ad esempio gli angoli di Eulero, l'utilizzo dei quaternioni o gli angoli di Tait-Bryan. Gli angoli di Eulero sono una rappresentazione matematica di tre rotazioni sugli assi di riferimento. Nel nostro caso gli assi sono orientati in modo che la direzione delle x positive siano il fronte del drone, le y positive la parte destra e l'asse delle z sia rivolto verso il basso. Questi tre angoli sono comunemente chiamati roll, pitch, yaw.

Se consideriamo un sistema orientato con la regola della mano destra, possiamo descrivere le tre rotazioni come:

- $R(x, \phi)$, rotazione attorno all'asse x,
- $R(y, \theta)$, rotazione attorno all'asse y,
- $R(z, \psi)$, rotazione attorno all'asse z.

In formule abbiamo:

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

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

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

La matrice di rotazione complessiva è data dall'unione delle tre singole rotazioni:

$$R(\phi, \theta, \psi) = R(x, \phi)R(y, \theta)R(z, \psi)$$

da cui ponendo, per semplicità, $\cos=c$ e $\sin=s$ si ottiene:

$$\mathbf{R}_b^i = \begin{bmatrix} c\psi c\theta & c\psi s\theta s\phi - s\psi c\phi & c\psi s\theta c\phi + s\psi s\phi \\ s\phi c\theta & s\psi s\theta s\phi + c\psi c\phi & s\psi s\theta c\phi - s\phi c\psi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix}$$

La matrice \mathbf{R}_b^i permette il passaggio da coordinate espresse nel body frame a coordinate nel sistema di riferimento inerziale. Essendo tale matrice ortogonale, le sue colonne e righe sono linearmente indipendenti.

Quindi il passaggio di coordinate inverse è espresso con la trasposta che coincide con l'inversa rappresentata nella matrice \mathbf{R}_i^b :

$$\mathbf{R}_i^b = \begin{bmatrix} c\psi c\theta & s\psi c\theta & -s\theta \\ c\psi s\theta s\phi - s\psi c\phi & s\psi s\theta s\phi + c\psi c\phi & c\theta s\phi \\ c\psi s\theta c\phi + s\psi s\phi & s\psi s\theta c\phi - s\phi c\psi & c\theta c\phi \end{bmatrix}$$

Le equazioni delle posizioni lineari e angolari del quadricottero nell' "inertial frame" e nel "body frame" possono essere espresse in forma vettoriale come:

$$P^b = \begin{bmatrix} x^b & y^b & z^b \end{bmatrix}^T$$

$$P^i = \begin{bmatrix} x^i & y^i & z^i \end{bmatrix}^T$$

$$\Lambda^b = \begin{bmatrix} \phi & \theta & \psi \end{bmatrix}^T$$

Fig. 7 – Posizioni lineari e angolari

In modo simile, le velocità lineari e angoli nel "body frame" possono essere scritte in forma vettoriale come

$$V^b = \begin{bmatrix} u & v & w \end{bmatrix}^T$$

$$\dot{\Lambda} = \begin{bmatrix} p & q & r \end{bmatrix}^T$$

Fig. 8 – Velocità lineari e angolari

La trasformazione delle velocità angolari del quadricottero dall' "inertial frame" al "body frame" può essere descritto dalla seguente equazione:

$$\omega^b = R(\phi) \cdot R(\theta) \cdot R(\psi) \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} + R(\phi) \cdot R(\theta) \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + R(\phi) \begin{bmatrix} 0 \\ 0 \\ \dot{\phi} \end{bmatrix}$$

dove:

ω^b = velocità angolare in "body frame";

$R(\phi)$ = matrice di rotazione lungo asse x (roll);

$R(\theta)$ = matrice di rotazione lungo asse y (pitch);

$R(\psi)$ = matrice di rotazione lungo asse z (yaw).

da cui:

$$\omega^b = \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & -s\theta \\ 0 & c\phi & c\theta s\phi \\ 0 & -s\phi & c\theta c\phi \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}$$

L'operazione inversa, cioè il passaggio delle velocità angolari verso il sistema inerziale, partendo da quello solidale con il drone, avviene determinando l'inversa della matrice ω^b , cioè:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & s\phi \frac{s\theta}{c\theta} & c\phi \frac{s\theta}{c\theta} \\ 0 & c\phi & -s\phi \\ 0 & s\phi \frac{1}{c\theta} & c\theta \frac{1}{c\theta} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

6 MODELLO MATEMATICO

Vediamo ora il modello matematico del quadricottero, sfruttando Newton e le equazioni di Eulero per il movimento di un corpo rigido.

Chiamiamo $[x \ y \ z \ \phi \ \theta \ \psi]^T$ il vettore contenente la posizione lineare e angolare del quadricottero nel sistema inerziale e con $[u \ v \ w \ p \ q \ r]^T$ il vettore contenente le velocità lineari e angolari nel "body frame". Dalla dinamica del corpo, ne consegue che i due sistemi di riferimento sono collegati dalle seguenti relazioni:

$$\mathbf{v} = \mathbf{R} \cdot \mathbf{v}_B,$$

$$\boldsymbol{\omega} = \mathbf{T} \cdot \boldsymbol{\omega}_B,$$

dove:

$$\mathbf{v} = [\dot{x} \ \dot{y} \ \dot{z}]^T \in \mathbb{R}^3, \boldsymbol{\omega} = [\dot{\phi} \ \dot{\theta} \ \dot{\psi}]^T \in \mathbb{R}^3, \mathbf{v}_B = [u \ v \ w]^T \in \mathbb{R}^3,$$

$$\boldsymbol{\omega}_B = \begin{bmatrix} 1 & c(\phi)t(\theta) & c(\phi)t(\theta) \\ p & q & r \\ 0 & \frac{s(\phi)}{c(\theta)} & \frac{c(\phi)}{c(\theta)} \end{bmatrix}^T \in \mathbb{R}^3(\phi)$$

con $t(\theta) = \tan(\theta)$.

Quindi il modello del quadricottero è:

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

La legge di Newton stabilisce la seguente relazione matriciale per la forza totale che agisce sul quadricottero:

$$m(\boldsymbol{\omega}_B \wedge \mathbf{v}_B + \dot{\mathbf{v}}_B) = \mathbf{f}_B$$

dove m è la massa del quadricottero, $\mathbf{f}_B = [f_x \ f_y \ f_z]$ è la forza totale.

L'equazione di Eulero ci fornisce la coppia totale applicata al quadricottero:

$$\mathbf{I} \cdot \dot{\boldsymbol{\omega}}_B + \boldsymbol{\omega}_B \wedge (\mathbf{I} \cdot \boldsymbol{\omega}_B) = \mathbf{m}_B,$$

dove $\mathbf{m}_B = [m_x \ m_y \ m_z]$ è la coppia totale, \mathbf{I} è la matrice inerziale diagonale:

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

Quindi il modello dinamico del quadricottero nel “body frame” è:

$$\left\{ \begin{array}{l} f_x = m(\dot{u} + qw - rv) \\ f_y = m(\dot{v} - pw + ru) \\ f_z = m(\dot{w} + pv - qu) \\ m_x = \dot{p}I_x - qrI_y + prI_z \\ m_y = \dot{q}I_y + prI_x - pqI_z \\ m_z = \dot{r}I_z - pqI_x + prI_y \end{array} \right.$$

7 FORZE E MOMENTI

Per completare lo studio, bisogna vedere quali sono le forze e i momenti esterni agenti sul velivolo.

Le forze esterne nel “body frame”, \mathbf{f}_B sono date da:

$$\mathbf{f}_B = mg\mathbf{R}^T \cdot \hat{\mathbf{e}}_z - f_t \hat{\mathbf{e}}_3$$

dove $\hat{\mathbf{e}}_z$ è il vettore unitario dell’asse z inerziale, $\hat{\mathbf{e}}_3$ è il vettore unitario dell’asse zeta nel “body frame”, g è l’accelerazione gravitazionale.

I momenti esterni sono dati dalla formula:

$$\mathbf{m}_B = \boldsymbol{\tau}_B - \mathbf{g}_a + \boldsymbol{\tau}_w$$

dove:

\mathbf{g}_a rappresenta i momenti giroscopici causati dalla rotazione combinata dei quattro rotori e del corpo del velivolo,

$\boldsymbol{\tau}_B = [\tau_x \quad \tau_y \quad \tau_z]^T \in \mathbb{R}^3$ sono le coppie di controllo generate dalla differenza di velocità nei rotori,

$\boldsymbol{\tau}_w = [\tau_{wx} \quad \tau_{wy} \quad \tau_{wz}]^T$ sono le coppie prodotte dal vento nei quadricotteri.

$$\mathbf{g}_a \text{ è data da } \quad \mathbf{g}_a = \sum_{i=1}^4 J_p (\boldsymbol{\omega}_B \wedge \hat{\mathbf{e}}_3) (-1)^{i+1} \Omega_i$$

dove J_p è l'inerzia di ogni rotore, Ω_i è la velocità angolare dell' i -esimo rotore.

Il termine J_p risulta essere piccolo e, per questa ragione, i momenti giroscopici vengono rimossi nella formulazione del controller. Inoltre, ci sono numerosi fenomeni aerodinamici e aeroelastici che influenzano il volo del quadricottero, come gli effetti al suolo: quando si vola vicino al suolo (o durante la fase di atterraggio), il flusso d'aria generato dalle eliche disturba la dinamica dei quadrotori. Quindi, il modello dinamico completo del quadrotore nel "body frame" viene sostituito sostituendo l'espressione della forza in:

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

Ora consideriamo gli input che possono essere applicati al sistema per controllare il comportamento del quadricottero. I rotori sono quattro mentre i gradi di libertà che controlliamo sono di più: comunemente, gli ingressi di controllo considerati sono uno per la spinta verticale e uno per ciascuno dei movimenti angolari. Nelle equazioni sotto consideriamo i valori delle forze di ingresso e delle coppie proporzionali alle velocità al quadrato dei rotori:

$$\begin{cases} f_t = b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \\ \tau_x = bl(\Omega_3^2 - \Omega_1^2) \\ \tau_y = bl(\Omega_4^2 - \Omega_2^2) \\ \tau_z = d(\Omega_2^2 + \Omega_4^2 - \Omega_1^2 - \Omega_3^2) \end{cases}$$

dove l è la distanza tra ogni rotore e il centro del drone, b è il fattore di spinta e d è il fattore di resistenza.

Sostituendo queste equazioni in quelle precedentemente trovate, otteniamo:

$$\begin{cases} -mg[s(\theta)] + f_{wx} = m(\dot{u} + qw - rv) \\ mg[c(\theta)s(\phi)] + f_{wy} = m(\dot{v} - pw + ru) \\ mg[c(\theta)c(\phi)] + f_{wz} - b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) = m(\dot{w} + pv - qu) \\ bl(\Omega_3^2 - \Omega_1^2) + \tau_{wx} = \dot{p}I_x - qrI_y + qrI_z \\ bl(\Omega_4^2 - \Omega_2^2) + \tau_{wy} = \dot{q}I_y + prI_x - prI_z \\ d(\Omega_2^2 + \Omega_4^2 - \Omega_1^2 + \Omega_3^2) + \tau_{wz} = \dot{r}I_z - pqI_x + pqI_y \end{cases}$$

Avendo il vettore degli stati così composto:

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

possiamo riscrivere le equazioni della dinamica del quadricottero in una forma spazio di stato:

$$\left\{ \begin{array}{l} \dot{\phi} = p + r[c(\phi)t(\theta)] + q[s(\phi)t(\theta)] \\ \dot{\theta} = q[c(\phi)] - r[s(\phi)] \\ \dot{\psi} = r\frac{c(\phi)}{c(\theta)} + q\frac{s(\phi)}{c(\theta)} \\ \dot{p} = \frac{I_y - I_z}{I_x}rq + \frac{\tau_x + \tau_{wx}}{I_x} \\ \dot{q} = \frac{I_z - I_x}{I_y}pr + \frac{\tau_y + \tau_{wy}}{I_y} \\ \dot{r} = \frac{I_x - I_y}{I_z}pq + \frac{\tau_z + \tau_{wz}}{I_z} \\ \dot{u} = rv - qw - g[s(\theta)] + \frac{f_{wx}}{m} \\ \dot{v} = pw - ru + g[s(\phi)c(\theta)] + \frac{f_{wy}}{m} \\ \dot{w} = qu - pv + g[c(\theta)c(\phi)] + \frac{f_{wz} - ft}{m} \\ \dot{x} = w[s(\phi)s(\psi) + c(\phi)c(\psi)s(\theta)] - v[c(\phi)s(\psi) - c(\psi)s(\phi)s(\theta)] + u[c(\psi)c(\theta)] \\ \dot{y} = v[c(\phi)c(\psi) + s(\phi)s(\psi)s(\theta)] - w[c(\psi)s(\phi) - c(\phi)s(\psi)s(\theta)] + u[c(\theta)s(\psi)] \\ \dot{z} = w[c(\phi)c(\theta)] - u[s(\theta)] + v[c(\theta)s(\phi)] \end{array} \right.$$

8 MODELLO LINEARIZZATO

Un sistema non lineare generico può essere scritto nella seguente forma:

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

in cui x è il vettore degli stati, u degli ingressi e t è il tempo.

Linearizziamo il sistema intorno ad un punto di lavoro $(x_0(t), u_0(t))$, che soddisfi:

$$\dot{x}_0 = f[x_0(t), u_0(t), t]$$

Definiamo ora tre variabili Δx , Δy , Δz che rappresentano rispettivamente lo scostamento dello stato, dell'ingresso e dell'uscita attorno ai valori in cui si è deciso di linearizzare. Abbiamo quindi:

$$\begin{aligned} x(t) &= x_0(t) + \Delta x(t) \\ u(t) &= u_0(t) + \Delta u(t) \\ y(t) &= h[x_0(t), u_0(t), t] + \Delta y(t) \end{aligned}$$

Utilizziamo ora gli sviluppi Taylor fino al primo ordine, ottenendo:

$$\begin{aligned} \dot{x} &= f[x_0(t), u_0(t), t] + \left. \frac{\partial f}{\partial x} \right|_{x_0(t), u_0(t)} \Delta x + \left. \frac{\partial f}{\partial u} \right|_{x_0(t), u_0(t)} \Delta u \\ y &= h[x_0(t), u_0(t), t] + \left. \frac{\partial h}{\partial x} \right|_{x_0(t), u_0(t)} \Delta x + \left. \frac{\partial h}{\partial u} \right|_{x_0(t), u_0(t)} \Delta u \end{aligned}$$

Sostituiamo

$$\dot{x}_0 = f[x_0(t), u_0(t), t]$$

nella prima equazione, e sapendo che:

$$y(t) = h[x_0(t), u_0(t), t] + \Delta y(t)$$

possiamo scrivere:

$$\begin{aligned}\Delta \dot{x} &= \left. \frac{\partial f}{\partial x} \right|_{x_0(t), u_0(t)} \Delta x + \left. \frac{\partial f}{\partial u} \right|_{x_0(t), u_0(t)} \Delta u \\ \Delta y &= \left. \frac{\partial h}{\partial x} \right|_{x_0(t), u_0(t)} \Delta x + \left. \frac{\partial h}{\partial u} \right|_{x_0(t), u_0(t)} \Delta u\end{aligned}$$

Le matrici A, B, C, D sono così ottenute:

$$\begin{aligned}A(t) &= \left. \frac{\partial f}{\partial x} \right|_{x_0(t), u_0(t)}, & B(t) &= \left. \frac{\partial f}{\partial u} \right|_{x_0(t), u_0(t)} \\ C(t) &= \left. \frac{\partial h}{\partial x} \right|_{x_0(t), u_0(t)}, & D(t) &= \left. \frac{\partial h}{\partial u} \right|_{x_0(t), u_0(t)}\end{aligned}$$

Possiamo quindi scrivere il sistema linearizzato nel seguente modo:

$$\begin{aligned}\dot{x} &= Ax + Bu \\ y &= Cx + Du\end{aligned}$$

9 MODELLO SIMULINK

Aprendo un nuovo progetto, negli esempi di AerospaceBlockset, con il comando `asbQuadcopterStart`, digitato sulla command window di Matlab, si apre il modello complessivo per simulare la dinamica del Mambo, come possiamo vedere in figura 9. Notiamo subito lo schema di controllo in controreazione dove il processo è dato dal blocco Airframe e la parte di controllo dal blocco FCS.

Le altre componenti servono per la generazione dei riferimenti, lettura dei valori dei sensori, simulazione dell'ambiente.

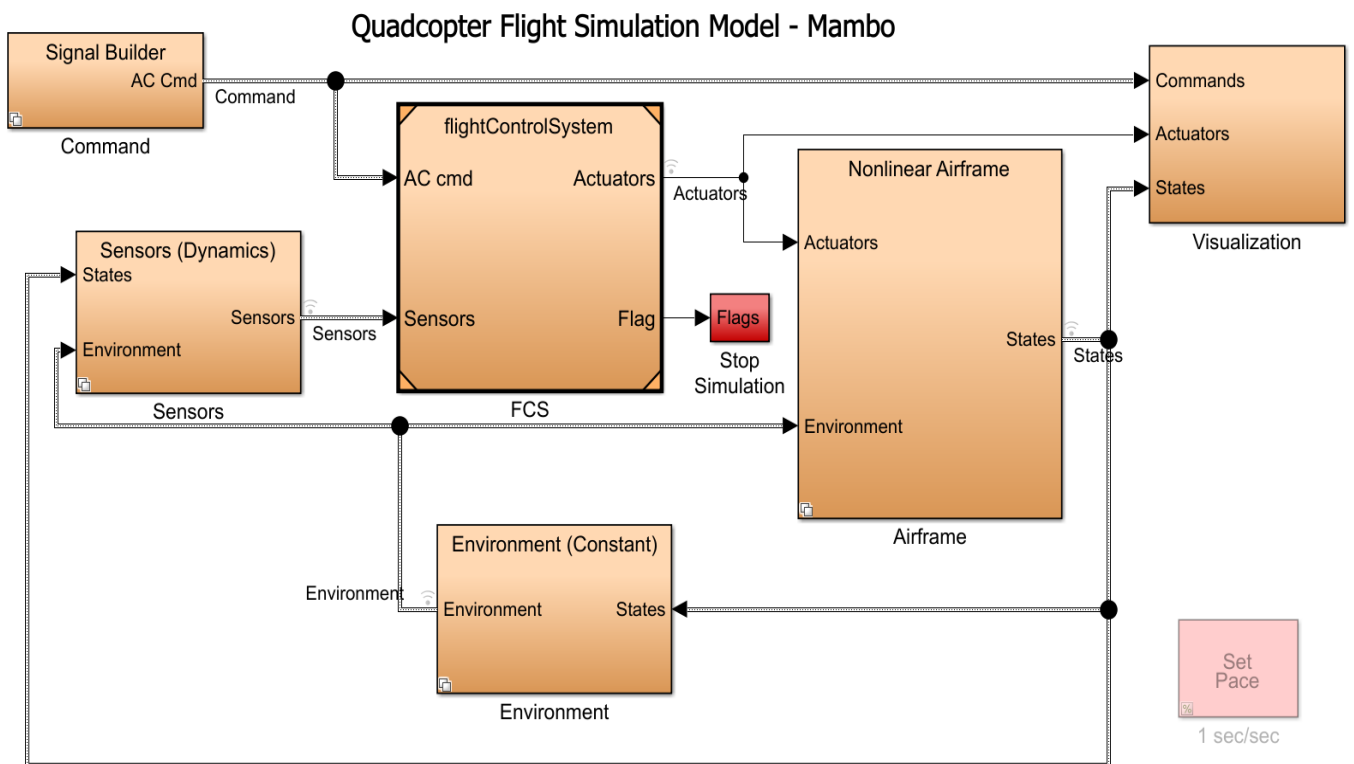


Fig. 9 – Modello a blocchi iniziale

9.1 PROCESSO

Nel blocco Airframe si trova tutta la dinamica lineare e non lineare. Tramite la variabile VSS_VEHICLE è possibile selezionare quale tra le due modalità andrà utilizzata; per scegliere la modalità lineare bisognerà settare la variabile al valore 0, per quella non lineare al valore 1, come possiamo vedere dalla Figura 10.

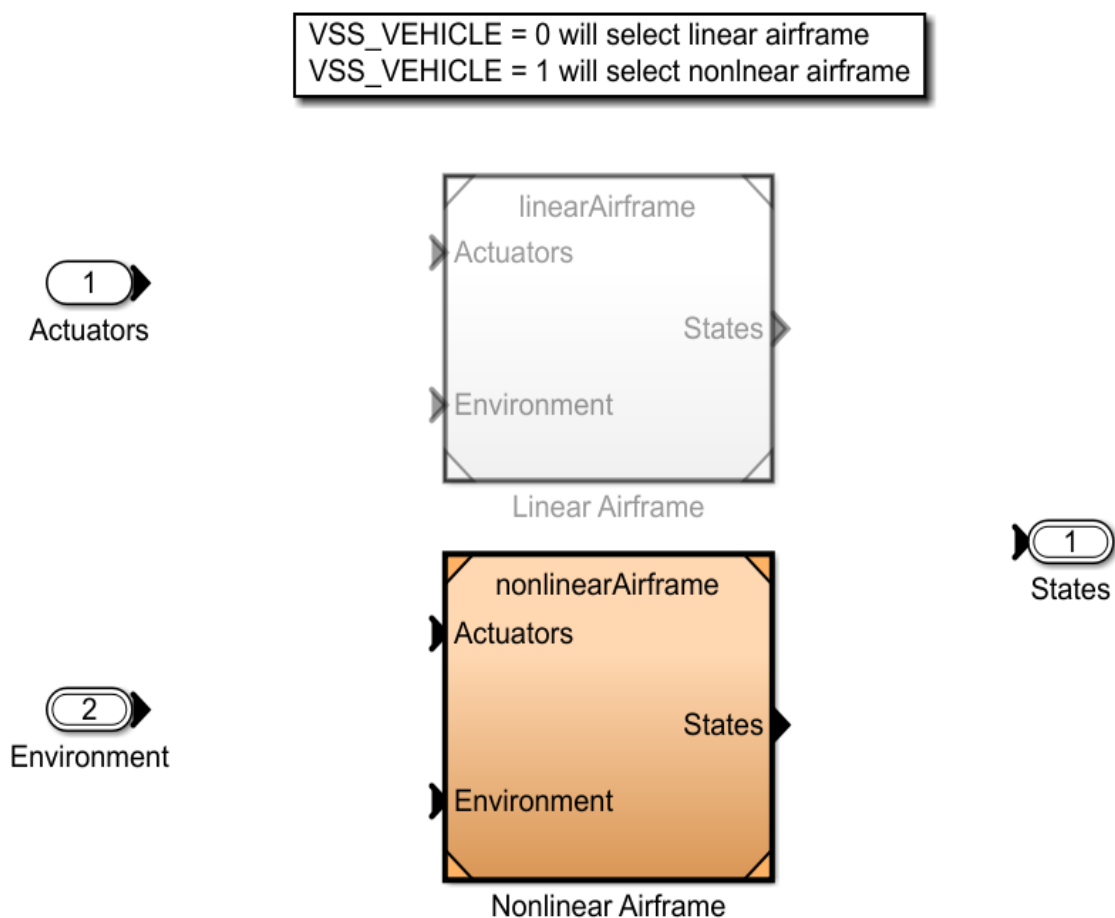


Fig. 10 – Modello lineare / non lineare

9.2 MODELLO NON LINEARE

Vediamo ora il modello Simulink non lineare:

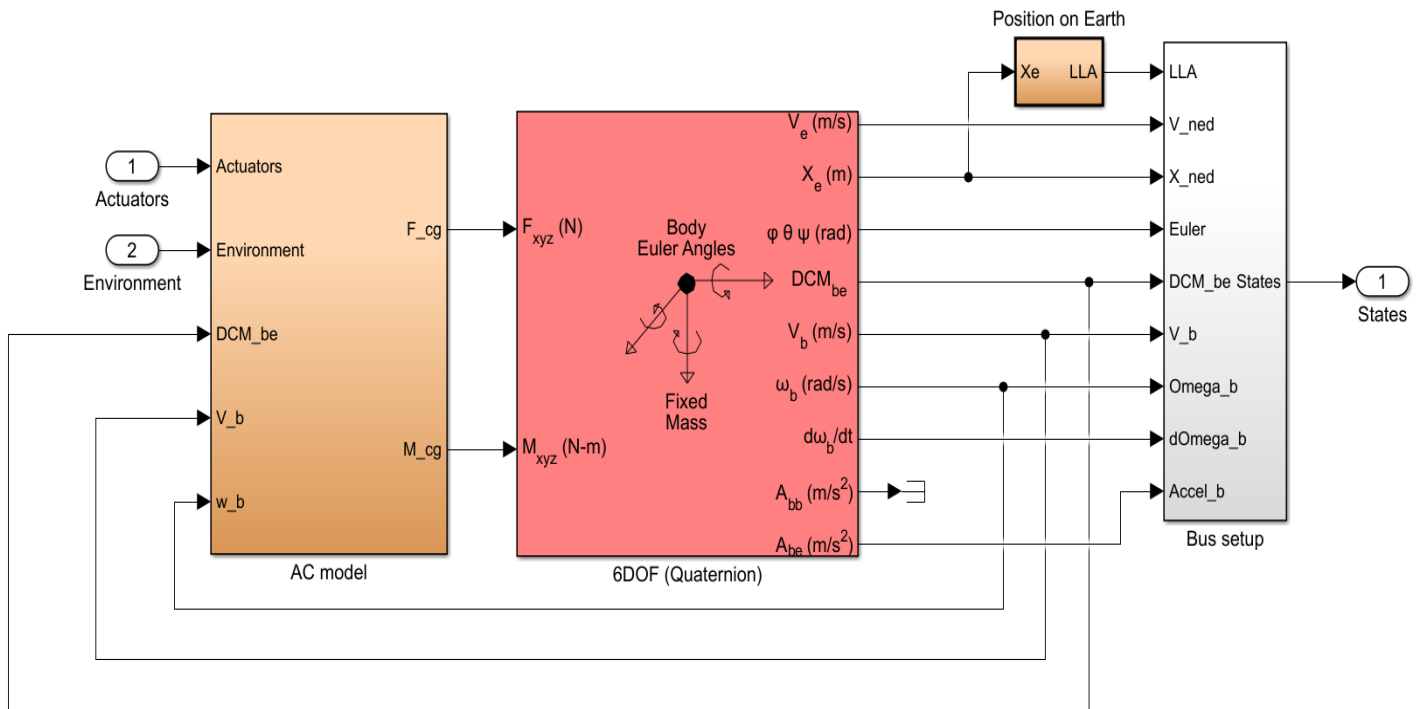


Fig. 11 – Modello non lineare

Come possiamo vedere dalla figura 11 abbiamo 3 blocchi:

- **AC model:** è il blocco che contiene la parte della dinamica. In ingresso troviamo le velocità angolari ω_b e le velocità lineari v_b espresse rispetto al “body frame” ed uscenti dal blocco 6DOF. In uscita si hanno le forze e i momenti su tutti e tre gli assi;
- **6DOF (Quaternion):** in ingresso prende le forze e i momenti, in uscita le grandezze di velocità, accelerazioni e angoli;
- **Bus setup:** è un vettore di bus che contiene tutte le velocità sia dell’ “inertial frame” che del “body frame”, la posizione, le accelerazioni e gli angoli di rotazione.

Aprendo il blocco AC model, abbiamo:

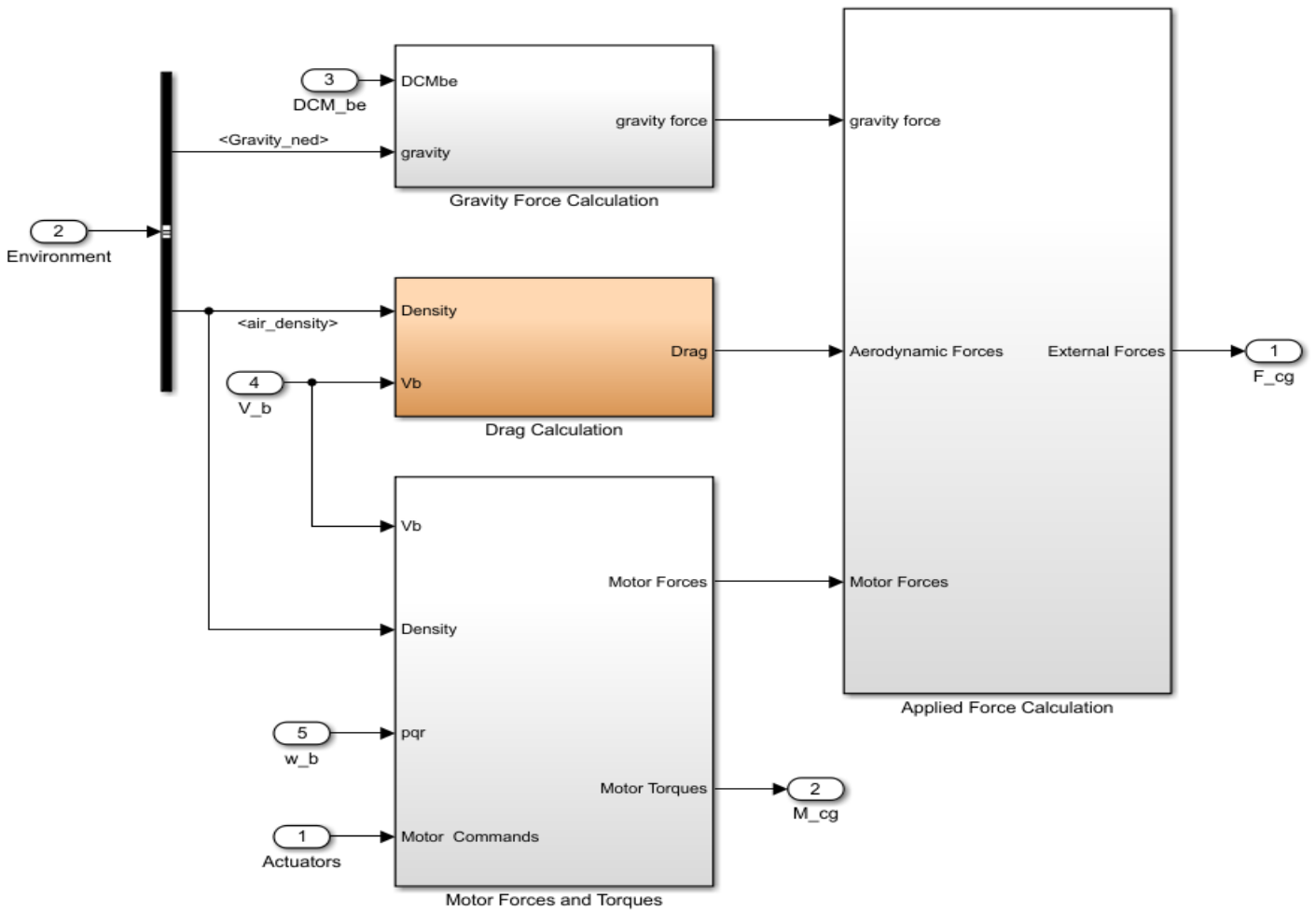


Fig. 12 – AC Model

Vediamo nel dettaglio i quattro sottoblocchi:

1. **Gravity Force Calculation:** calcola la forza di gravità rispetto al “body frame”. Come ingresso ha la matrice DCM_be e la costante di gravità.

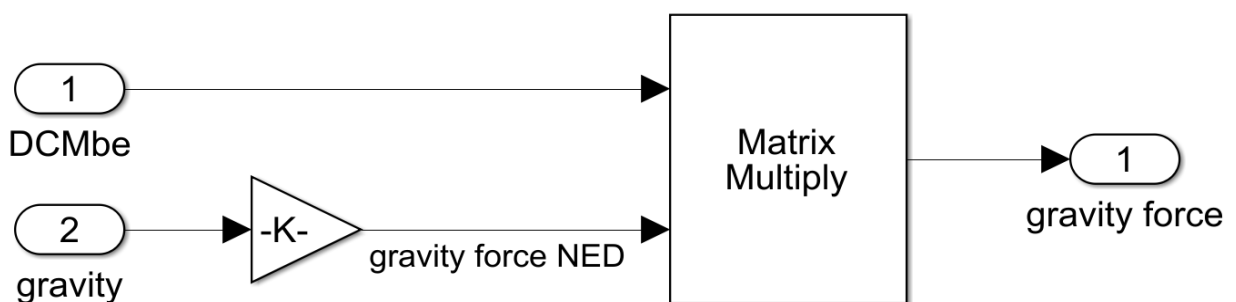


Fig. 13 - Gravity Force Calculation

2. **Drag Calculation:** calcola la forza di resistenza che l'aria imprime al drone in movimento. Tale forza è dovuta allo spostamento del drone ed è esprimibile come:

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

dove:

ρ è la densità dell'aria;

V è la velocità di traslazione del drone

S è la superficie che impatta l'aria durante la traslazione;

C_D è il coefficiente di drag.

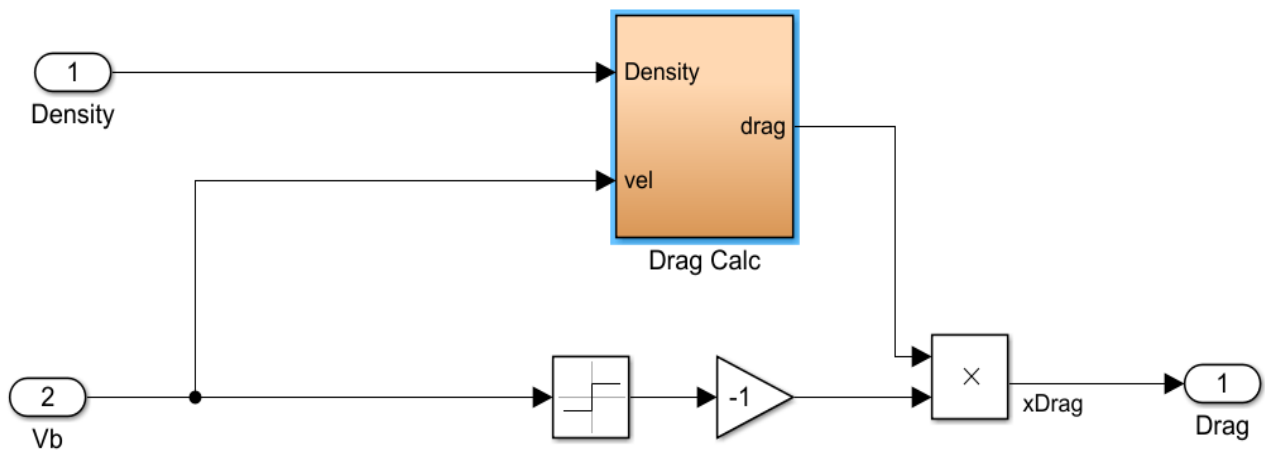


Fig. 14 - Drag Calculation

3. **Applied Force Calculation:** si ha un blocco contenente la forza di gravità, la forza di drag e le forze generate dai motori.

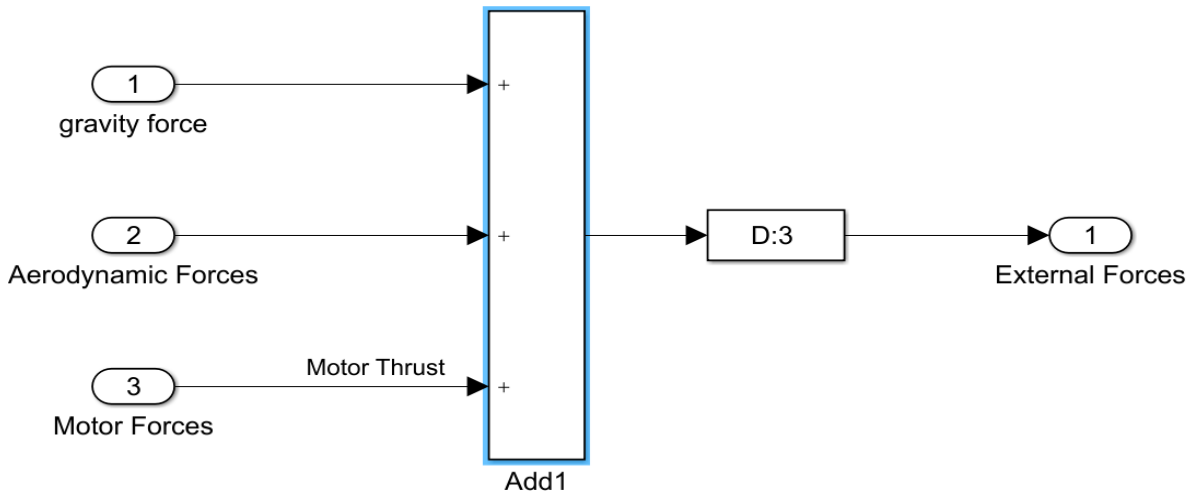


Fig. 15 - Applied Force Calculation

4. **Motor Forces and Torques**

Contiene il modello descritto e ricavato dalle equazioni viste in precedenza. In figura 16 notiamo un blocco centrale che ha come ingressi le velocità lineari e angolari rispetto al “body frame”, il vettore Motor Commands che 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. Il blocco MotorsToW trasforma lo sforzo di controllo in un segnale interpretabile dai motori.

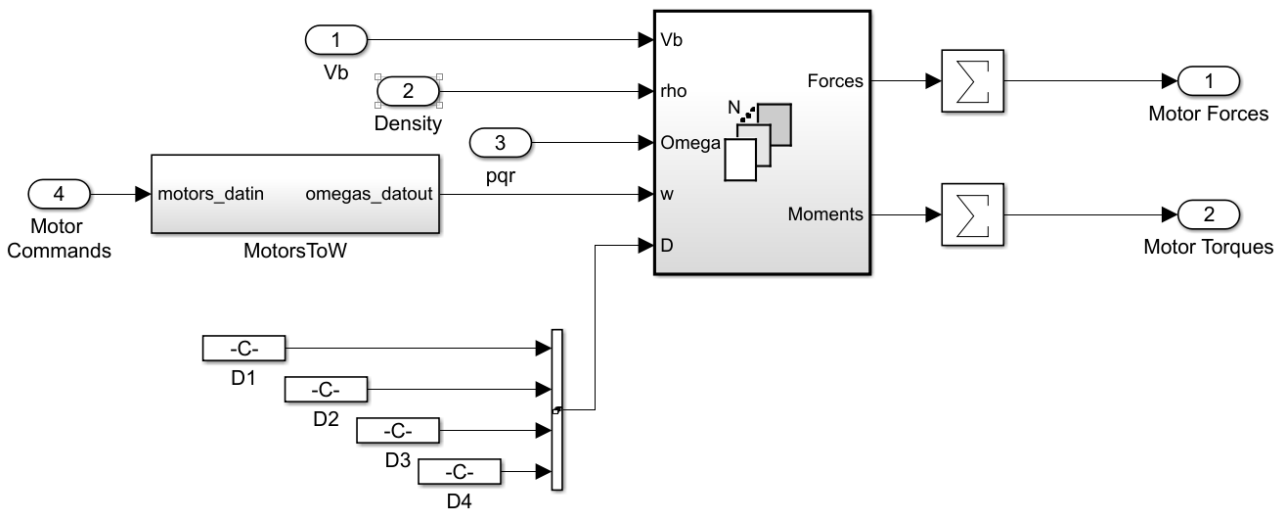


Fig. 16 - Motor Forces and Torques

In figura 17 vediamo il sottoblocco centrale:

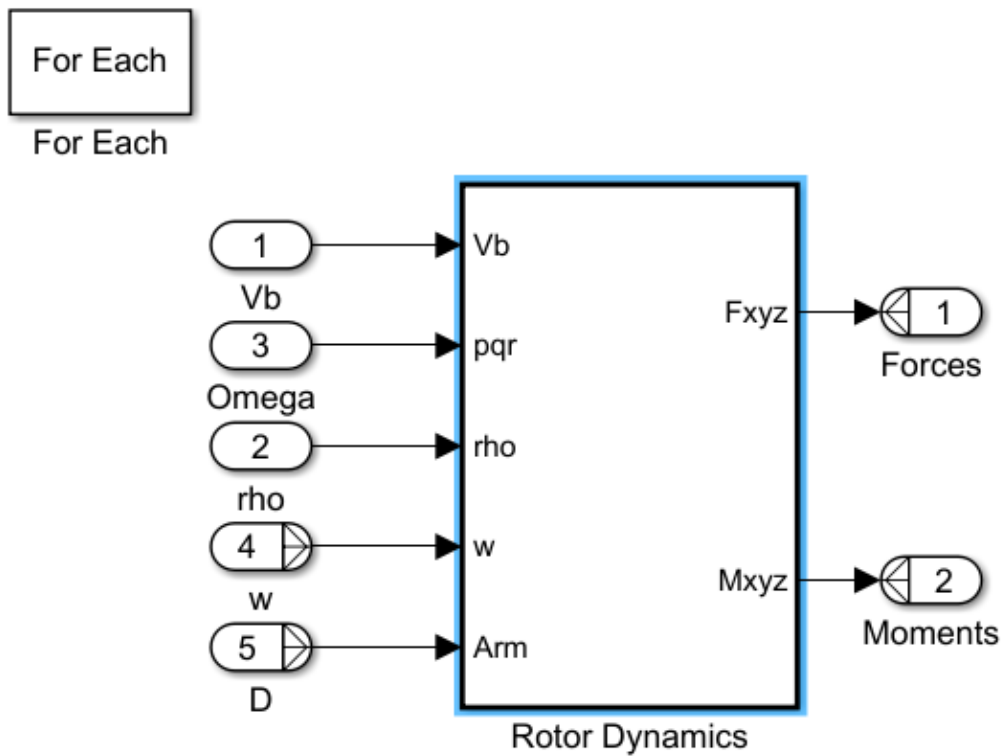


Fig. 17 - Dinamica per ogni motore

Tramite il blocco For EAcH viene stabilito che la dinamica sia uguale per tutti e quattro gli attuatori.

Entrato nel blocco "Rotor Dynamics", nelle figure 18a e 18b (in cui 18a è la parte sinistra mentre 18b è quella destra) troviamo la rappresentazione del modello matematico descritto in precedenza:

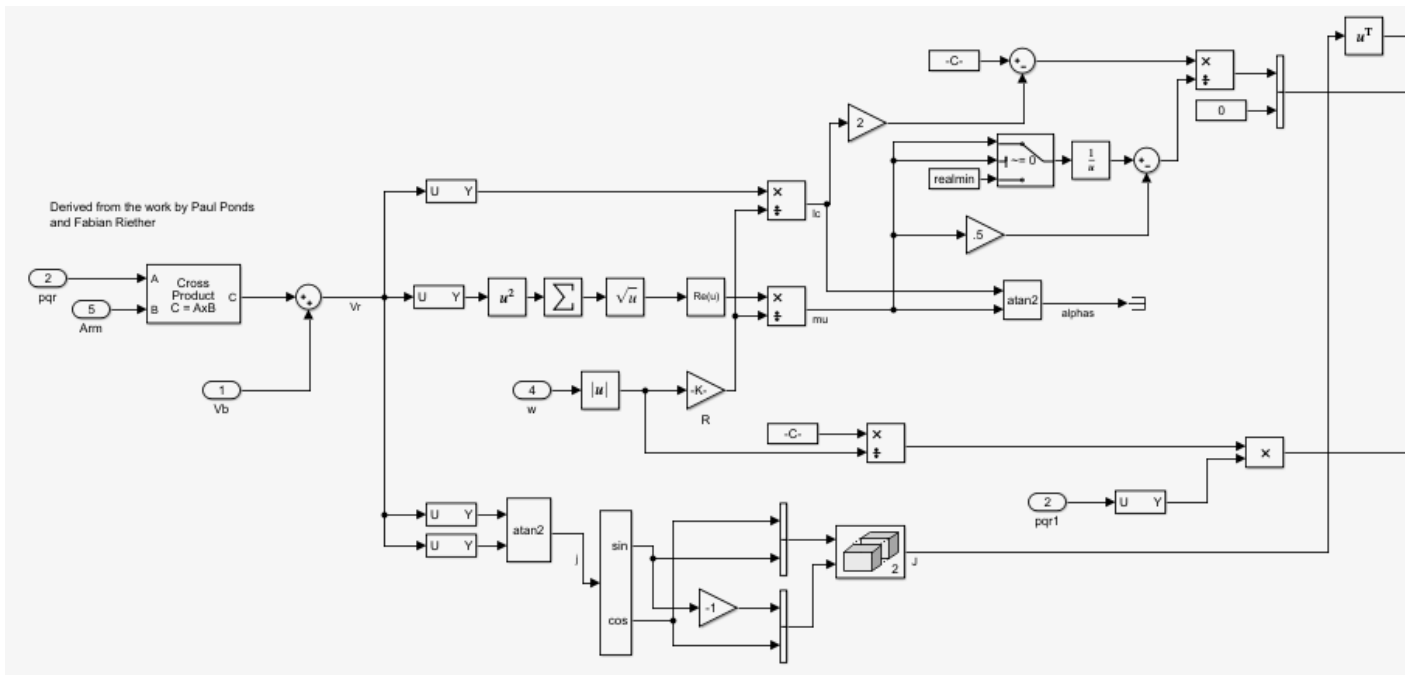


Fig. 18a - Rotor Dynamics s_x

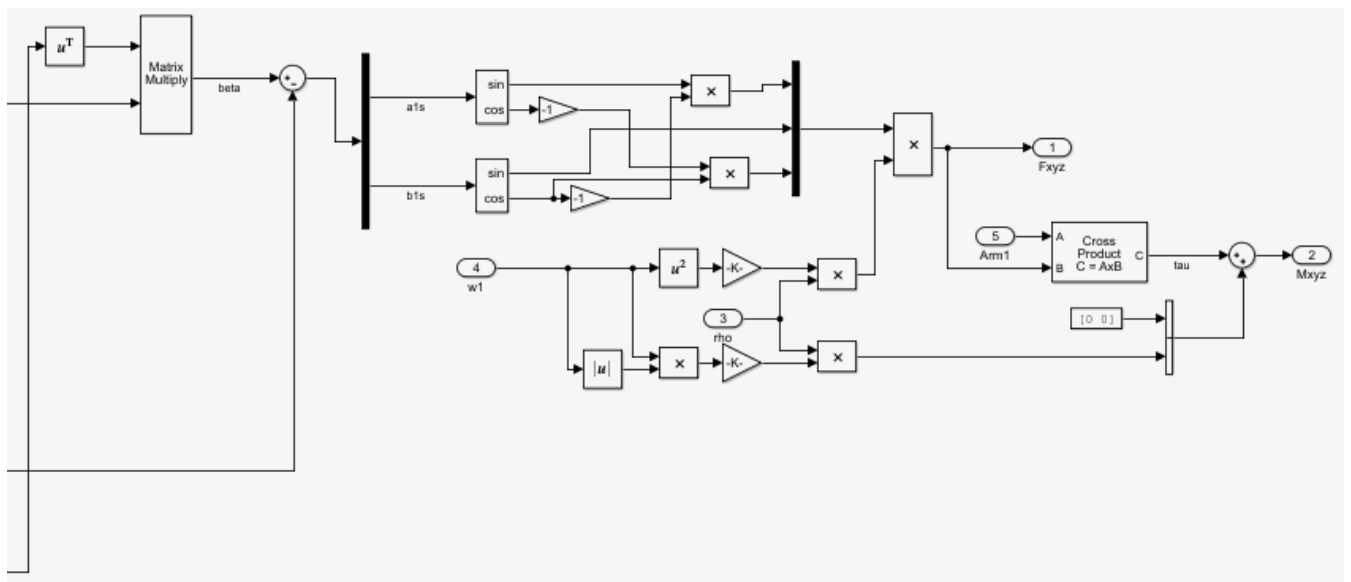


Fig. 18b - Rotor Dynamics dx

9.3 MODELLO LINEARE

All'interno del blocco Airframe, in LinearAirframe troviamo la linearizzazione e le matrici A, B, C, D:

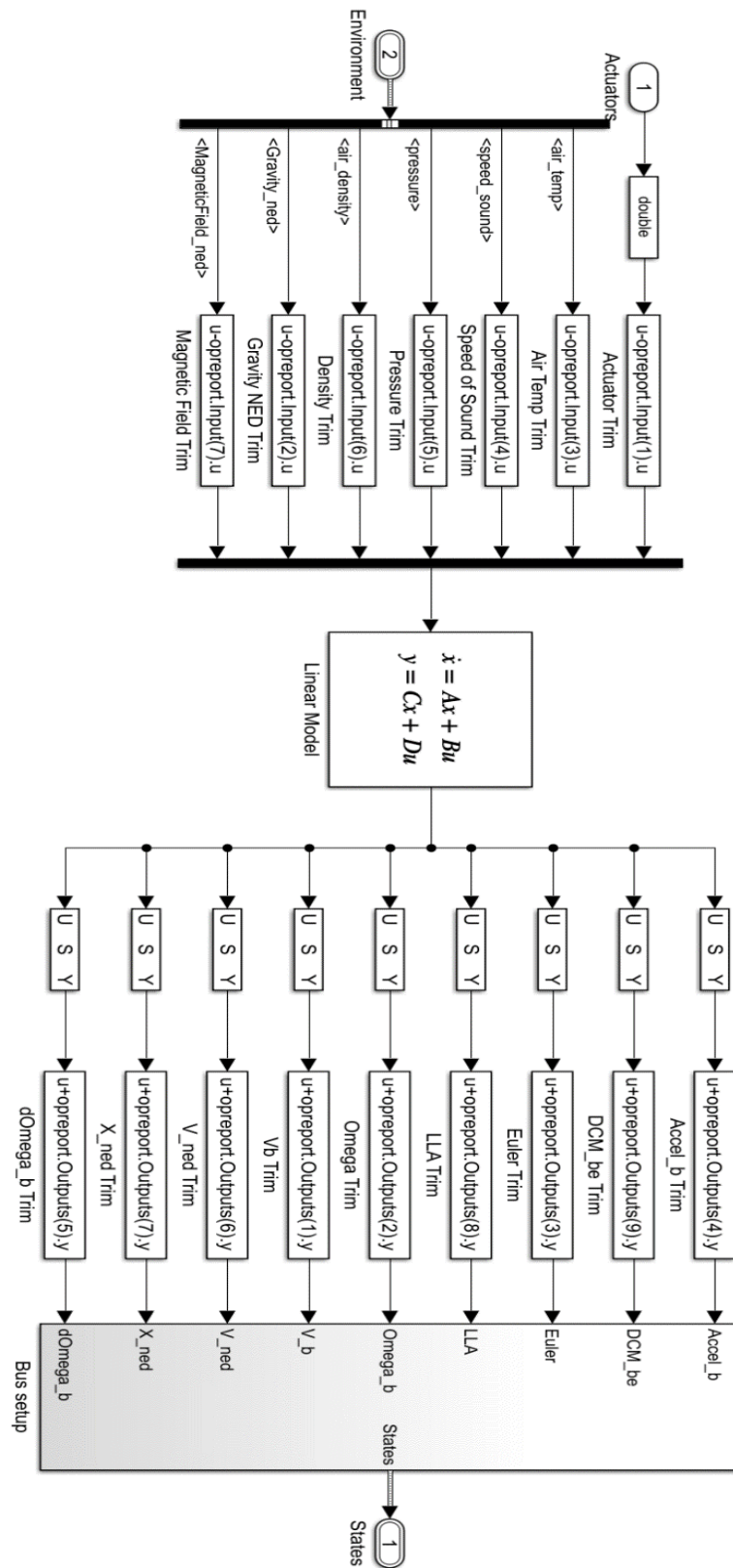


Fig. 19 - Modello lineare

9.4 CONTROLLO

Tutta la parte relativa al controllo la troviamo nel blocco FCS, presente nel modello iniziale in figura 9.

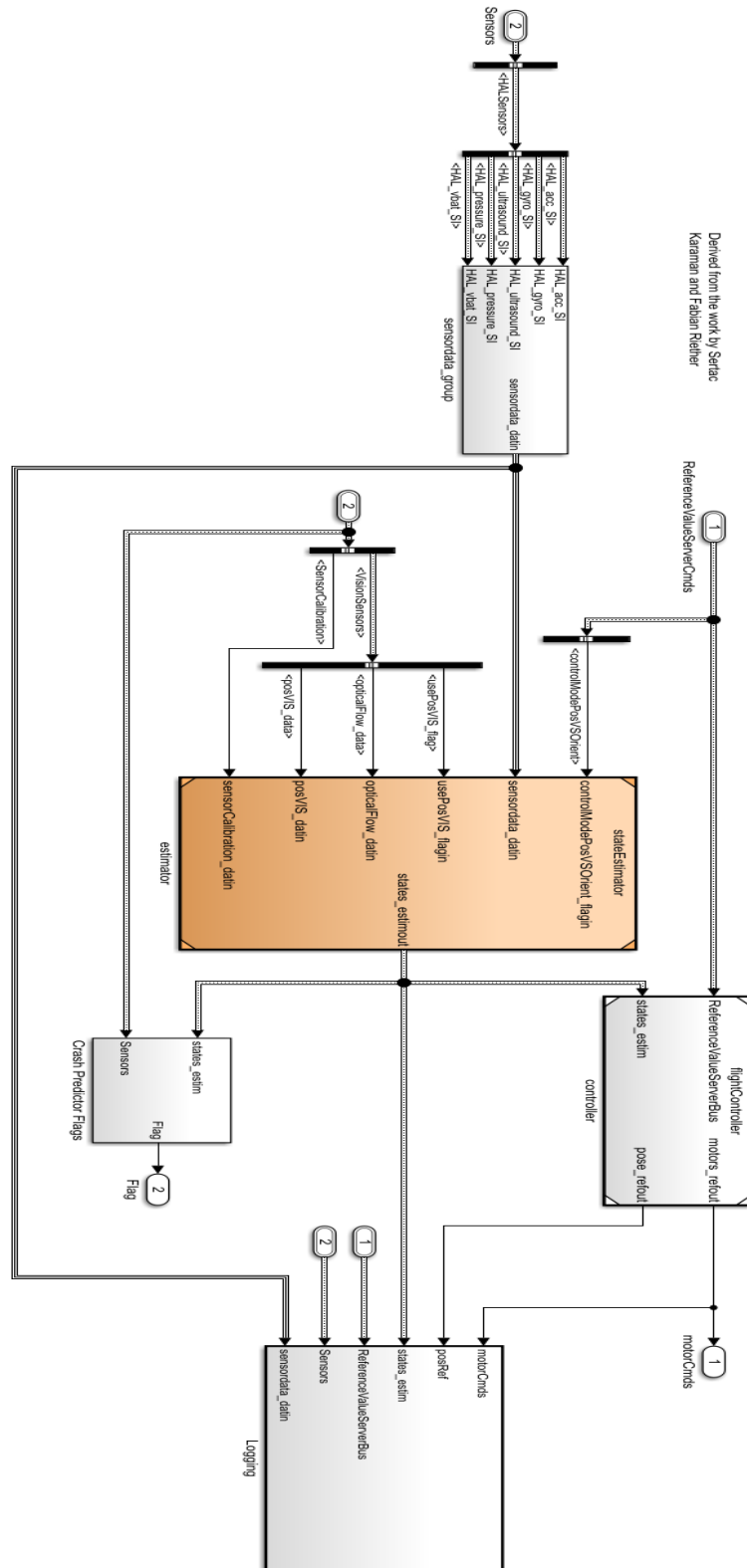


Fig. 20 – Blocco FCS

Vediamo che ci sono 5 sottoblocchi:

1. **Sensordata_group**: prende dal bus principale i valori dei sensori

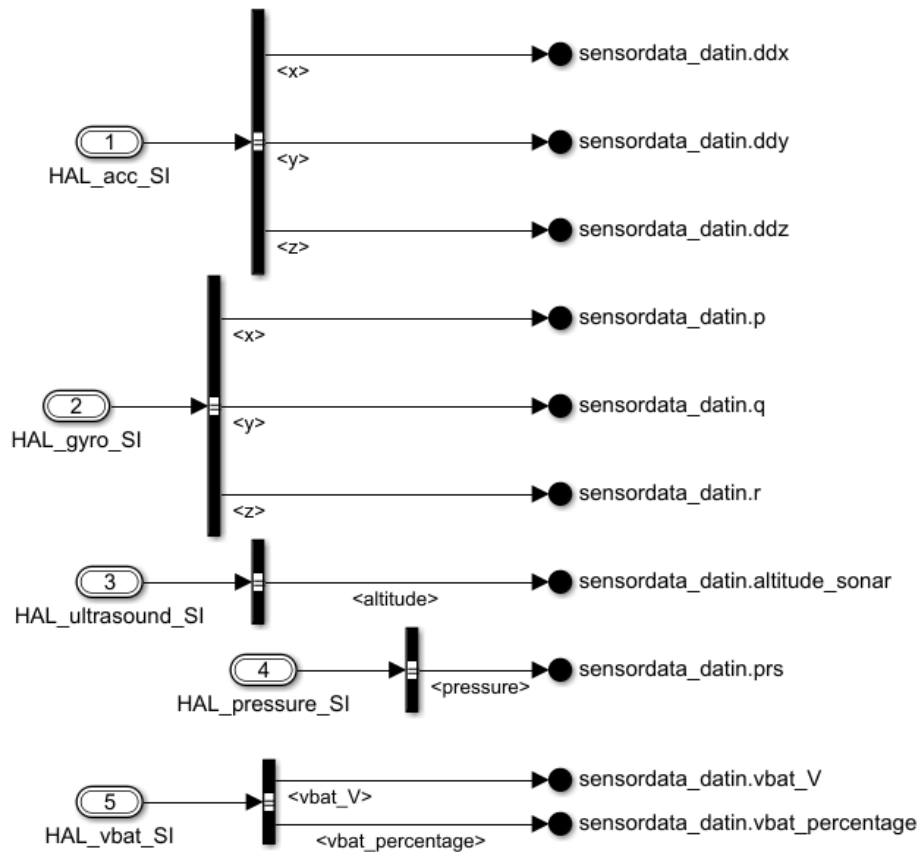


Fig. 21 - Sensori

2. **Crash Predictor Flag**: è un blocco di sicurezza per il drone. Troviamo un flag che termina la simulazione e/o spegne gli attuatori in casi di pericolo per il quadricottero;
3. **Logging**: crea dei file di log che servono per verificare il corretto andamento desiderato del drone;
4. **Estimator**: blocco mostrato in figura 22 in cui avviene la stima delle grandezze utilizzate dal controllore;

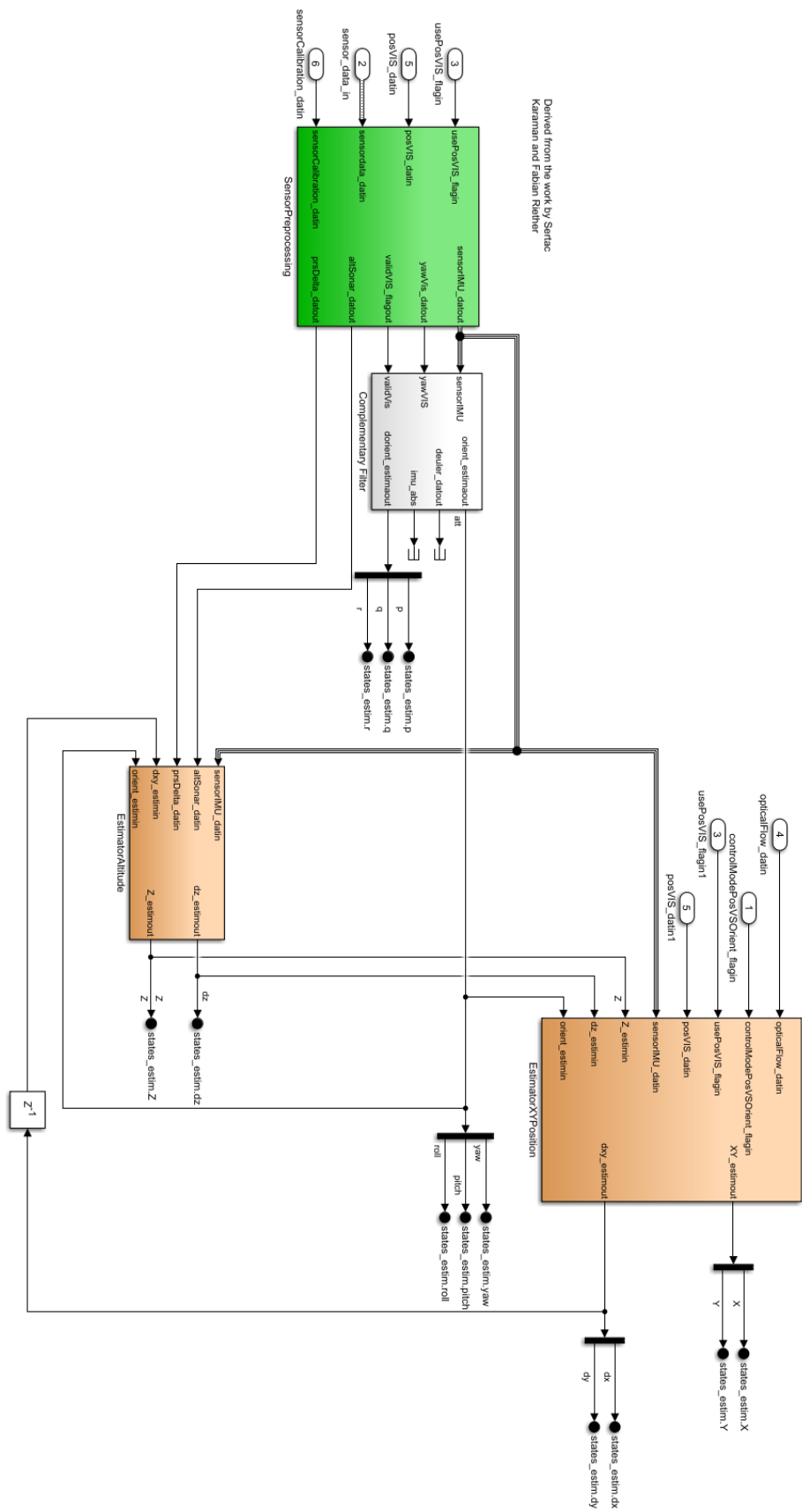


Fig. 22 – estimator

5. **Controller:** in figura 23 troviamo il suo schema:

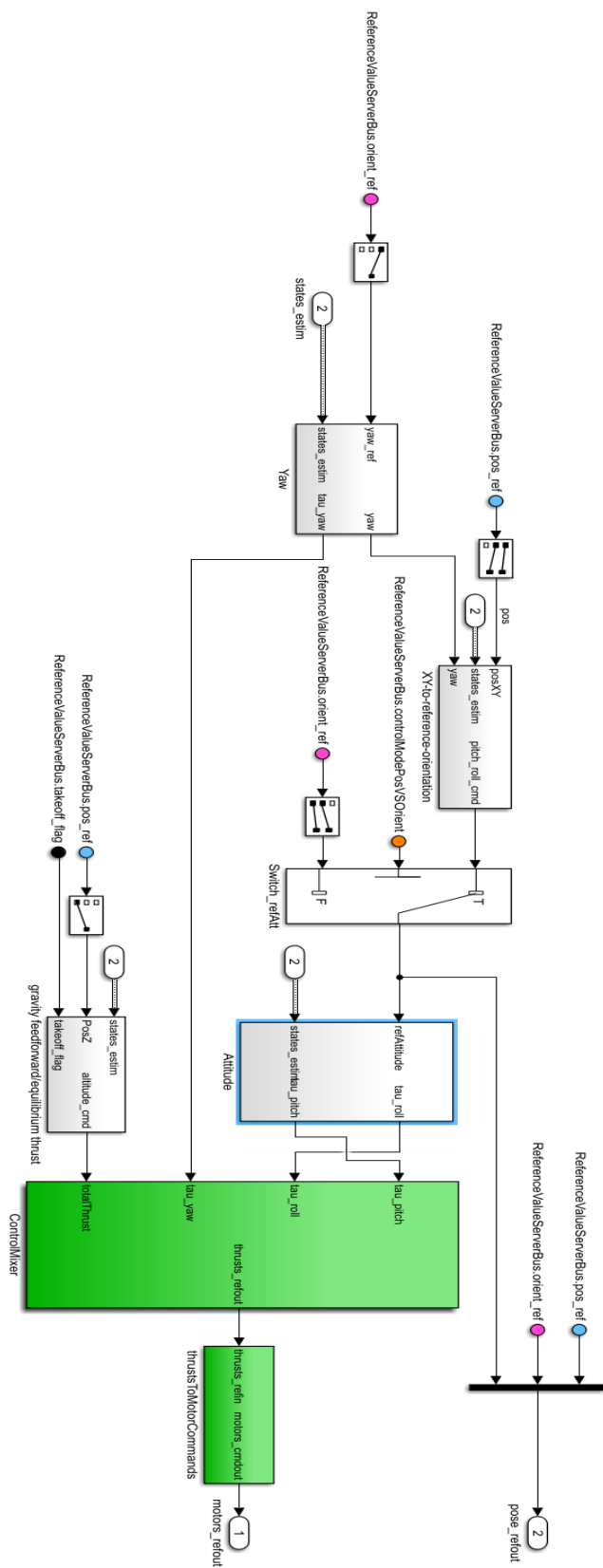
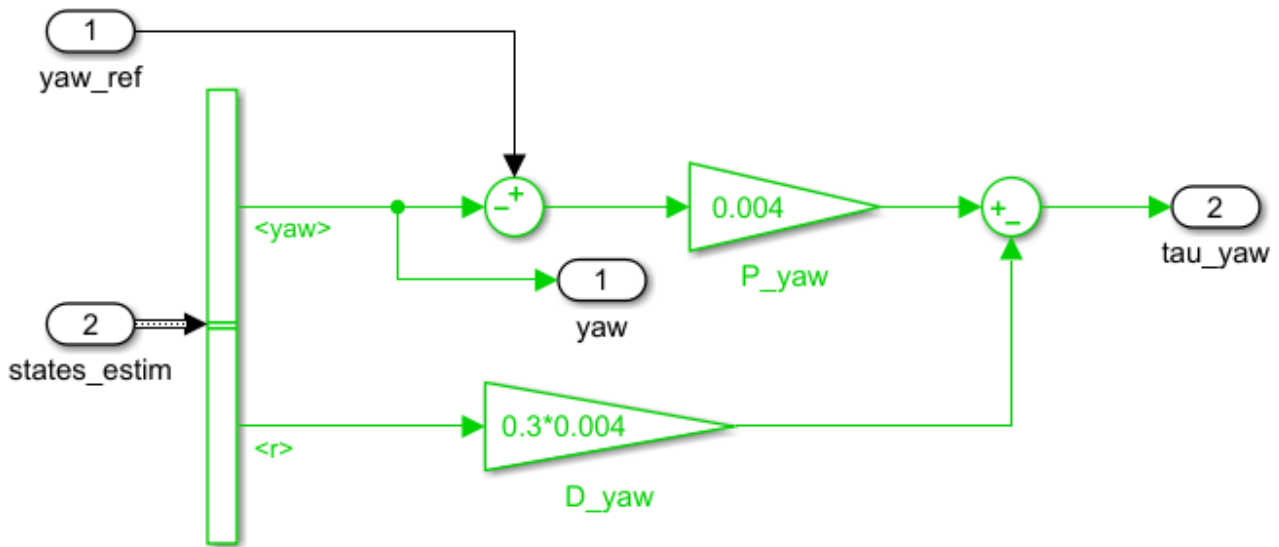


Fig. 23 – Flight Controller

All'interno del Flight Controller in figura 23 troviamo i seguenti sottoblocchi:

1. **Yaw:** abbiamo il controllore PID per l'angolo di yaw; è un PD ovvero un controllore Proporzionale-Derivativo;



2. **XY-to-reference-orientation:** come vediamo in figura 24, abbiamo un controllore PD (Proporzionale-Derivativo) per le grandezze del piano XY. L'uscita di questo blocco è un segnale doppio che va in ingresso al controllore degli angoli di pitch e di roll. L'angolo di yaw influenza tale uscita.

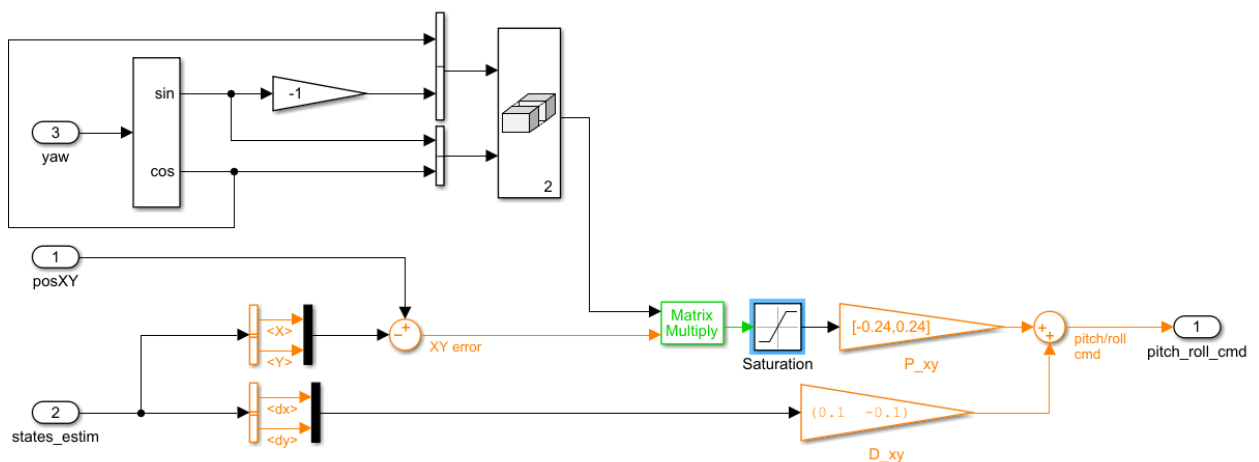


Fig. 24 – PID XY

3. **Attitude:** mostrato in figura 25, prende in ingresso la stima degli stati proveniente dal blocco estimator, considerando solo gli angoli di pitch e roll e le velocità angolari che sono rispettivamente q e p . Gli ingressi vengono selezionati tramite un busSelector. Notiamo che abbiamo un controllore PID (Proporzionale-Integrativo-Derivativo).

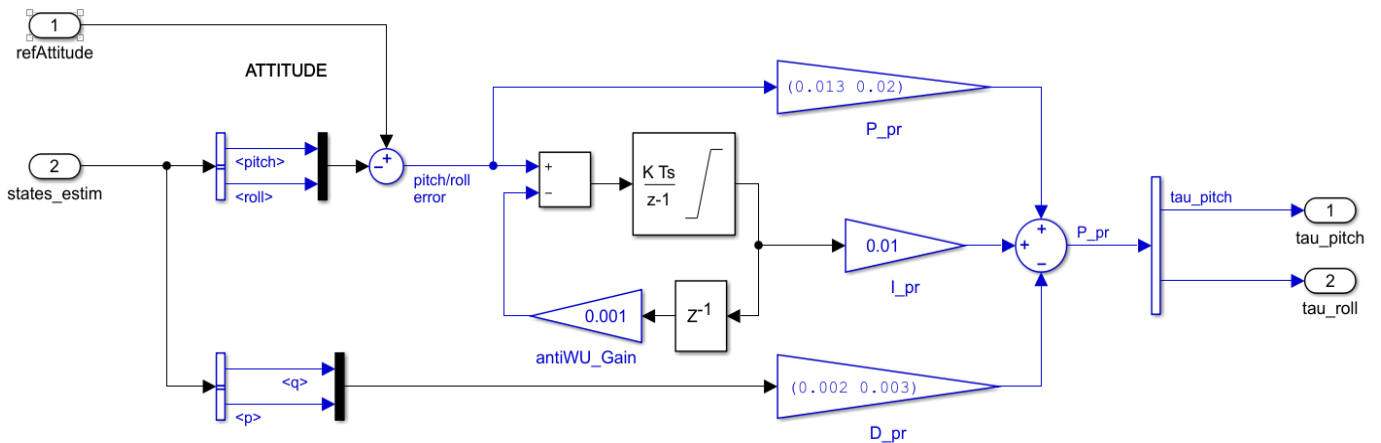


Fig. 25 – Attitude

4. **gravity feedforward / equilibrium thrust:** lo troviamo in figura 26 e contiene il controllore di tipo PD (Proporzionale-Derivativo) che agisce sulla z .

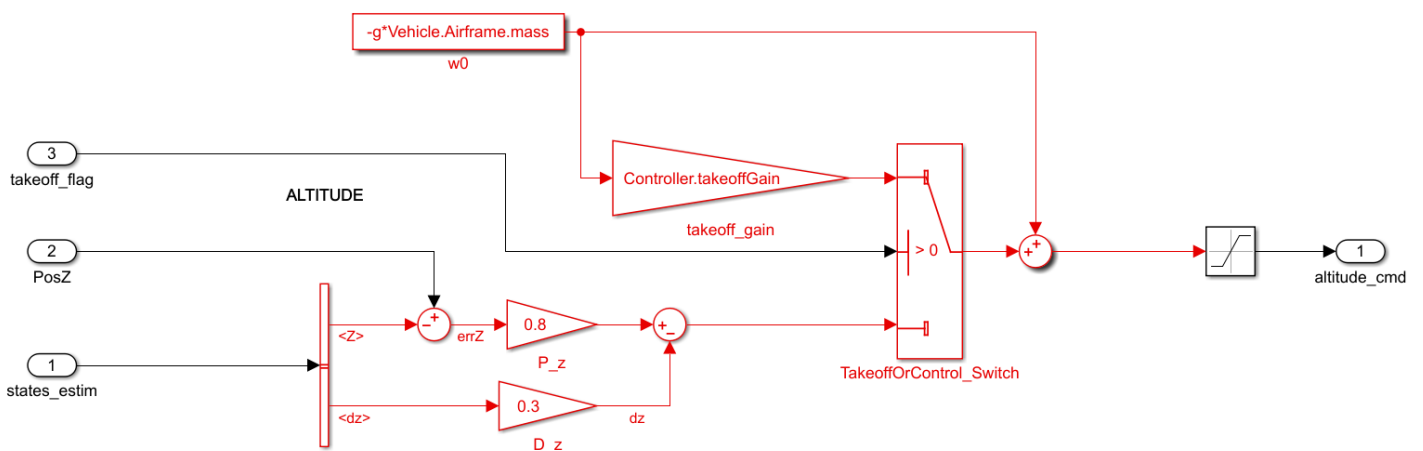


Fig. 26 - gravity feedforward / equilibrium thrust

5. **ControlMixer**: i valori uscenti da tutti i controllori vengono moltiplicati per la matrice ControllerQ2Ts. Più avanti vedremo il valore di tale matrice.

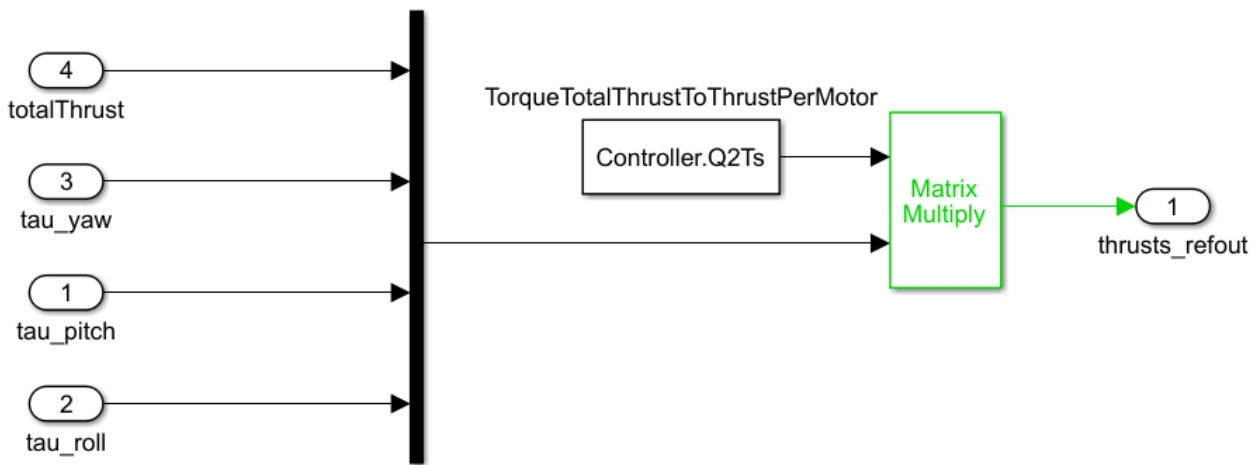


Fig. 27 – ControlMixer

6. **thrustsToMotorCommands**: Troviamo dei gain e dei saturatori che prendono in ingresso i valori grezzi uscenti dal ControlMixer. Il loro scopo è quello di proteggere i motori da eventuali sforzi non concessi.

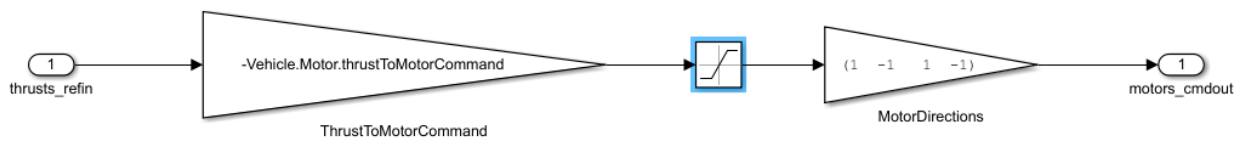


Fig. 28 – thrustToMotorCommands

10 SISTEMA DI CONTROLLO PER L'ANGOLO DI PITCH

10.1 CONTROLLO

Come detto nell'introduzione, esistono varie tipologie di droni: monocottero, tricottero, quadricottero, esacottero, octacottero, che vengono utilizzati per vari scopi. Ma, in linea generale, possiamo presupporre che il funzionamento sia simile per tutte le classificazioni.

Il mio progetto si è basato sullo studio e sullo sviluppo di un sistema di controllo per il mini-drone "Mambo" della casa Parrot. Più precisamente nello sviluppo di un controllore per l'angolo di pitch, uno dei sei gradi di libertà.

La tecnica da me utilizzata è la sintesi in frequenza.

10.2 SINTESI IN FREQUENZA

In figura 29 è riportato un generico sistema di controllo in catena chiusa. Notiamo due blocchi:

- $G(s)$ rappresenta il controllore;
- $P(s)$ rappresenta il processo.

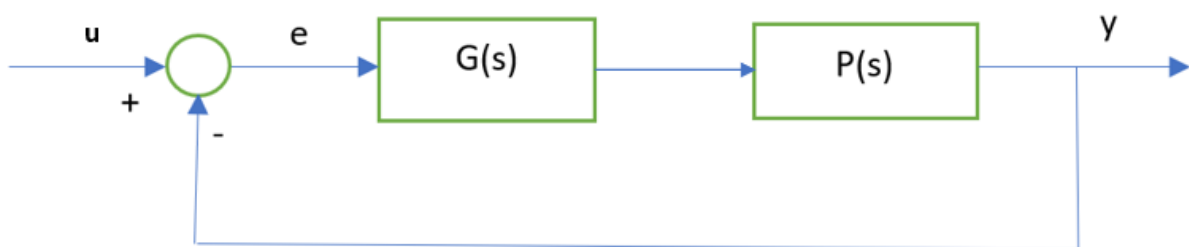


Fig. 29 – Generico sistema di controllo

La sintesi consiste nello scegliere $G(s)$ in modo tale che l'uscita y abbia determinate proprietà, cioè soddisfi alcune specifiche. Queste caratteristiche possono essere di due tipi:

-*specifiche univoche*: cioè possono essere soddisfatte in un solo modo;

-*specifiche lasche*: dipendono dalla forma.

Essa è una sintesi per tentativi; inizialmente si verifica che siano soddisfatte le specifiche univoche con un controllore di primo tentativo, successivamente si passa alle specifiche lasche. Nel caso in cui queste ultime non siano soddisfatte, si procede utilizzando una funzione compensatrice $R(s)$.

Quindi abbiamo:

$$G(s) = \frac{K_G}{s^h} R(s)$$

in cui $R(s)$ può essere o una funzione anticipatrice, o attenuatrice. Nel mio caso ho dovuto utilizzare una funzione anticipatrice, la cui formula è:

$$R_a(s) = \frac{1 + \tau_a s}{1 + \frac{\tau_a}{m_a} s}, \quad \tau_a > 0, \quad m_a > 1$$

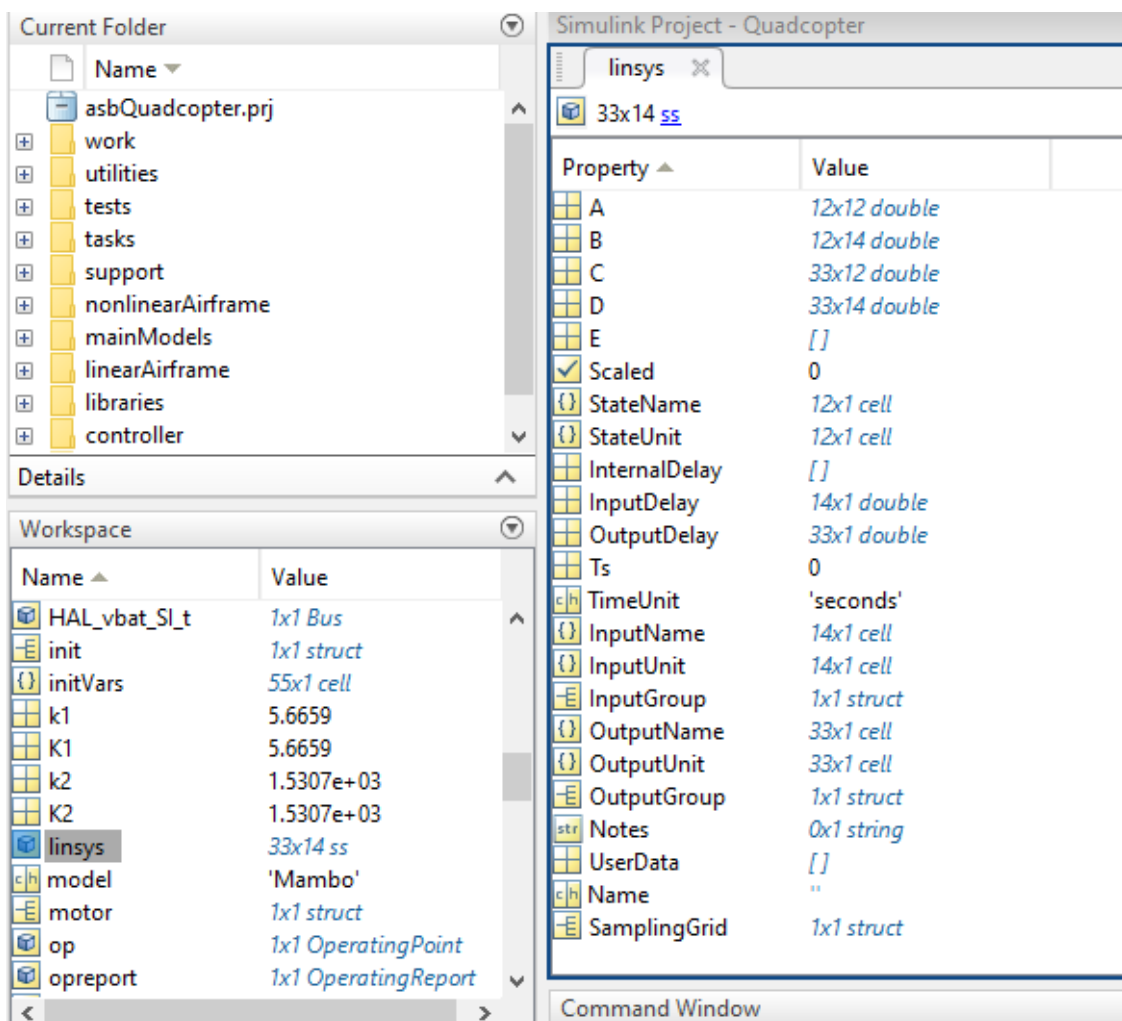
I parametri K_G e h vengono scelti in modo da soddisfare le specifiche nel regime permanente. Con $R(s)$, invece, vengono imposte le specifiche sulla risposta armonica e viene presa a guadagno unitario in modo tale che il guadagno di $G(s)$ coincida con quello di K_G .

10.3 PROCESSO

Per poter lavorare sul controllore, ho bisogno della funzione di trasferimento. Essa caratterizza il comportamento di un sistema dinamico tempo-invariante nel dominio della frequenza, mettendo in relazione l'ingresso e l'uscita.

Vediamo ora come ricavare le matrici A, B, C, D.

Nel Workspace di Matlab, alla voce linsys, troviamo le quattro matrici, come mostrato in figura 30.



The screenshot shows the MATLAB environment with the Simulink Project - Quadcopter open. The workspace contains several variables, with 'linsys' selected. The properties window for 'linsys' is displayed, showing the following table:

Property	Value
A	12x12 double
B	12x14 double
C	33x12 double
D	33x14 double
E	[]
Scaled	0
StateName	12x1 cell
StateUnit	12x1 cell
InternalDelay	[]
InputDelay	14x1 double
OutputDelay	33x1 double
Ts	0
TimeUnit	'seconds'
InputName	14x1 cell
InputUnit	14x1 cell
InputGroup	1x1 struct
OutputName	33x1 cell
OutputUnit	33x1 cell
OutputGroup	1x1 struct
Notes	0x1 string
UserData	[]
Name	''
SamplingGrid	1x1 struct

Fig. 30 – linsys

Per sapere a cosa corrispondono le righe e le colonne di A, B, C, D dobbiamo tener conto anche di StateName, InputName e OutputName.

Quindi:

- A avrà StateName come intestazione delle righe e delle colonne;
- B avrà StateName per quanto riguarda le righe e InputName per quanto riguarda le colonne;
- C avrà OutputName per quanto riguarda le righe e StateName per le colonne;
- D avrà OutputName per quanto riguarda le righe e InputName per le colonne;

Le quattro matrici della dinamica hanno le seguenti dimensioni:

- $A = 12 \times 12$ (figura 31);
- $B = 12 \times 14$ (figura 32);
- $C = 33 \times 12$ (figura 33);
- $D = 33 \times 14$ (figura 34);

dove 12 corrisponde al numero degli stati, 14 agli ingressi e 33 alle uscite.

La matrice A è la matrice della dinamica di stato del sistema; B è la matrice degli ingressi; C è la matrice delle uscite e D è la matrice di legame ingresso uscita del sistema.

Nelle successive quattro figure vedremo le matrici:

	'phi theta psi(1)'	'phi theta psi(2)'	'phi theta psi(3)'	'ub,vb,wb(1)'	'ub,vb,wb(2)'	'ub,vb,wb(3)'	'xe,ye,ze(1)'	'xe,ye,ze(2)'	'xe,ye,ze(3)'	'p,q,r(1)'	'p,q,r(2)'	'p,q,r(3)'
'phi theta psi(1)'	0	3,66E-17	0	0	0	0	0	0	0	1	3,03E-23	-5,23E-12
'phi theta psi(2)'	-3,66E-17	0	0	0	0	0	0	0	0	0	1	5,23E-12
'phi theta psi(3)'	-6,83E-13	-2,03E-28	0	0	0	0	0	0	0	0	-5,23E-12	1
'ub,vb,wb(1)'	0	-9,81	0	-3,44E-09	-3,44E-09	6,85E-13	0	0	0	-5,46E-11	0,137987461	-3,28E-13
'ub,vb,wb(2)'	9,81	-2,85E-22	0	-3,66E-09	-3,66E-09	-6,85E-13	0	0	0	-0,137987	5,81E-11	-3,52E-13
'ub,vb,wb(3)'	5,45E-11	5,45E-11	0	-6,83E-13	6,83E-13	0	0	0	0	3,29E-13	3,51E-13	0
'xe,ye,ze(1)'	1,72E-24	1,35E-19	3,29E-13	1	-1,50E-18	-5,23E-12	0	0	0	0	0	0
'xe,ye,ze(2)'	-1,35E-19	2,02E-37	3,51E-13	1,50E-18	1	5,23E-12	0	0	0	0	0	0
'xe,ye,ze(3)'	-3,29E-13	-3,51E-13	0	5,23E-12	-5,23E-12	1	0	0	0	0	0	0
'p,q,r(1)'	0	0	0	-5,76E-08	-5,76E-08	-2,73E-14	0	0	0	-2,171528	9,14E-10	4,33E-13
'p,q,r(2)'	0	0	0	4,04E-08	4,04E-08	-2,16E-14	0	0	0	6,41E-10	-1,61920486	-5,14E-13
'p,q,r(3)'	0	0	0	-1,44E-19	-1,28E-19	3,62E-19	0	0	0	1,22E-13	1,12E-13	-6,88E-09

Fig. 31 – Matrice A

	'Actuators(1)'	'Actuators(2)'	'Actuators(3)'	'Actuators(4)'	'AtmosphereBus.air_density'	'AtmosphereBus.air_temp'	'AtmosphereBus.pressure'	'AtmosphereBus.speed_sound'	'Gravity_ned(1)'	'Gravity_ned(2)'	'Gravity_ned(3)'	'MagneticField_ned(1)'	'MagneticField_ned(2)'	'MagneticField_ned(3)'
'phi theta psi(1)'	0	0	0	0	0	0	0	0	0	0	0	0	0	0
'phi theta psi(2)'	0	0	0	0	0	0	0	0	0	0	0	0	0	0
'phi theta psi(3)'	0	0	0	0	0	0	0	0	0	0	0	0	0	0
'ub,vb,wb(1)'	-6,90E-17	6,90E-17	-6,90E-17	6,90E-17	-9,93E-14	0	0	0	1	1,50E-18	5,23E-12	0	0	0
'ub,vb,wb(2)'	6,76E-17	-6,76E-17	6,76E-17	-6,76E-17	9,82E-14	0	0	-1,50E-18	1	-5,23E-12	0	0	0	0
'ub,vb,wb(3)'	-0,00960712	0,00960712	-0,00960712	0,00960712	-8,285473838	0	0	0	0	0	1	0	0	0
'xe,ye,ze(1)'	0	0	0	0	0	0	0	0	0	0	0	0	0	0
'xe,ye,ze(2)'	0	0	0	0	0	0	0	0	0	0	0	0	0	0
'xe,ye,ze(3)'	0	0	0	0	0	0	0	0	0	0	0	0	0	0
'p,q,r(1)'	0,420192	0,420192	-0,420192	-0,420192	2,53E-14	0	0	0	0	0	0	0	0	0
'p,q,r(2)'	0,31331708	-0,31331708	-0,31331708	0,31331708	1,89E-14	0	0	0	0	0	0	0	0	0
'p,q,r(3)'	-0,01154363	-0,01154363	-0,01154363	-0,01154363	-7,94E-16	0	0	0	0	0	0	0	0	0

Fig. 32 – Matrice B

	'phi theta psi(1)'	'phi theta psi(2)'	'phi theta psi(3)'	'ub,vb,wb(1)'	'ub,vb,wb(2)'	'ub,vb,wb(3)'	'xe,ye,ze(1)'	'xe,ye,ze(2)'	'xe,ye,ze(3)'	'p,q,r(1)'	'p,q,r(2)'	'p,q,r(3)'
'Accel_body(1)'	0	-9,81	0	-3,44E-09	-3,44E-09	1,84E-15	0	0	0	-5,46E-11	0,13799	7,21E-16
'Accel_body(2)'	9,81	-2,85E-22	0	-3,66E-09	-3,66E-09	-1,73E-15	0	0	0	-0,13799	5,81E-11	-7,21E-16
'Accel_body(3)'	5,45E-11	5,45E-11	0	0	0	0	0	0	0	0	0	0
'DCM_be(1)'	0	5,55E-12	0	0	0	0	0	0	0	0	0	0
'DCM_be(2)'	-5,23E-12	-5,23E-12	-1	0	0	0	0	0	0	0	0	0
'DCM_be(3)'	1,50E-18	1	-5,23E-12	0	0	0	0	0	0	0	0	0
'DCM_be(4)'	0	8,31E-30	1	0	0	0	0	0	0	0	0	0
'DCM_be(5)'	5,55E-12	0	0	0	0	0	0	0	0	0	0	0
'DCM_be(6)'	-1	1,50E-18	-5,23E-12	0	0	0	0	0	0	0	0	0
'DCM_be(7)'	0	-1	0	0	0	0	0	0	0	0	0	0
'DCM_be(8)'	1	-2,90E-23	0	0	0	0	0	0	0	0	0	0
'DCM_be(9)'	5,55E-12	5,55E-12	0	0	0	0	0	0	0	0	0	0
'Euler(1)'	1	0	0	0	0	0	0	0	0	0	0	0
'Euler(2)'	0	1	0	0	0	0	0	0	0	0	0	0
'Euler(3)'	0	0	1	0	0	0	0	0	0	0	0	0
'LLA(1)'	0	0	0	0	0	0	9,00E-06	0	0	0	0	0
'LLA(2)'	0	0	0	0	0	0	0	1,21E-05	0	0	0	0
'LLA(3)'	0	0	0	0	0	0	0	0	-1	0	0	0
'Omega_body(1)'	0	0	0	0	0	0	0	0	0	1	0	0
'Omega_body(2)'	0	0	0	0	0	0	0	0	0	0	1	0
'Omega_body(3)'	0	0	0	0	0	0	0	0	0	0	0	1
'V_body(1)'	0	0	0	1	0	0	0	0	0	0	0	0
'V_body(2)'	0	0	0	0	1	0	0	0	0	0	0	0
'V_body(3)'	0	0	0	0	0	1	0	0	0	0	0	0
'V_ned(1)'	1,72E-24	1,35E-19	3,29E-13	1	-1,50E-18	-5,23E-12	0	0	0	0	0	0
'V_ned(2)'	-1,35E-19	2,02E-37	3,51E-13	1,50E-18	1	5,23E-12	0	0	0	0	0	0
'V_ned(3)'	-3,29E-13	-3,51E-13	0	5,23E-12	-5,23E-12	1	0	0	0	0	0	0
'X_ned(1)'	0	0	0	0	0	0	1	0	0	0	0	0
'X_ned(2)'	0	0	0	0	0	0	0	1	0	0	0	0
'X_ned(3)'	0	0	0	0	0	0	0	0	1	0	0	0
'dOmega_body(1)'	0	0	0	-5,76E-08	-5,76E-08	-2,73E-14	0	0	0	-2,17153	9,14E-10	4,33E-13
'dOmega_body(2)'	0	0	0	4,04E-08	4,04E-08	-2,16E-14	0	0	0	6,41E-10	-1,6192	-5,14E-13
'dOmega_body(3)'	0	0	0	-1,44E-19	-1,28E-19	3,62E-19	0	0	0	1,22E-13	1,12E-13	-6,88E-09

Fig. 33 – Matrice C

	'phi theta psi(1)'	'phi theta psi(2)'	'phi theta psi(3)'	'ub,vb,wb(1)'	'ub,vb,wb(2)'	'ub,vb,wb(3)'	'xe,ye,ze(1)'	'xe,ye,ze(2)'	'xe,ye,ze(3)'	'p,q,r(1)'	'p,q,r(2)'	'p,q,r(3)'
'Accel_body(1)'	0	-9,81	0	-3,44E-09	-3,44E-09	1,84E-15	0	0	0	-5,46E-11	0,13799	7,21E-16
'Accel_body(2)'	9,81	-2,85E-22	0	-3,66E-09	-3,66E-09	-1,73E-15	0	0	0	-0,13799	5,81E-11	-7,21E-16
'Accel_body(3)'	5,45E-11	5,45E-11	0	0	0	0	0	0	0	0	0	0
'DCM_be(1)'	0	5,55E-12	0	0	0	0	0	0	0	0	0	0
'DCM_be(2)'	-5,23E-12	-5,23E-12	-1	0	0	0	0	0	0	0	0	0
'DCM_be(3)'	1,50E-18	1	-5,23E-12	0	0	0	0	0	0	0	0	0
'DCM_be(4)'	0	8,31E-30	1	0	0	0	0	0	0	0	0	0
'DCM_be(5)'	5,55E-12	0	0	0	0	0	0	0	0	0	0	0
'DCM_be(6)'	-1	1,50E-18	-5,23E-12	0	0	0	0	0	0	0	0	0
'DCM_be(7)'	0	-1	0	0	0	0	0	0	0	0	0	0
'DCM_be(8)'	1	-2,90E-23	0	0	0	0	0	0	0	0	0	0
'DCM_be(9)'	5,55E-12	5,55E-12	0	0	0	0	0	0	0	0	0	0
'Euler(1)'	1	0	0	0	0	0	0	0	0	0	0	0
'Euler(2)'	0	1	0	0	0	0	0	0	0	0	0	0
'Euler(3)'	0	0	1	0	0	0	0	0	0	0	0	0
'LLA(1)'	0	0	0	0	0	0	9,00E-06	0	0	0	0	0
'LLA(2)'	0	0	0	0	0	0	0	1,21E-05	0	0	0	0
'LLA(3)'	0	0	0	0	0	0	0	0	-1	0	0	0
'Omega_body(1)'	0	0	0	0	0	0	0	0	0	1	0	0
'Omega_body(2)'	0	0	0	0	0	0	0	0	0	0	1	0
'Omega_body(3)'	0	0	0	0	0	0	0	0	0	0	0	1
'V_body(1)'	0	0	0	1	0	0	0	0	0	0	0	0
'V_body(2)'	0	0	0	0	1	0	0	0	0	0	0	0
'V_body(3)'	0	0	0	0	0	1	0	0	0	0	0	0
'V_ned(1)'	1,72E-24	1,35E-19	3,29E-13	1	-1,50E-18	-5,23E-12	0	0	0	0	0	0
'V_ned(2)'	-1,35E-19	2,02E-37	3,51E-13	1,50E-18	1	5,23E-12	0	0	0	0	0	0
'V_ned(3)'	-3,29E-13	-3,51E-13	0	5,23E-12	-5,23E-12	1	0	0	0	0	0	0
'X_ned(1)'	0	0	0	0	0	0	1	0	0	0	0	0
'X_ned(2)'	0	0	0	0	0	0	0	1	0	0	0	0
'X_ned(3)'	0	0	0	0	0	0	0	0	1	0	0	0
'dOmega_body(1)'	0	0	0	-5,76E-08	-5,76E-08	-2,73E-14	0	0	0	-2,17153	9,14E-10	4,33E-13
'dOmega_body(2)'	0	0	0	4,04E-08	4,04E-08	-2,16E-14	0	0	0	6,41E-10	-1,6192	-5,14E-13
'dOmega_body(3)'	0	0	0	-1,44E-19	-1,28E-19	3,62E-19	0	0	0	1,22E-13	1,12E-13	-6,88E-09

Fig. 34 – Matrice D

Le matrici sono abbastanza corpose, dato che contengono tutte le variabili di stato, tutti gli ingressi e tutte le uscite disponibili. Il mio progetto di controllore si basa sull'angolo di pitch, quindi ho scorporato le matrici interessate, considerando i 4 motori come ingresso, il pitch e la q (velocità angolare) come variabili di stato, per poi andare a vedere quali uscite vengono coinvolte.

Le sottomatrici ottenute sono le seguenti:

$$\begin{aligned} A &= \begin{bmatrix} 0 & 1.0000 \\ 0 & -1.6190 \end{bmatrix} \\ B &= \begin{bmatrix} 0 \\ 0.3133 \end{bmatrix} \\ C &= \begin{bmatrix} -9.8100 & 0.1380 \\ 1.0000 & 0 \\ 0 & 1.0000 \end{bmatrix} \\ D &= \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \end{aligned}$$

Fig. 35 – Sottomatrici A, B, C, D

Dobbiamo considerare anche due coefficienti, K1 e K2, che verranno moltiplicati per il processo.

Con il comando `Controller.Q2Ts` otteniamo in output una matrice (figura 36) che ci fornisce la partizione tra il nostro segnale e i motori. La prima colonna corrisponde al thrust, la seconda allo yaw, la terza al pitch e la quarta al roll. Quindi, lavorando sul pitch, il mio K1 avrà come valore 5,6659.

```
>> Controller.Q2Ts  
  
ans =  
  
    0.2500    103.5736   -5.6659   -5.6659  
    0.2500   -103.5736   -5.6659    5.6659  
    0.2500    103.5736    5.6659    5.6659  
    0.2500   -103.5736    5.6659   -5.6659
```

Fig. 36 – Coefficiente K1

Invece, con il comando `Vehicle.Motor.thrustToMotorCommand` otteniamo in output il coefficiente relativo al K2 (figura 37), che è uguale a 1530,7.

```
>> Vehicle.Motor.thrustToMotorCommand  
  
ans =  
  
    1.5307e+03
```

Fig. 37 – Coefficiente K2

Una volta raccolti tutti i valori necessari, si prosegue con il calcolo della funzione di trasferimento e successivamente del processo, ottenendo:

```
processo =  
  
      2717  
-----  
s^2 + 1.619 s
```

Fig. 38 - Processo

Il matlab mette a disposizione una funzione chiamata “sisotool” in cui, per la parte che ci riguarda, vengono tracciati i diagrammi degli strumenti utili per lo studio della sintesi in frequenza, come Bode, Nichols, Nyquist, luogo delle radici, etc.

Lanciando il sisotool in questa fase della progettazione, impostando il processo come in figura 38 e un controllore a guadagno unitario, otteniamo una soluzione di questo tipo:

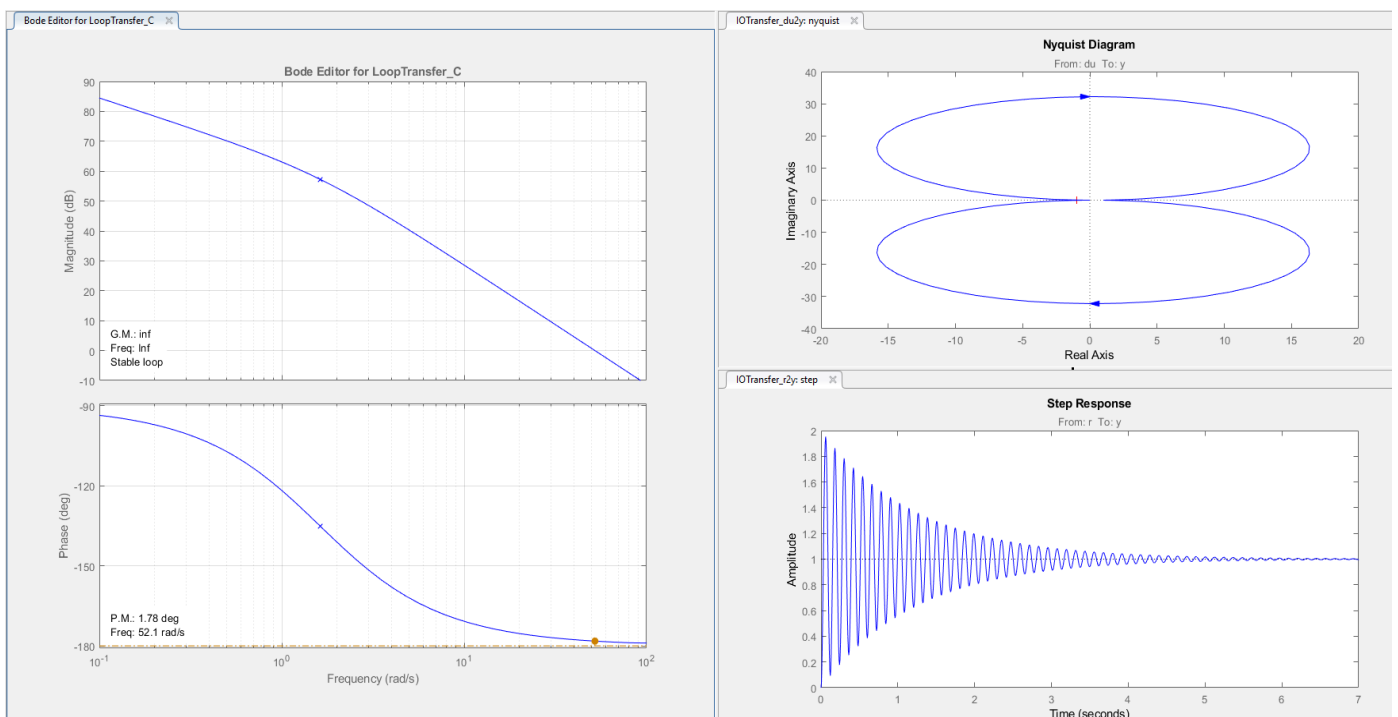


Fig. 39 – Sisotool processo

Nei due riquadri a sinistra troviamo i diagrammi di Bode (in alto quello per il modulo, in basso quello per la fase), mentre nella parte destra troviamo in alto il diagramma di Nyquist e in basso la risposta del sistema. Basta vedere quest'ultimo riquadro per capire che l'uscita del sistema è instabile. Bisogna quindi procedere con la progettazione di un nuovo controllore.

10.4 PROGETTAZIONE DEL CONTROLLORE

Come detto nel paragrafo della sintesi in frequenza, il controllore deve rispettare alcune specifiche, in questo caso da me imposte.

Esse sono:

- Sistema di tipo 1;
- Errore $\leq 0,01$;
- Margine di risonanza $\leq 2\text{dB}$;
- Banda passante = 2Hz.

Traducendole otteniamo:

- Margine di fase $=47^\circ$;
- Una pulsazione di attraversamento $\omega_t=10$

Dopo aver fatto un primo tentativo, impostando un guadagno $K_g = 0,037$, vediamo che le specifiche lasche non sono soddisfatte:

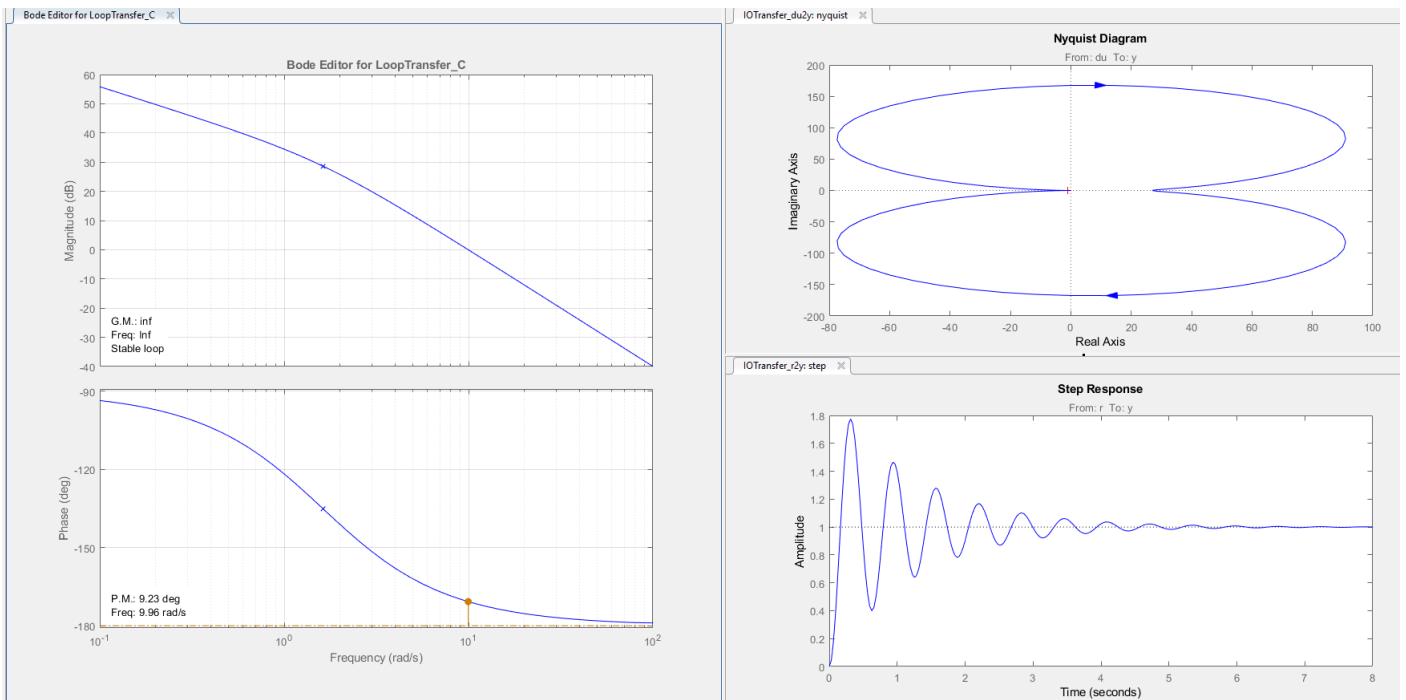


Fig. 40 – Ambiente sisotool dopo un primo tentativo

Procedo quindi con un secondo tentativo utilizzando una funzione anticipatrice.

L'azione o correzione anticipatrice consiste nell'introdurre, tramite il controllore $G(s)$, un anticipo di fase.

Utilizzando gli appositi diagrammi universali per la funzione compensatrice e la relativa formula, si ottiene la seguente situazione, in figura 41:

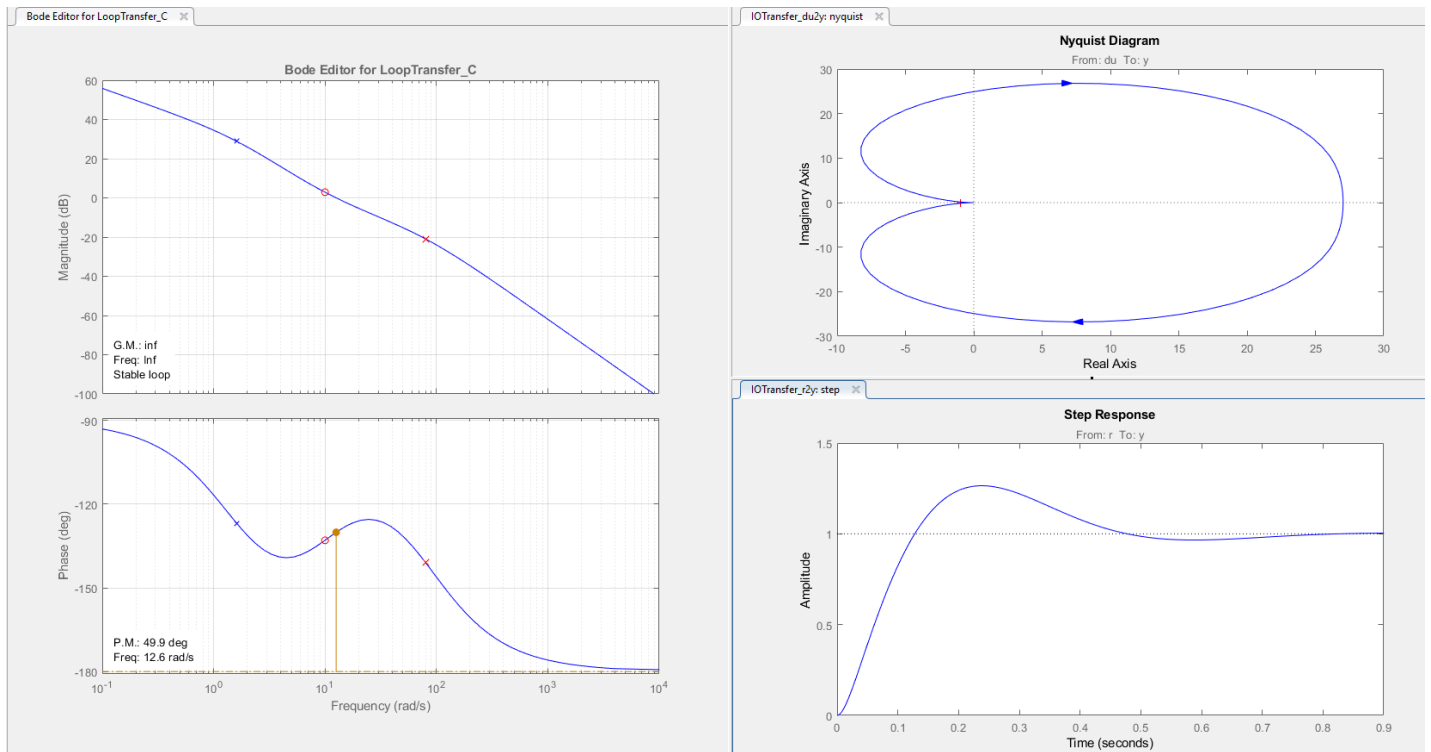


Fig. 41 – Ambiente sisotool con nuovo controllore implementato

Le specifiche sono soddisfatte e l'uscita del sistema è stabile.

Una volta progettato il controllore, si può procedere con la sostituzione del vecchio con il nuovo.

11 PID

Dato che, l'obiettivo del mio lavoro è stato quello di progettare un controllore per l'angolo di pitch che andasse a sostituire il PID originale, di seguito spiegherò cos'è il PID.

Il controllo Proporzionale-Integrale-Derivativo, abbreviato come PID, è un sistema in retroazione negativa ampiamente impiegato nei sistemi di controllo.

Il controllore acquisisce in ingresso un valore da un processo e lo confronta con un valore di riferimento. La differenza, il cosiddetto segnale di errore, viene quindi usata per determinare il valore della variabile di uscita del controllore, che è la variabile manipolabile del processo.

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).

Le tre azioni di un PID vengono calcolate separatamente e poi sommate algebricamente:

$$u = u_P + u_I + u_D$$

Azione proporzionale (P)

L'azione proporzionale è ottenuta moltiplicando il segnale d'errore "e" con un'opportuna costante:

$$u_P = K_P \cdot e(t)$$

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

Azione integrale (I)

L'azione integrale è proporzionale all'integrale nel tempo del segnale di errore "e", moltiplicato per la costante K_I

$$u_I = K_I \cdot \int e(t) dt$$

Questa definizione dell'azione integrale fa sì che il controllore abbia memoria dei valori passati del segnale d'errore; in particolare, il valore dell'azione integrale non è necessariamente nullo se è nullo il segnale d'errore. Questa proprietà dà al PID la capacità di portare il processo esattamente al punto di riferimento richiesto, dove la sola azione proporzionale risulterebbe nulla.

Azione derivativa (D)

Viene aggiunta per migliorare le prestazioni del controllore:

$$u_D = K_D \cdot \frac{de(t)}{dt}$$

Lo scopo è quello di compensare rapidamente le variazioni del segnale di errore: se vediamo che "e" sta aumentando, l'azione derivativa cerca di compensare questa deviazione in ragione della sua velocità di cambiamento, senza aspettare che l'errore diventi significativo (azione proporzionale) o che persista per un certo tempo (azione integrale).

12 PID ORIGINALE

Dalla figura 23, entrando nel blocco Attitude possiamo vedere, in figura 42, il controllore originario che è presente sul modello che è un PID così implementato:

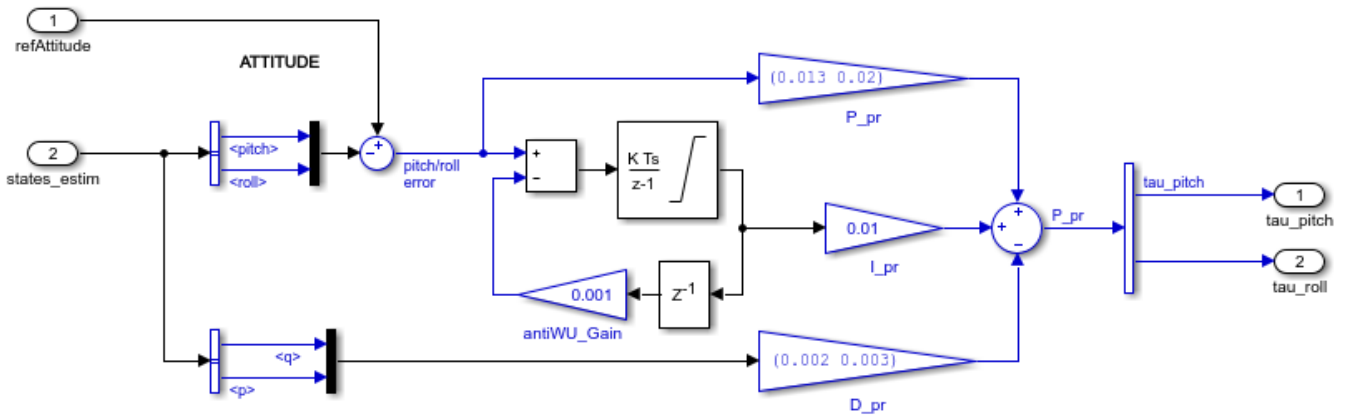


Fig. 42 – PID originale

Notiamo che avviene sia il controllo per l'angolo di pitch, sia per l'angolo di roll, quindi abbiamo due uscite: tau_pitch e tau_roll.

Anche gli ingressi sono due:

- Da "states_estim" si ottengono le variabili di stato che provengono dal blocco "estimator" in figura 22. Da questo blocco escono 12 output, quindi tramite un BusSelector vengono selezionate solo le variabili che vanno in input al PID, che sono pitch e roll e le rispettive velocità angolari q e p.
- Da "refAttitude" si ottiene il riferimento proveniente dalla figura 43 e che passando successivamente nella 44 arriva in ingresso al PID.

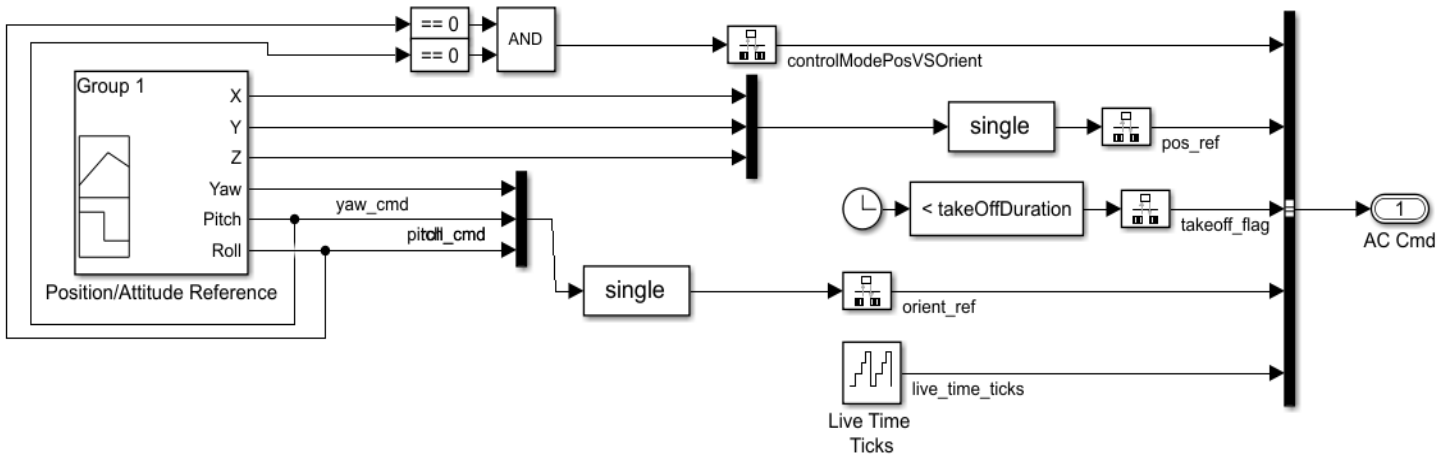


Fig. 43 – Riferimenti iniziali

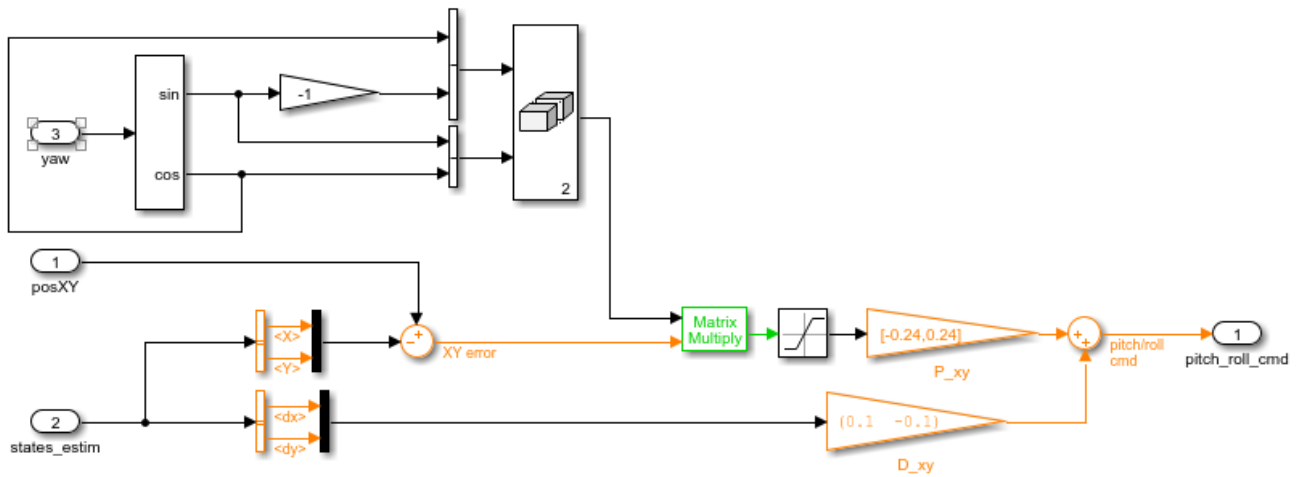


Fig. 44 – Uscita blocco XY-to-reference-orientation e conseguente entrata nel blocco Attitude

13 CONTROLLORE IN FREQUENZA

Andando a sostituire il PID originale con il controllore da me progettato otteniamo:

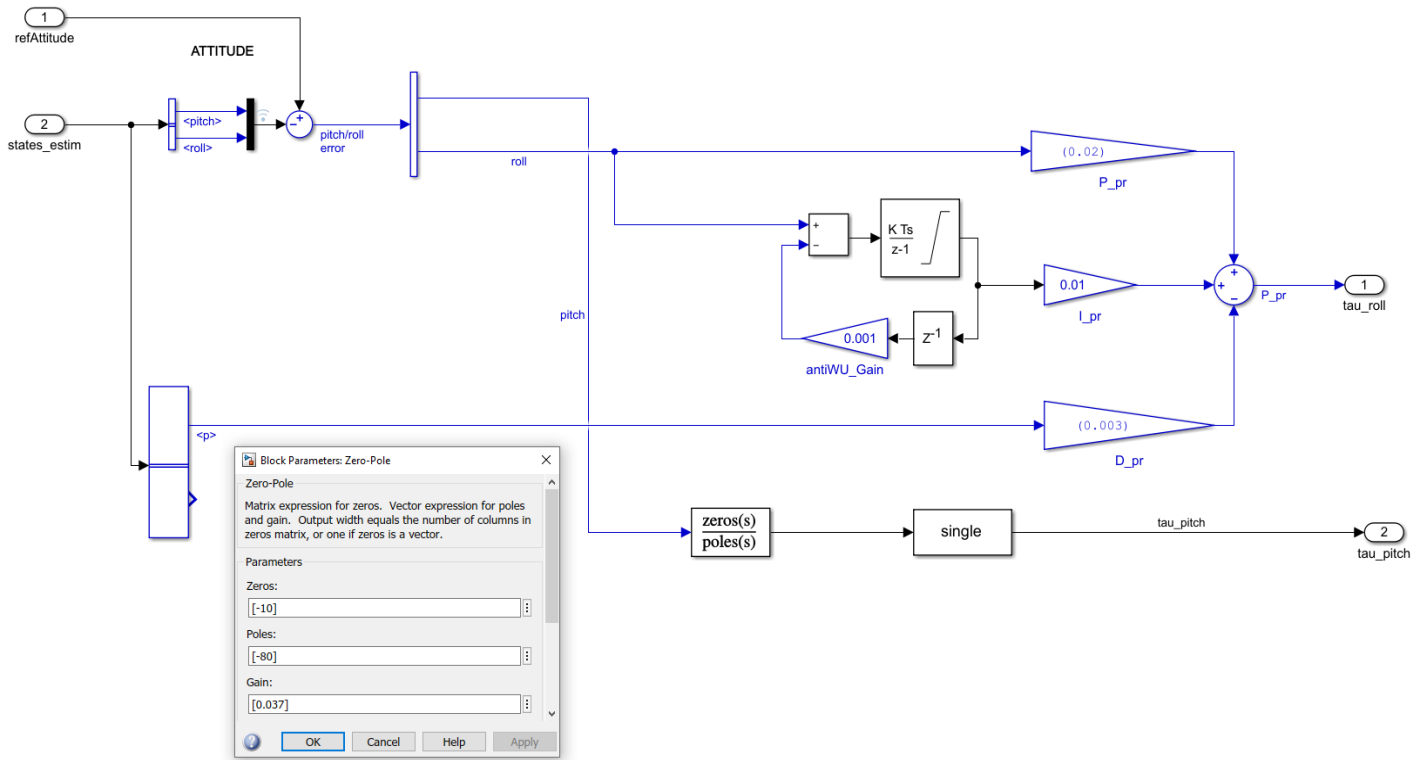


Fig. 45 – Diagramma a blocchi del nuovo controllore con indicazione del guadagno, del polo e dello zero

I due ingressi sono ricavati con gli stessi passaggi del PID originale.

Per quanto riguarda l'angolo di roll viene lasciato il PID, mentre per il pitch viene utilizzato il controllore da me progettato. Tramite un demux, prendo il segnale del pitch, proveniente dalla differenza tra il "refAttitude" e lo "states_estim" e lo mando in ingresso al controllore. Nel riquadro della figura 45 viene mostrato il polo, lo zero e il guadagno del controllore.

In figura 46 vediamo i riferimenti delle variabili. Nel mio caso il pitch è impostato ad un valore pari a 0 radianti.

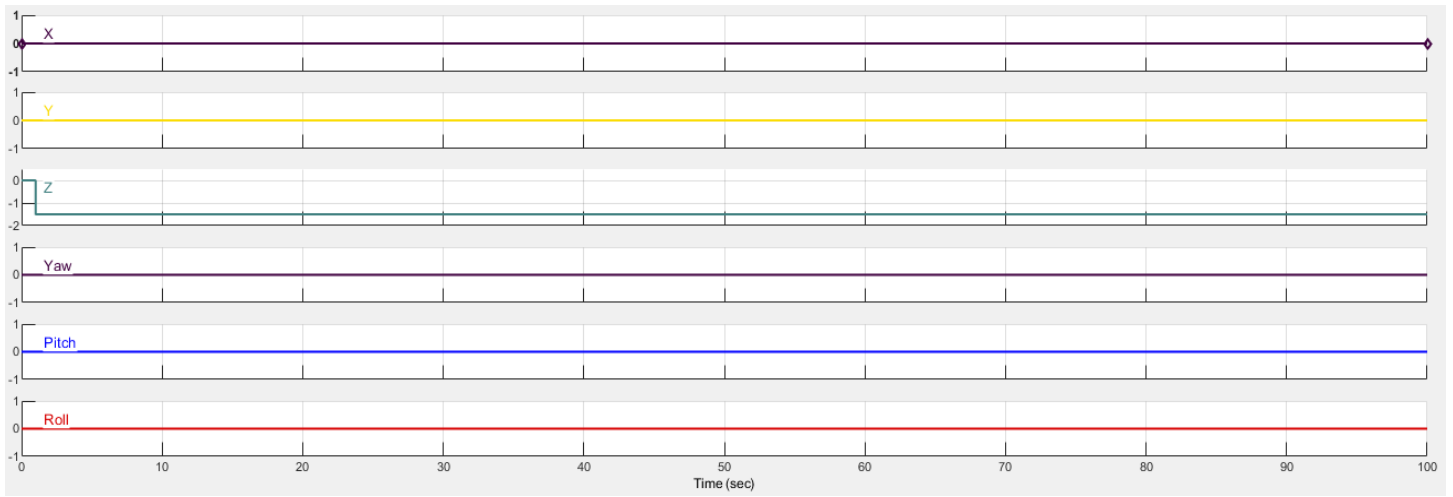


Fig. 46 - Riferimenti

14 CONTROLLORE PID ORIGINALE NEL CASO LINEARE

Per la simulazione nel caso lineare con il PID originale, bisogna digitare il comando “VSS_VEHICLE=0;” sulla Command window di Matlab, in modo tale da passare dalla modalità non lineare, che è quella impostata di default, a quella lineare.

Si ottengono i seguenti risultati:

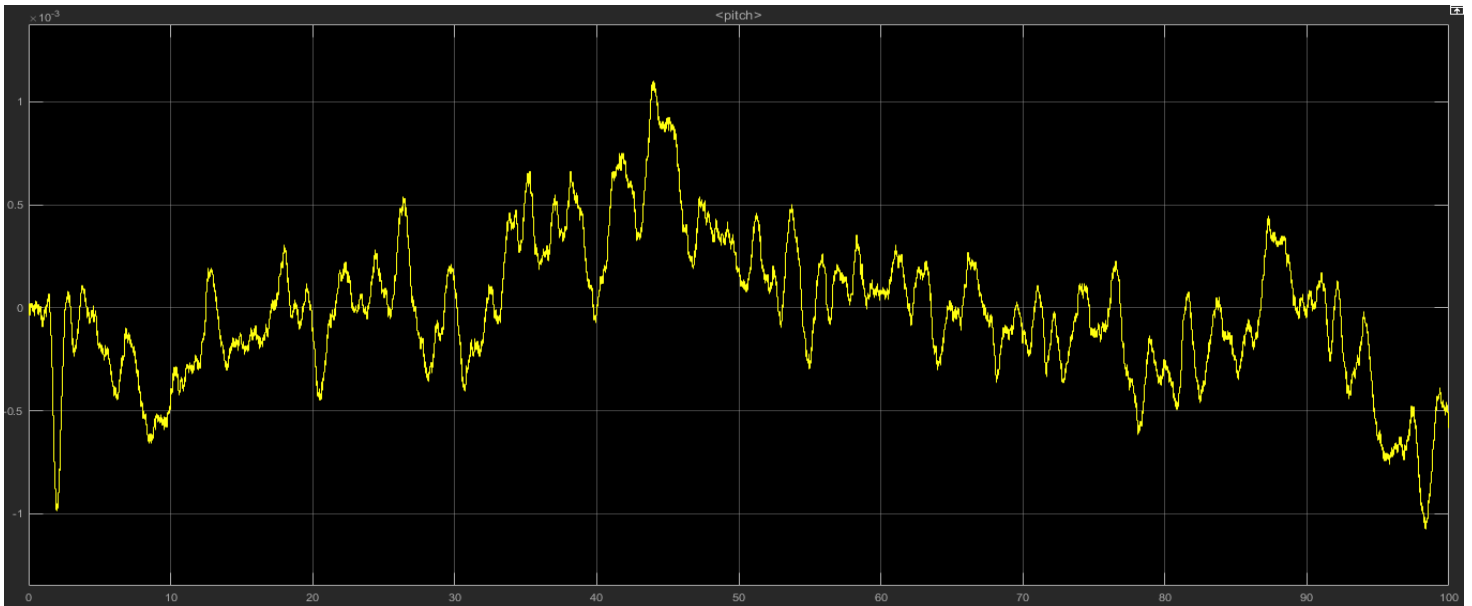


Fig. 47 – pitch

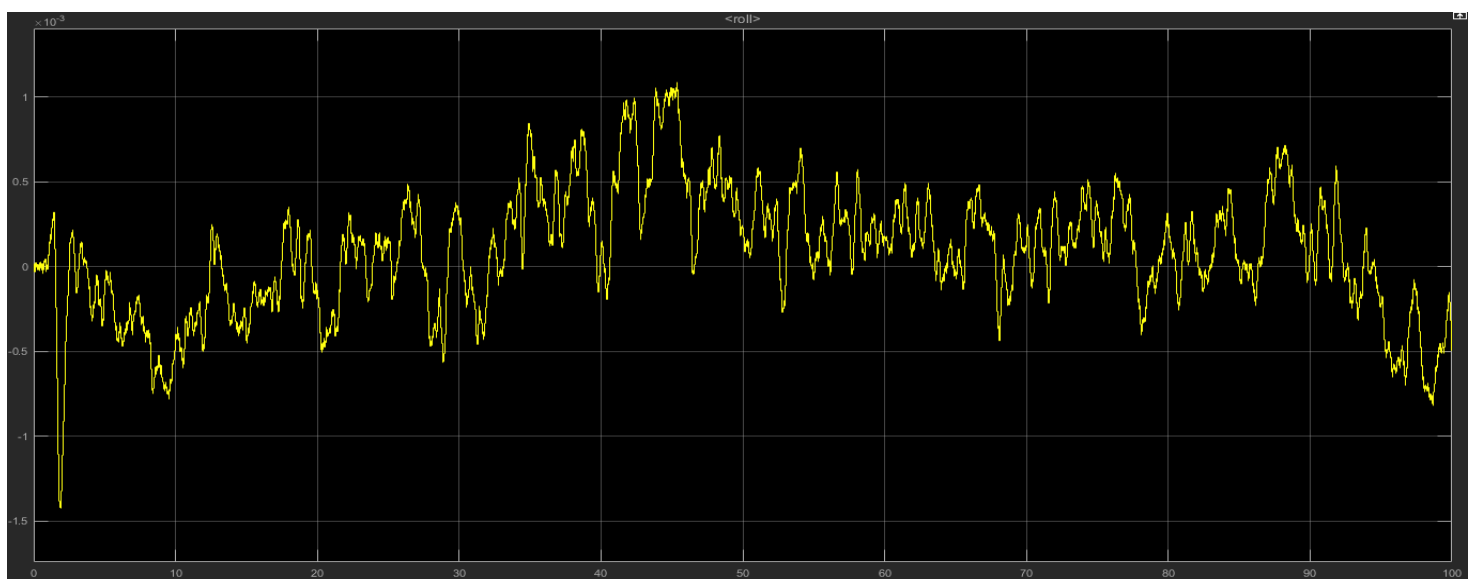


Fig. 48 – roll

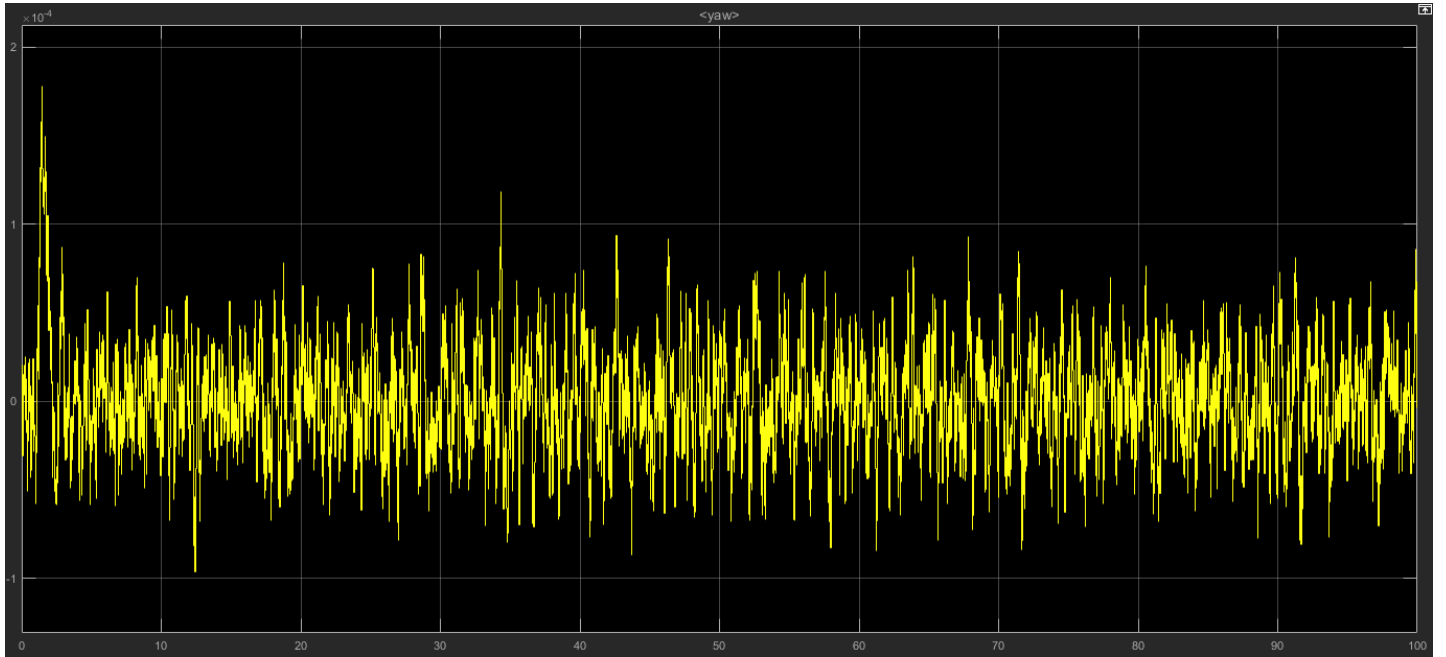


Fig. 49 – yaw

Notiamo che le oscillazioni delle uscite di pitch, roll e yaw risultano molto basse. Essendo dell'ordine di 10^{-3} il loro andamento assume valori molto vicini allo zero.

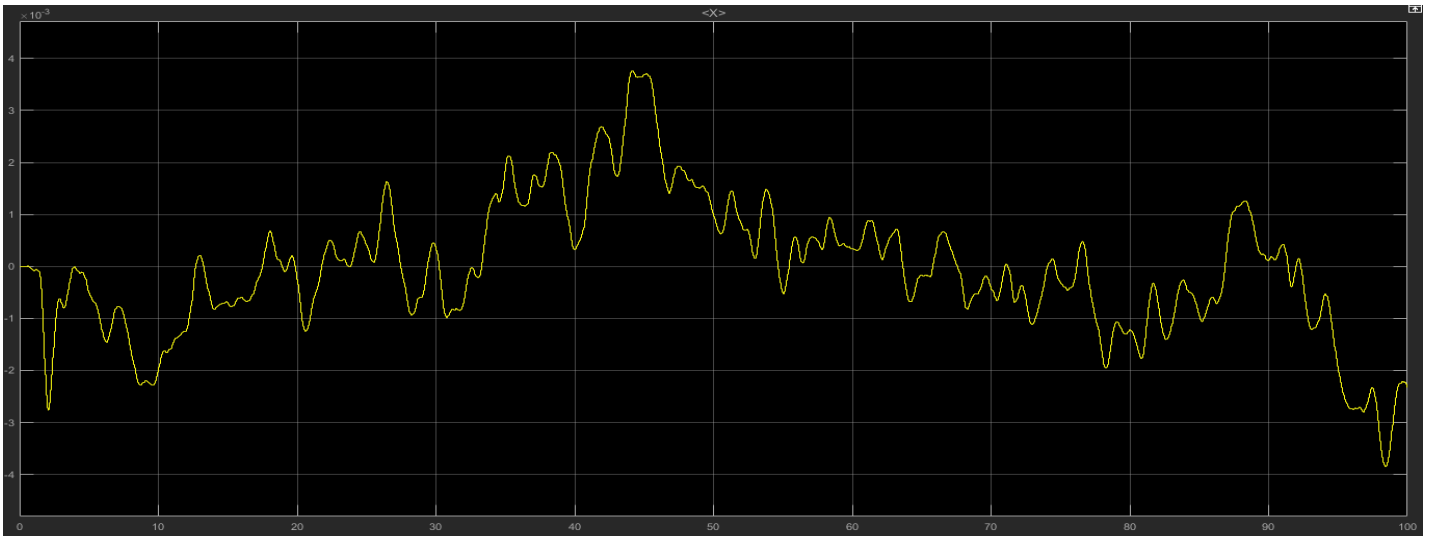


Fig. 50 – x

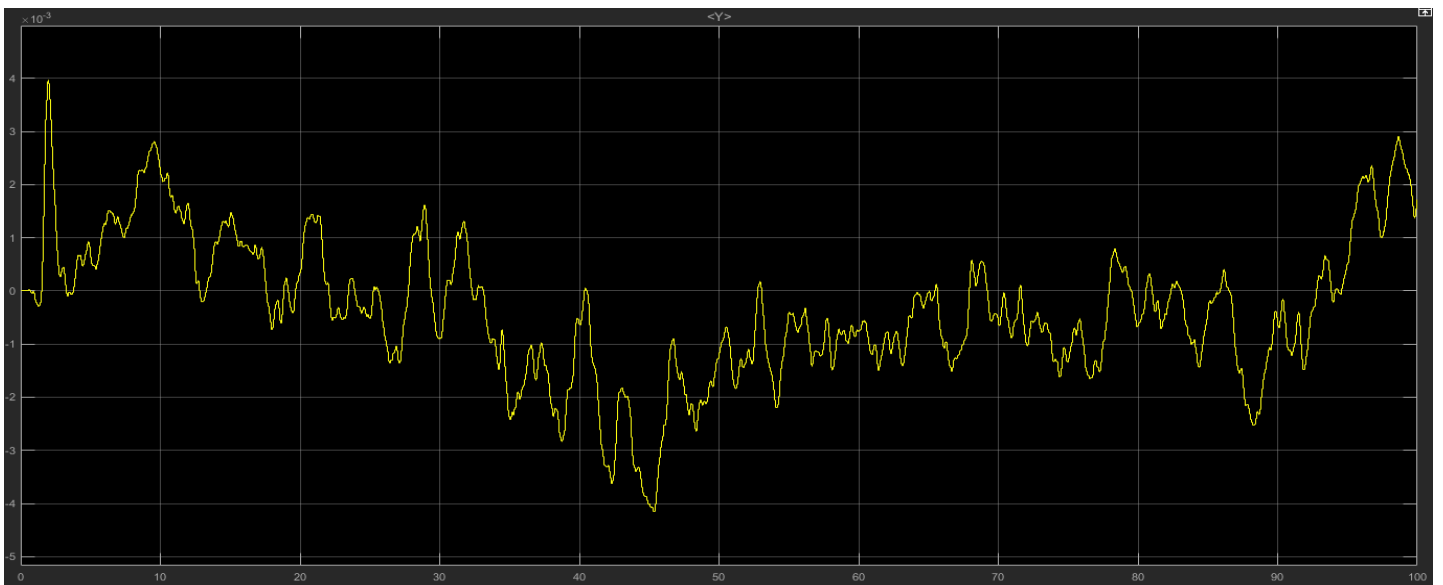


Fig. 51 – y

Anche le oscillazioni di x e y vanno bene, in quanto sono dell'ordine di 10^{-3} , quindi molto piccole.

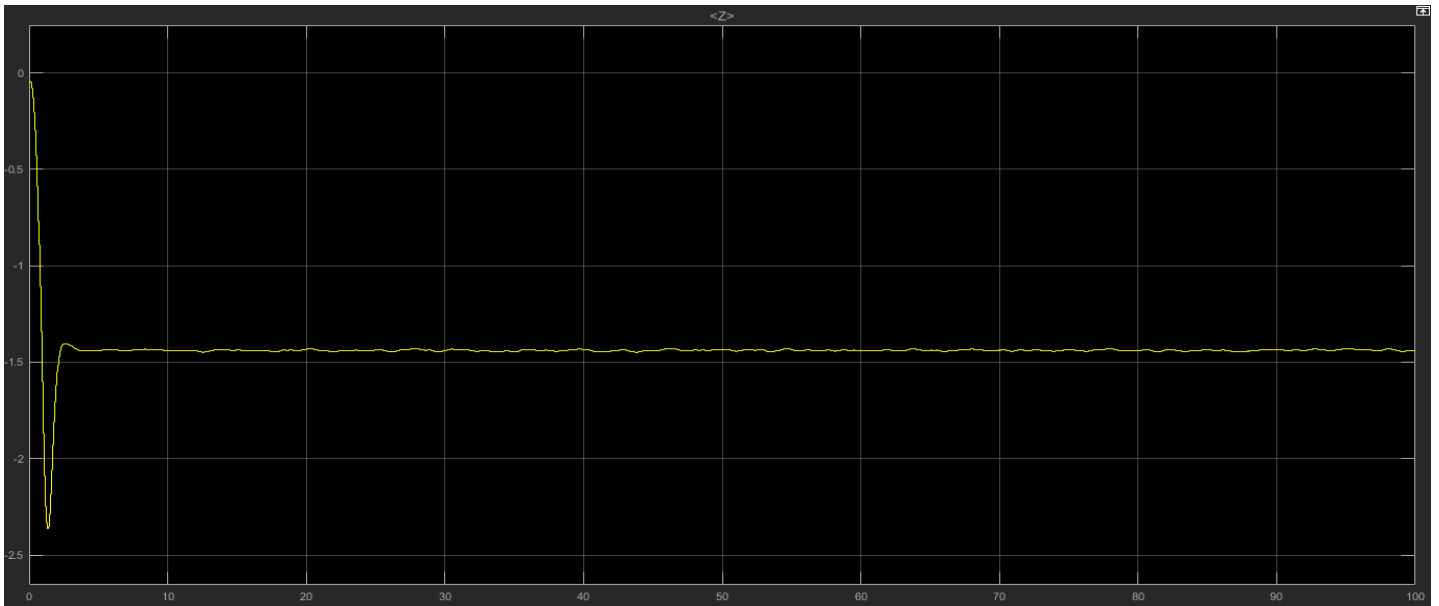


Fig. 52 - z

Grazie alla figura 27, che rappresenta l'altezza, notiamo che il drone rimane costante ad una quota fissa pare a 1.5 metri, che è il riferimento datogli in ingresso.

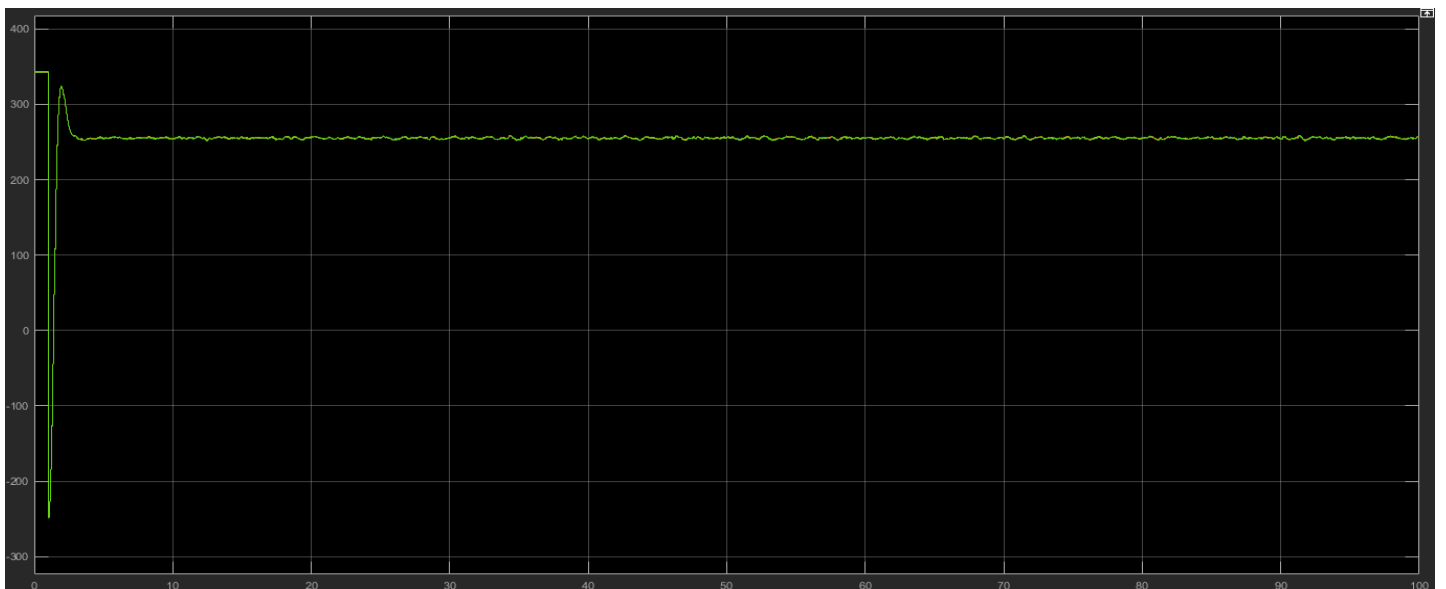


Fig. 53 – Saturazione dei motori

15 CONTROLLORE PID ORIGINALE NEL CASO NON LINEARE

Lanciando la simulazione nel caso non lineare, che è quello impostato di default, otteniamo i seguenti grafici con risultati e comportamenti pressappoco uguali a quelli del caso lineare.

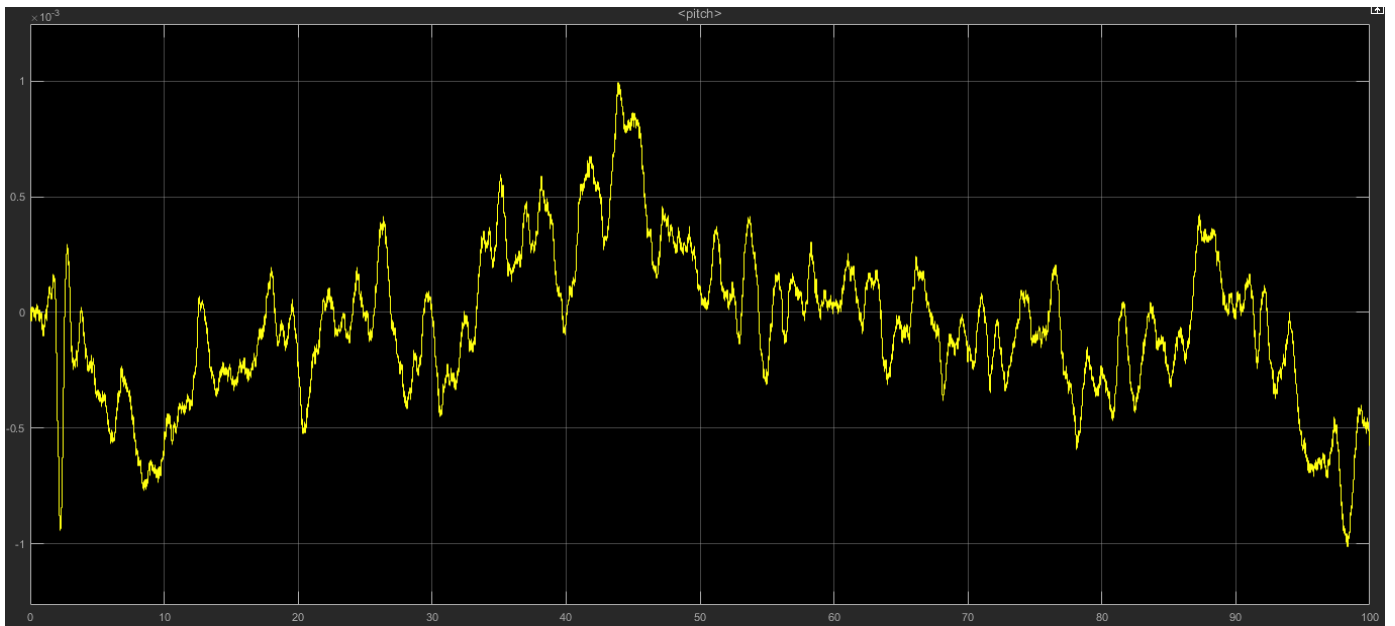


Fig. 54 – pitch

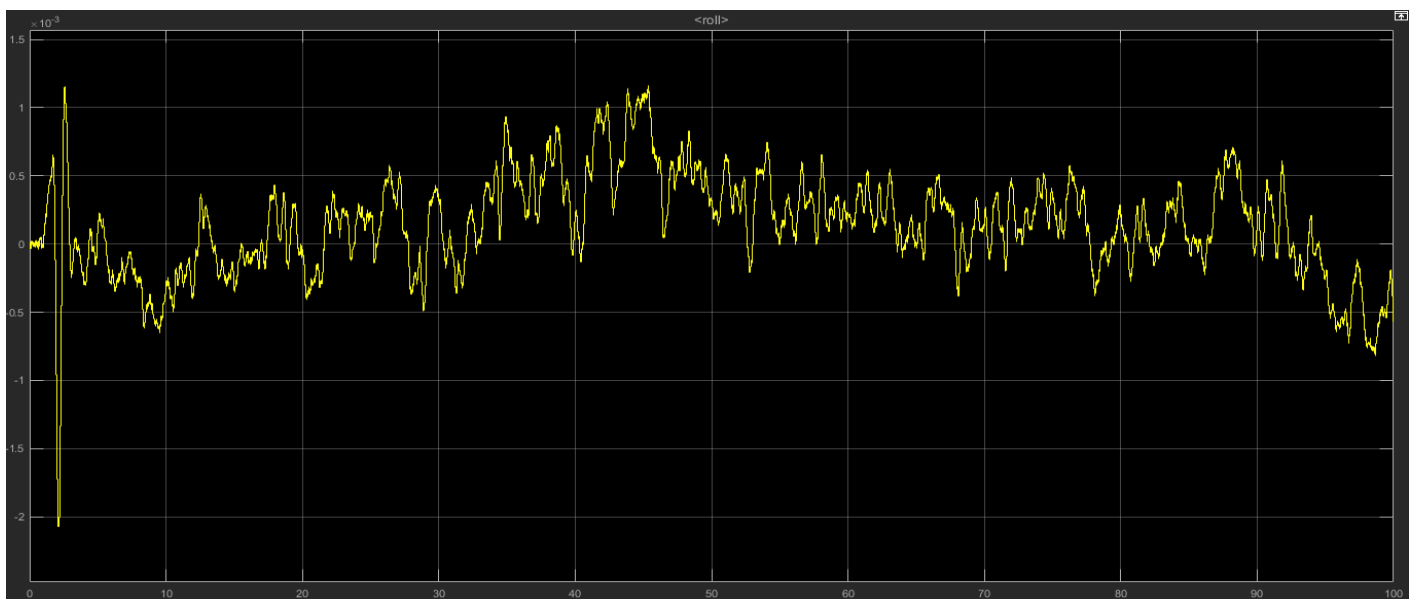


Fig. 55 – roll

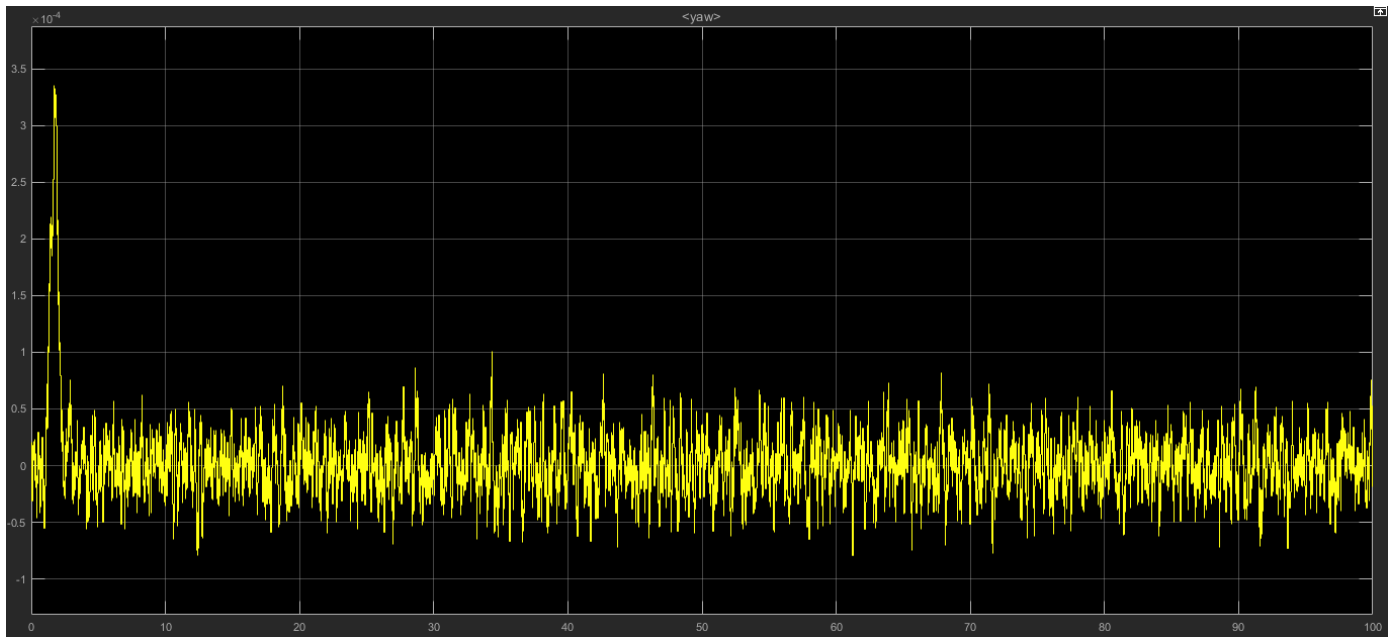


Fig. 56 – yaw

Anche qui notiamo che le oscillazioni delle uscite di pitch, roll e yaw risultano molto basse. Essendo dell'ordine di 10^{-3} il loro andamento assume valori molto vicini allo zero.

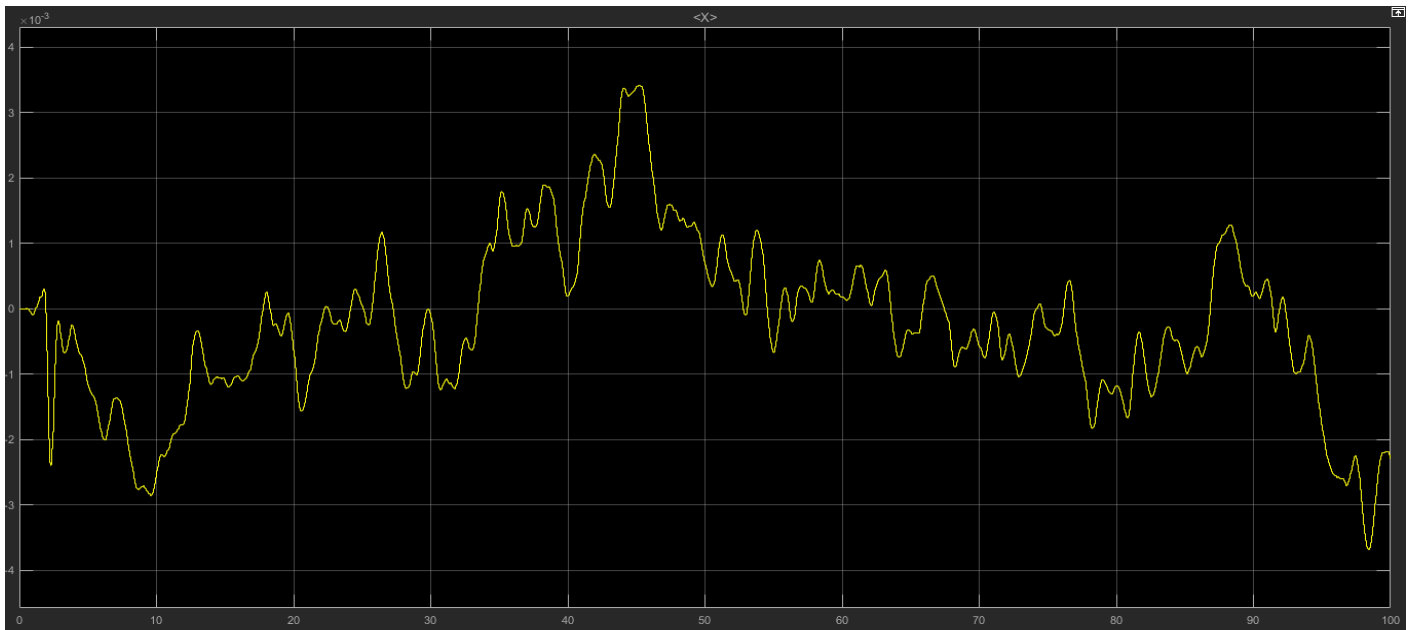


Fig. 57 – x

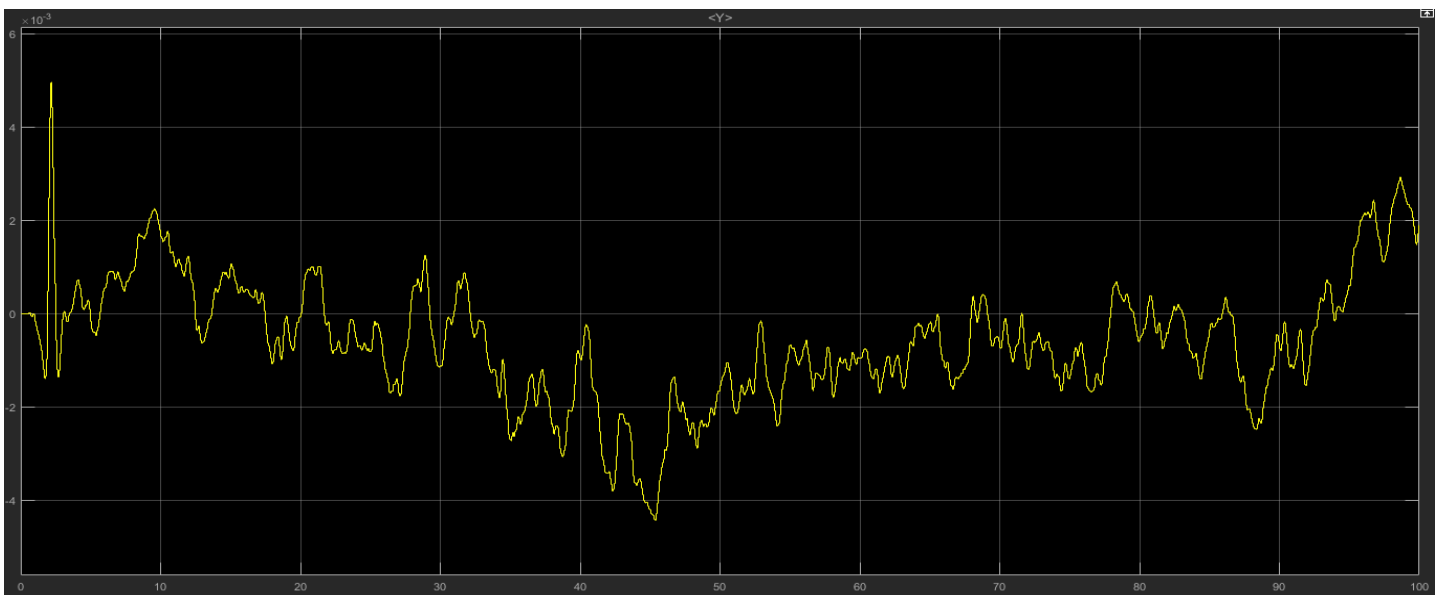


Fig. 58 – y

Anche le oscillazioni di x e y vanno bene, in quanto sono dell'ordine di 10^{-3} , quindi molto piccole.

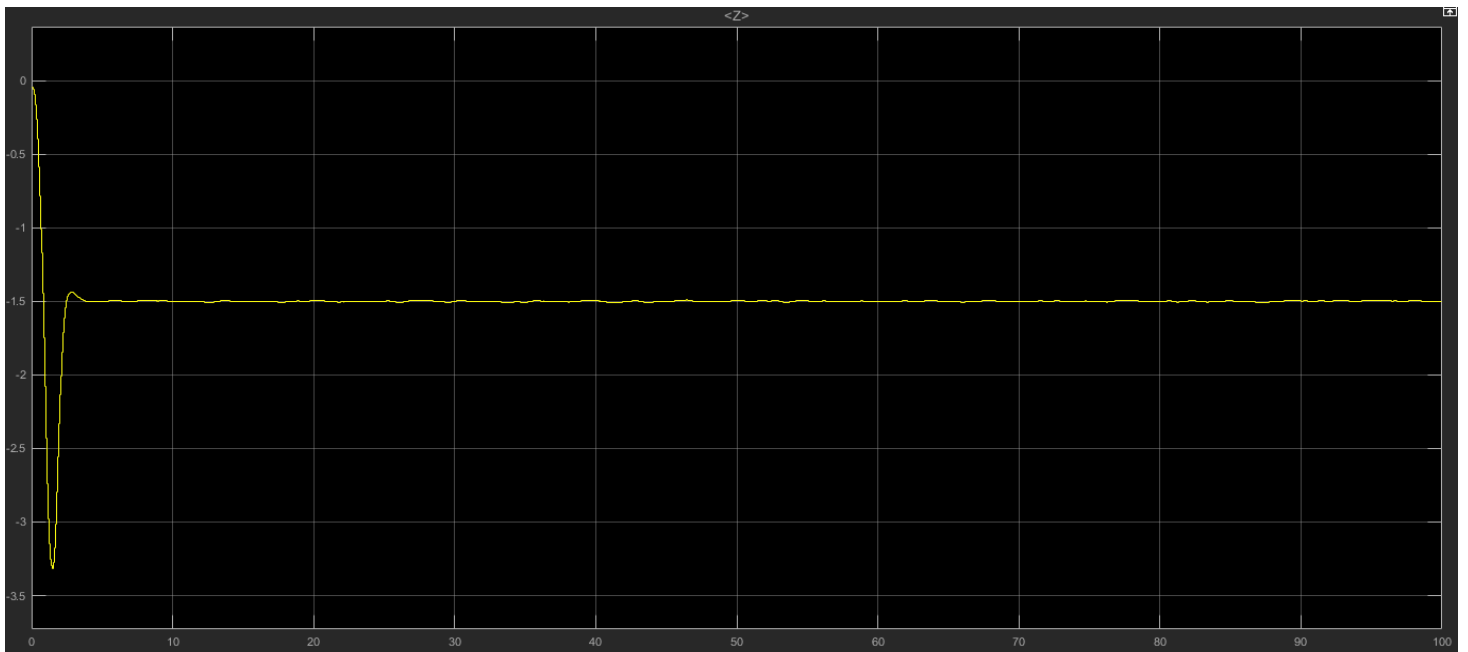


Fig. 59 – z

Notiamo che la quota si stabilizza ad un'altezza di 1.5 metri dal suolo, rispettando quindi quanto indicato nel riferimento.

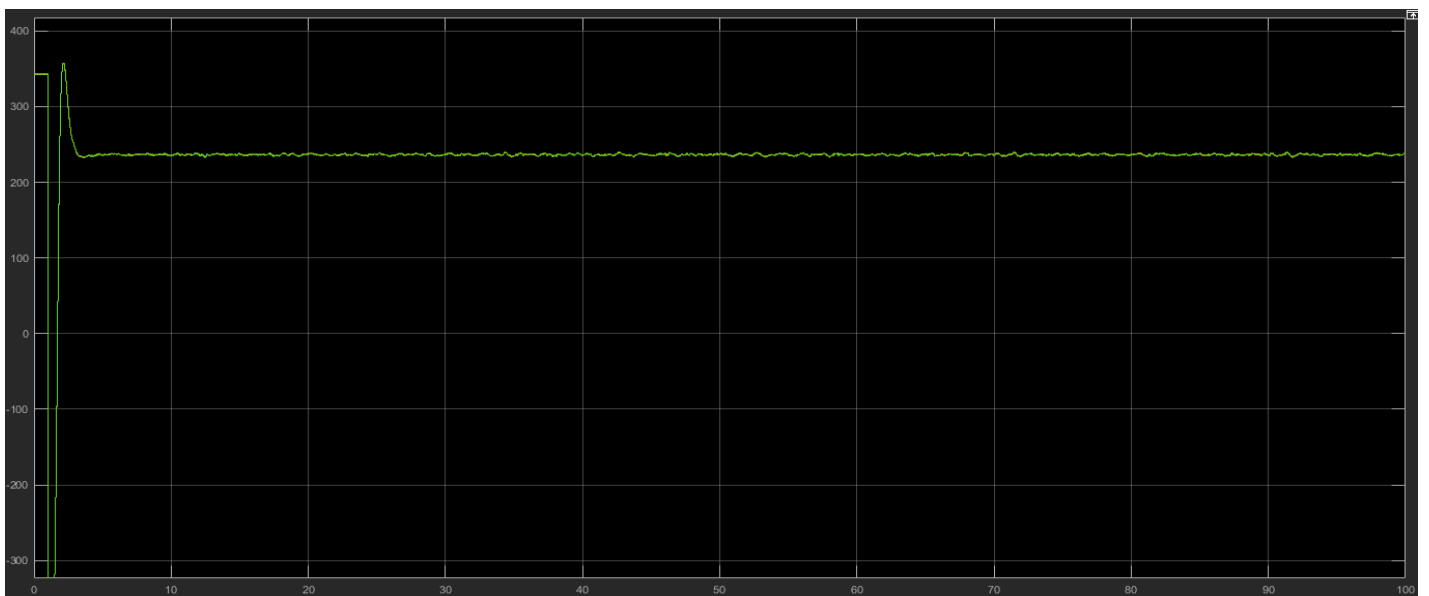


Fig. 60 – Saturazione dei motori

16 CONTROLLORE IN FREQUENZA NEL CASO LINEARE

Vediamo ora i risultati della simulazione dopo aver sostituito il PID originale per il controllo del pitch, con il controllore da me realizzato. Si ottengono i seguenti risultati:

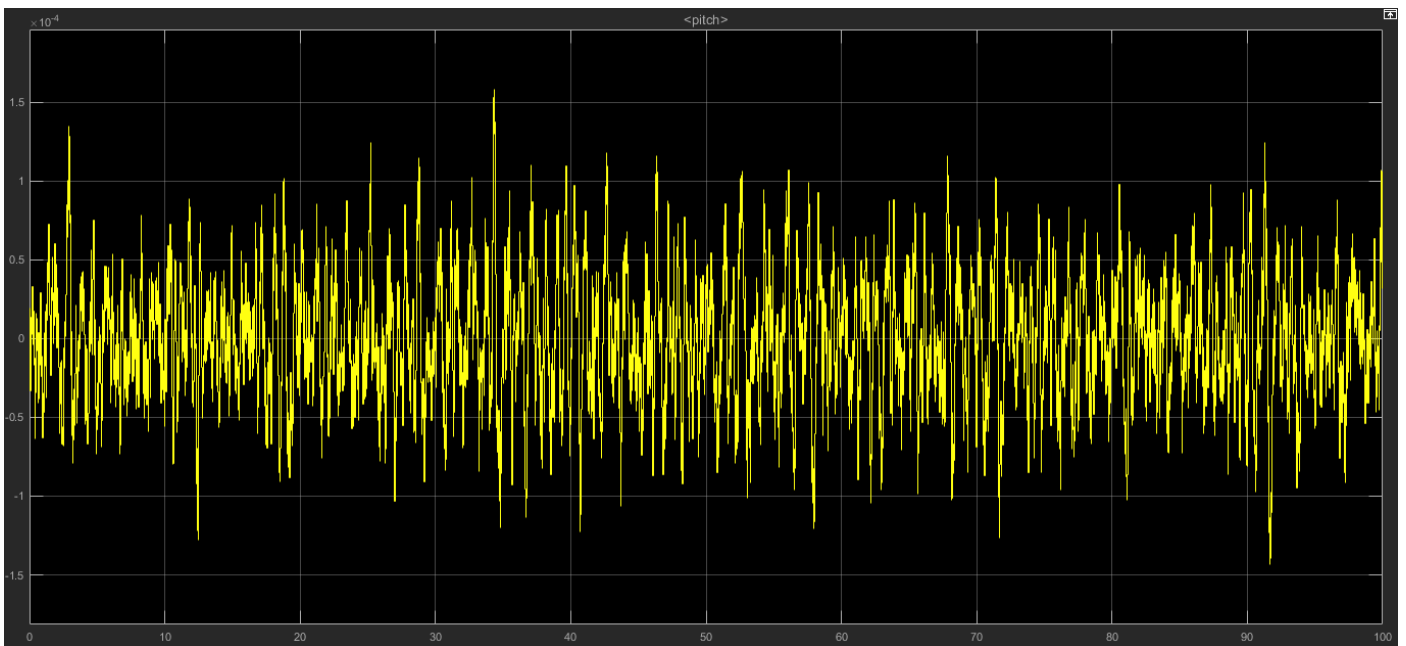


Fig. 61 – pitch

Vediamo che l'uscita del pitch ha valori dell'ordine di 10^{-4} quindi il controllore progettato con la sintesi in frequenza realizza l'obiettivo di controllo della variabile.

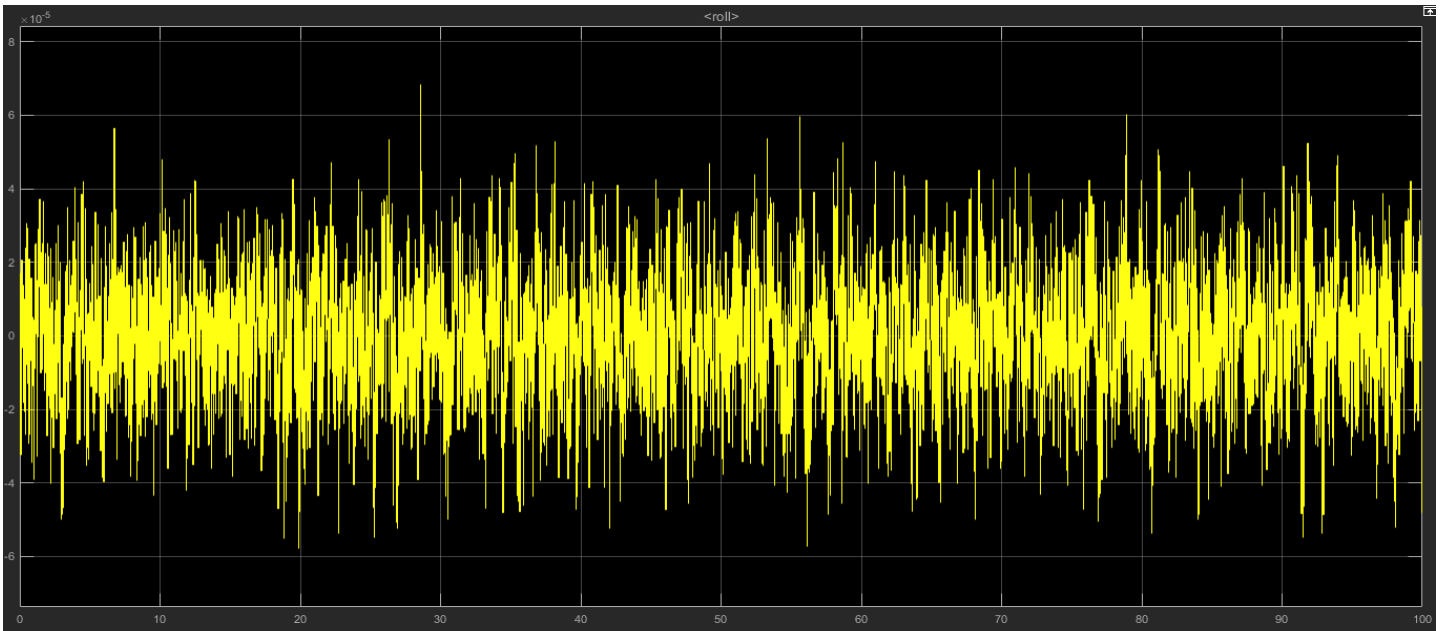


Fig. 62 – roll

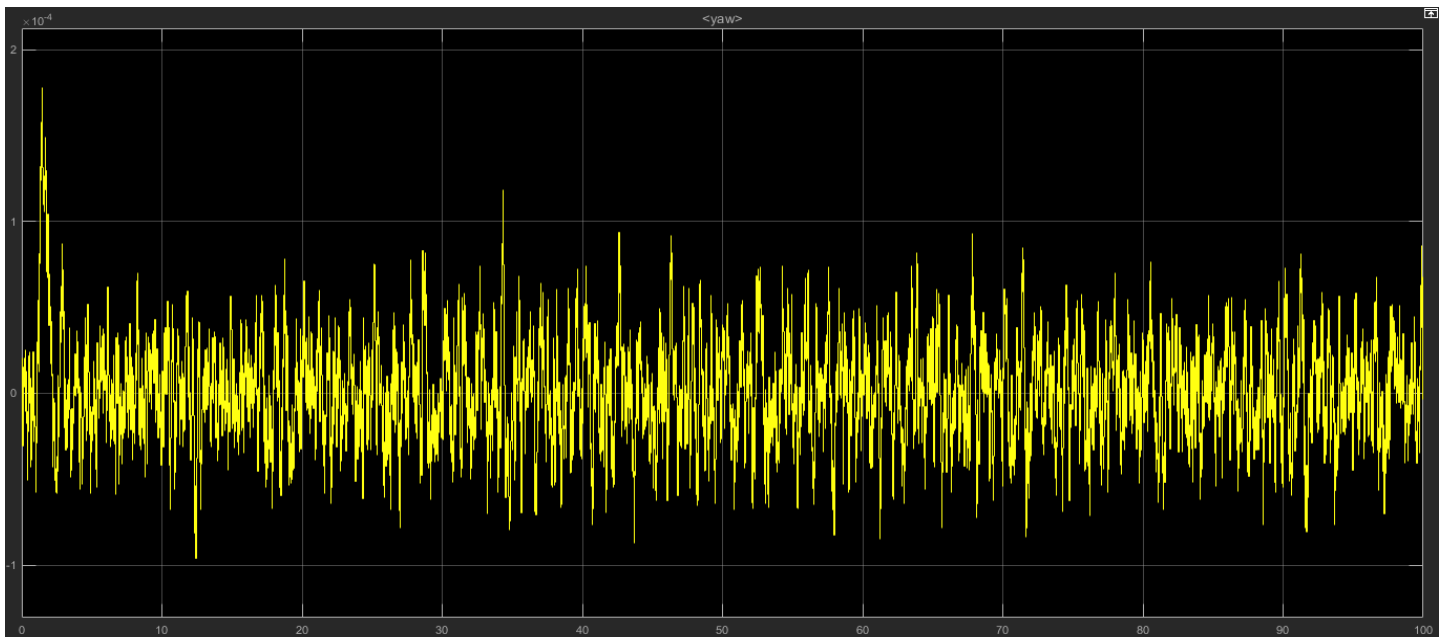


Fig. 63 - Yaw

Le oscillazioni di roll e yaw risultano molto basse, in quanto il loro andamento assume valori prossimi allo zero, essendo dell'ordine di 10^{-4} .

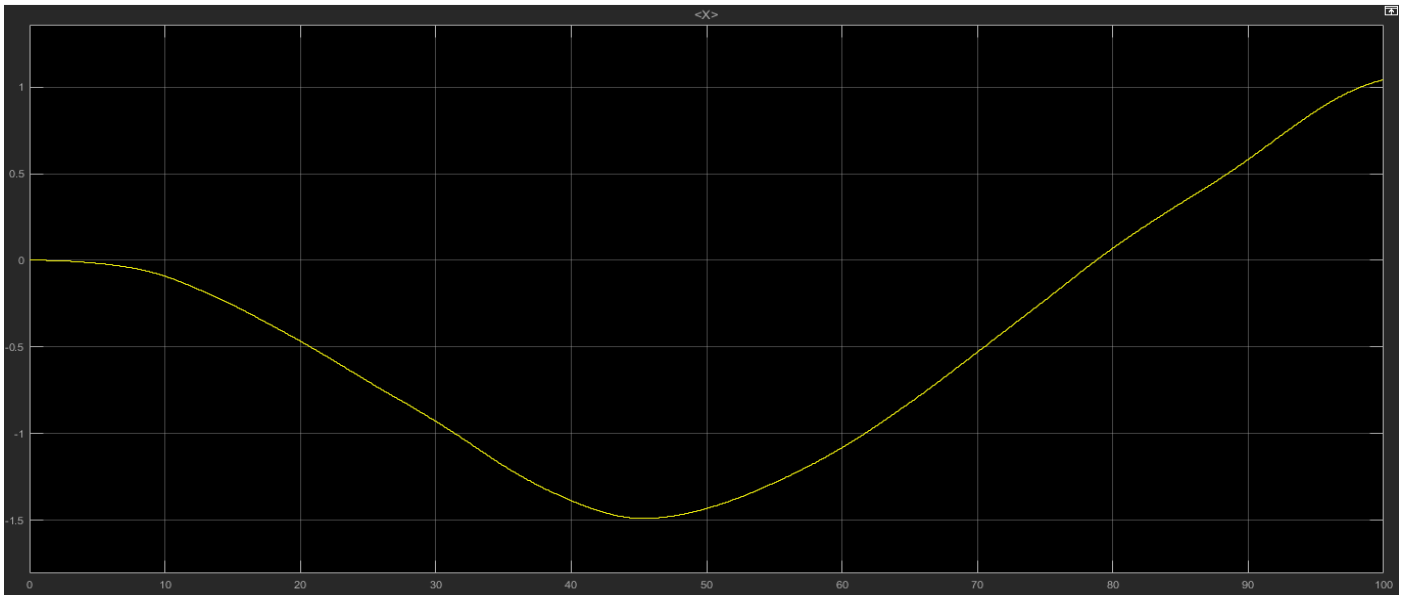


Fig. 64 - x

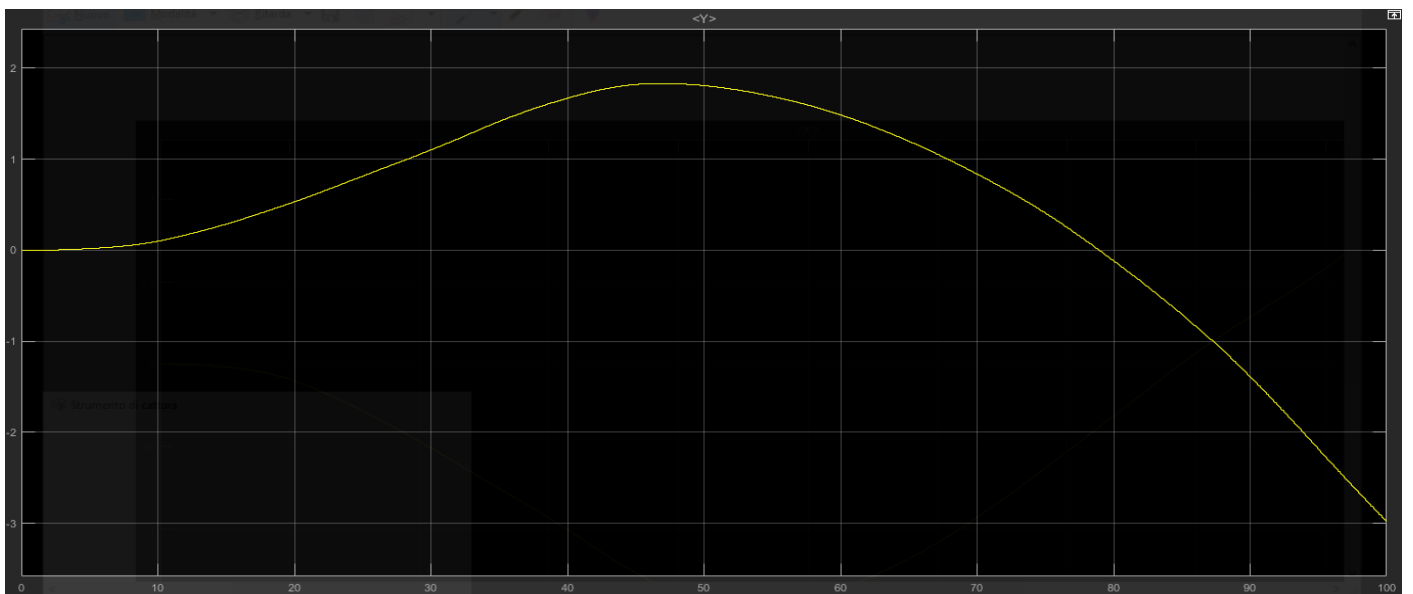


Fig. 65 - y

Come possiamo vedere nella figura 64 e nella figura 65, le uscite di x e y non seguono il valore di riferimento, che è impostato a 0. Questo è dovuto alla presenza di uno switch che impone o il controllo dell'orientamento o della posizione. Questo concetto verrà spiegato con maggiori dettagli nelle conclusioni.

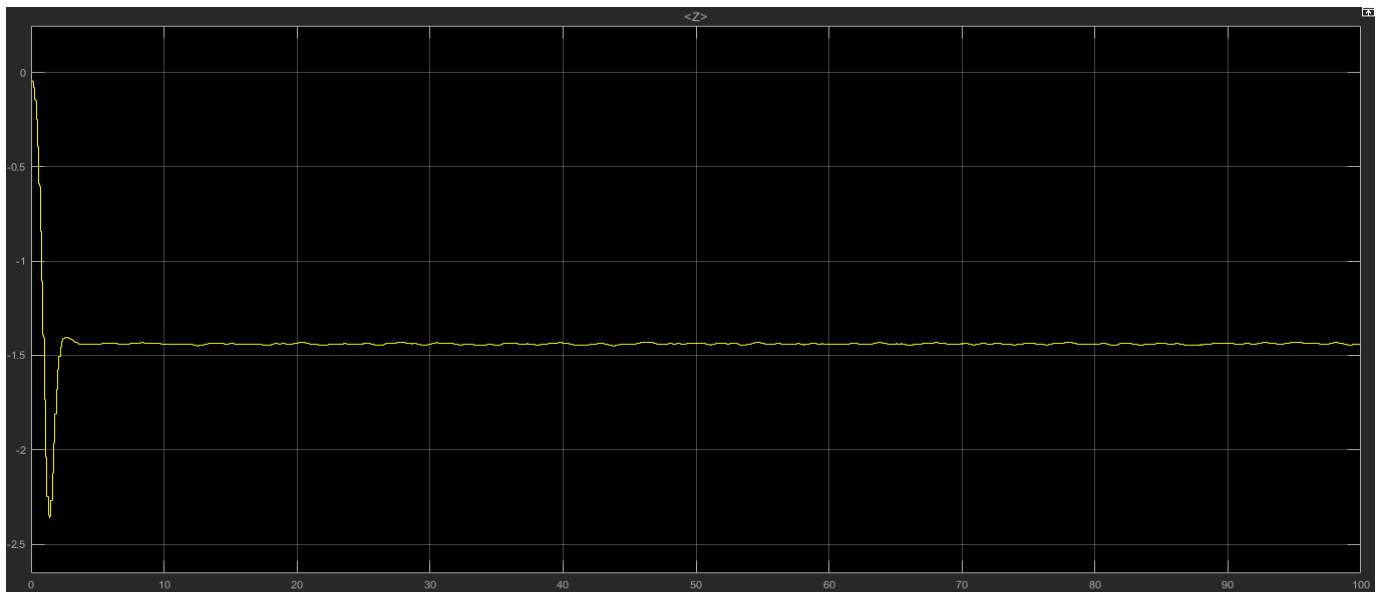


Fig. 66 – z

Tramite la figura 66 possiamo notare che viene mantenuta la quota costante imposta dal riferimento, che è pari a 1.5 metri di altezza dal suolo.

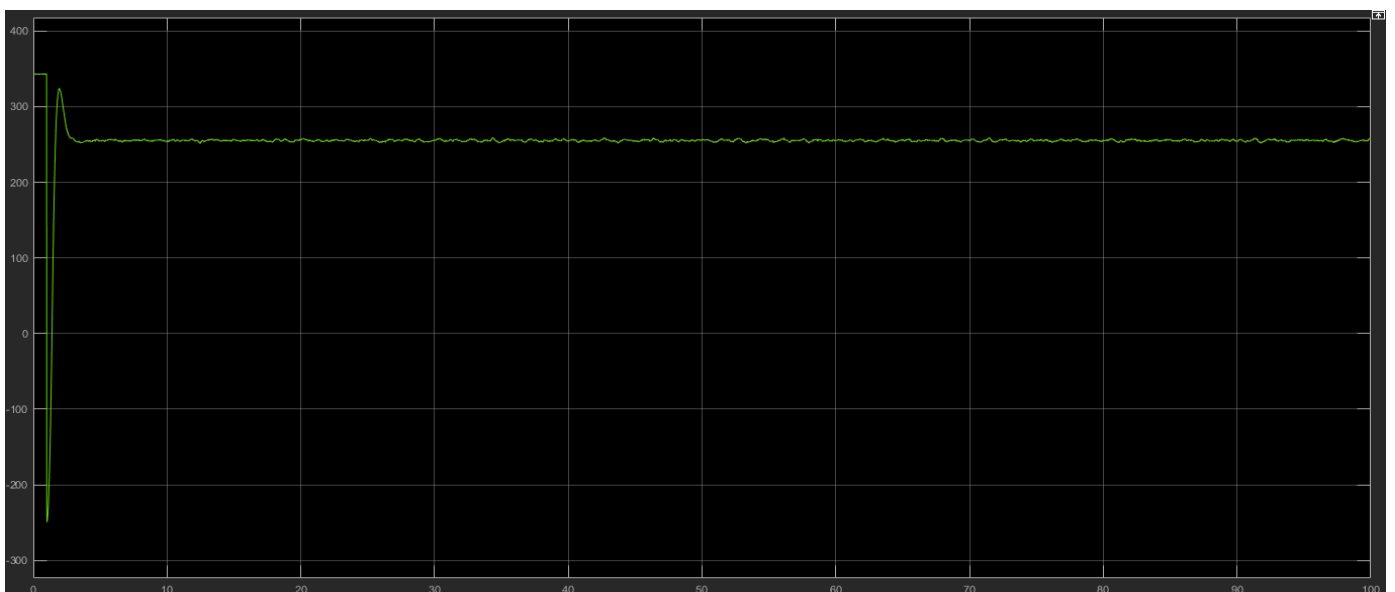


Fig. 67 – Saturazione dei motori

In figura 67 è rappresentata la saturazione dei motori. Essi raggiungono un valore molto alto, quello di saturazione, solo nei primi istanti quando devono avere una forza tale da riuscire ad innalzare il mini-drone, dopo di che, una volta raggiunta l'altezza desiderata, i motori si stabilizzano tutti sullo stesso valore, ottenendo una condizione di hovering.

17 CONTROLLORE IN FREQUENZA CASO NON LINEARE

Lanciando ora la simulazione nel caso non lineare, otteniamo i seguenti risultati:

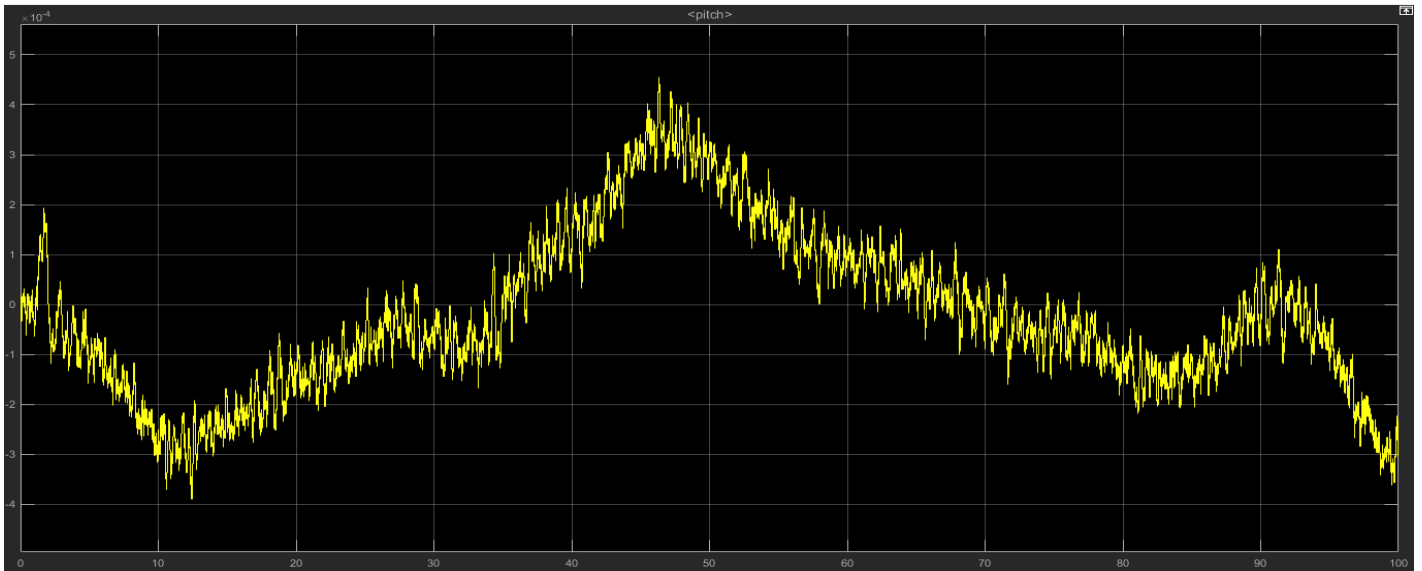


Fig. 68 – pitch

Notiamo che le oscillazioni del pitch sono sempre molto piccole, essendo dell'ordine di 10^{-4} , quindi possiamo affermare che il controllore progettato fa il suo dovere.

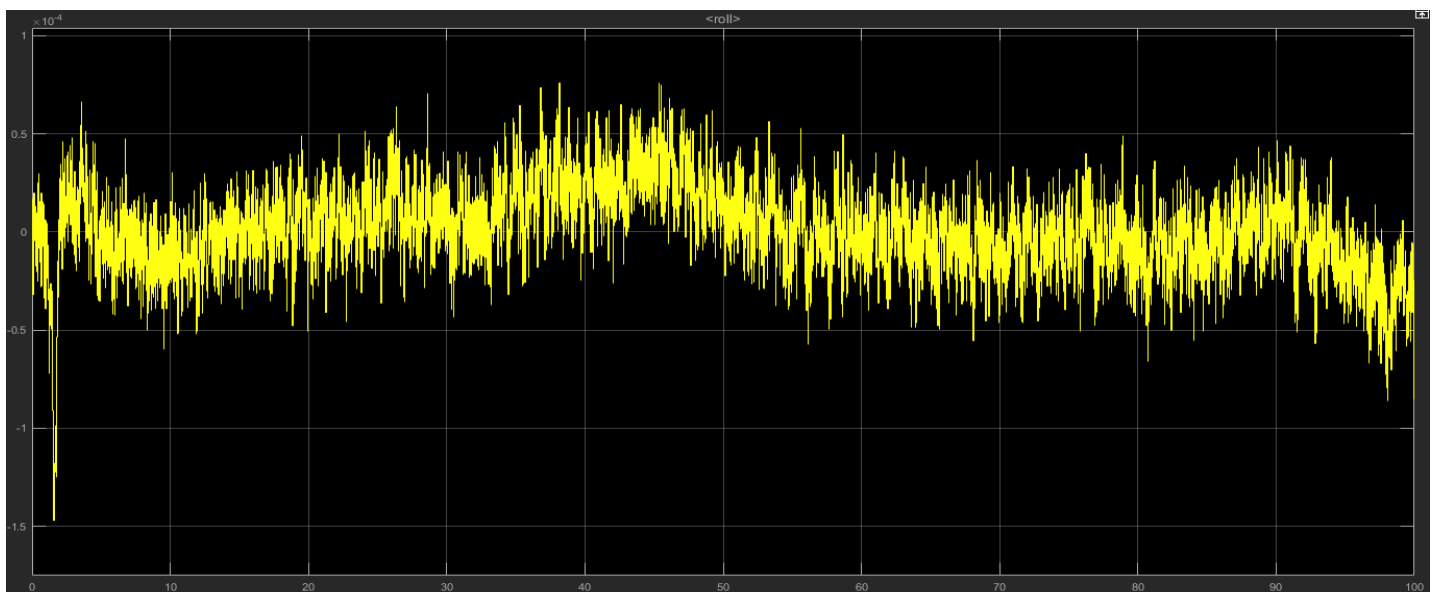


Fig. 69 – roll

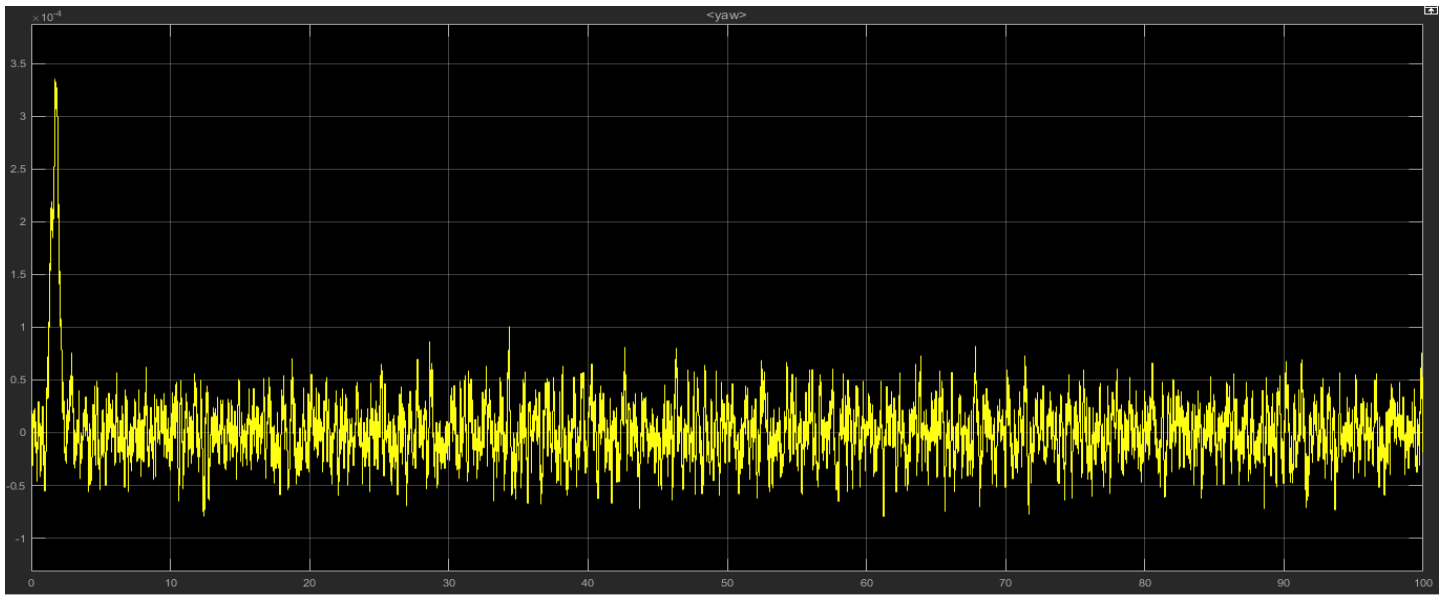


Fig. 70 – yaw

Anche le oscillazioni degli angoli di roll e yaw sono dell'ordine di 10^{-4} , quindi assumono valori molto vicini allo zero.

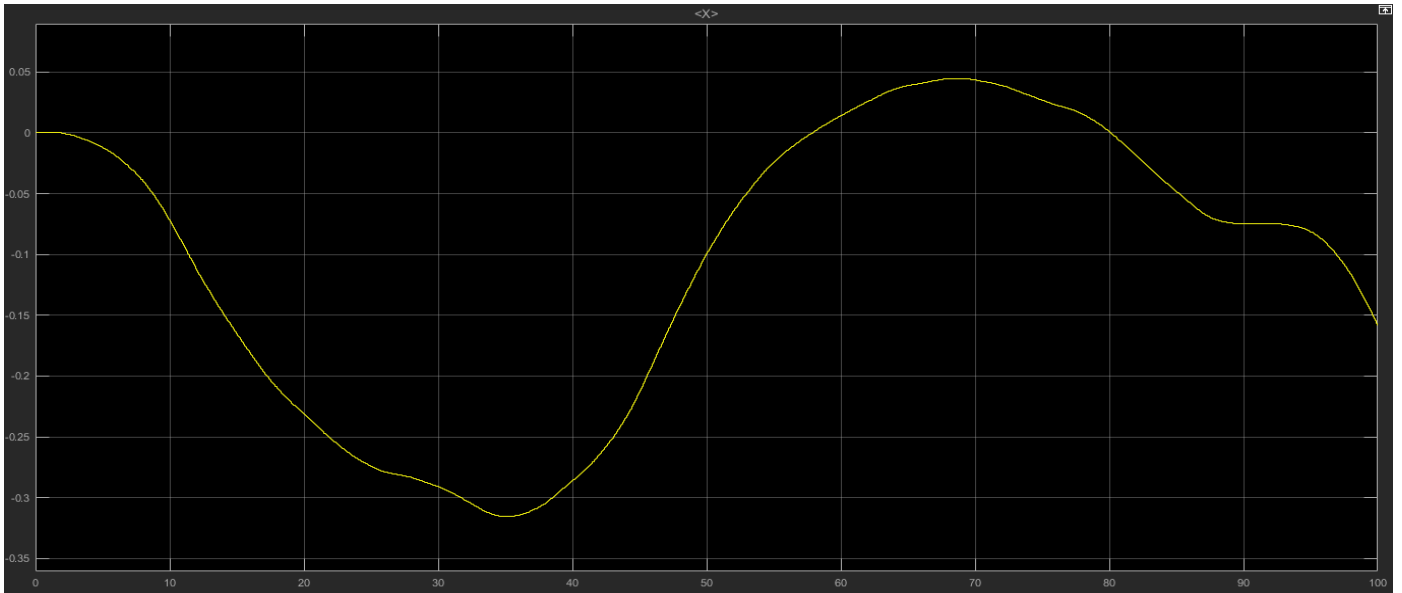


Fig. 71 - x

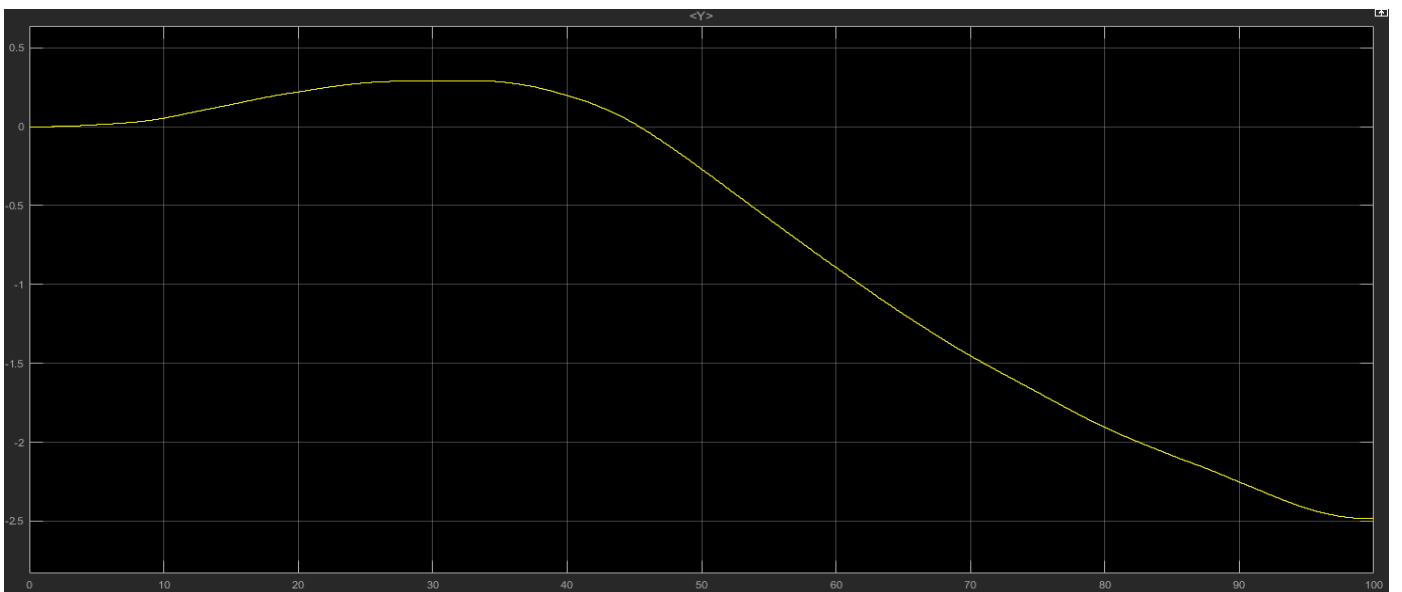


Fig. 72 - y

Per quanto riguarda l'andamento delle variabili di x e y , la spiegazione è la stessa del caso non lineare, e che verrà dettagliata nelle conclusioni.

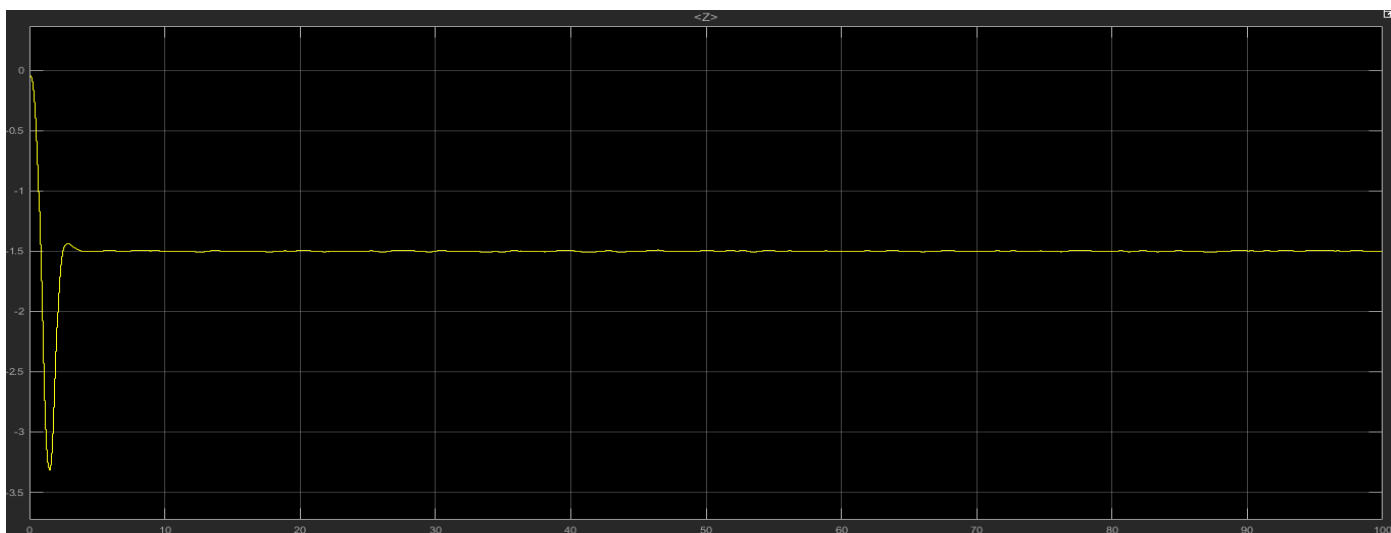


Fig. 73 - z

Anche qui possiamo notare che la quota pari a 1.5 metri, imposta dal riferimento, viene mantenuta.

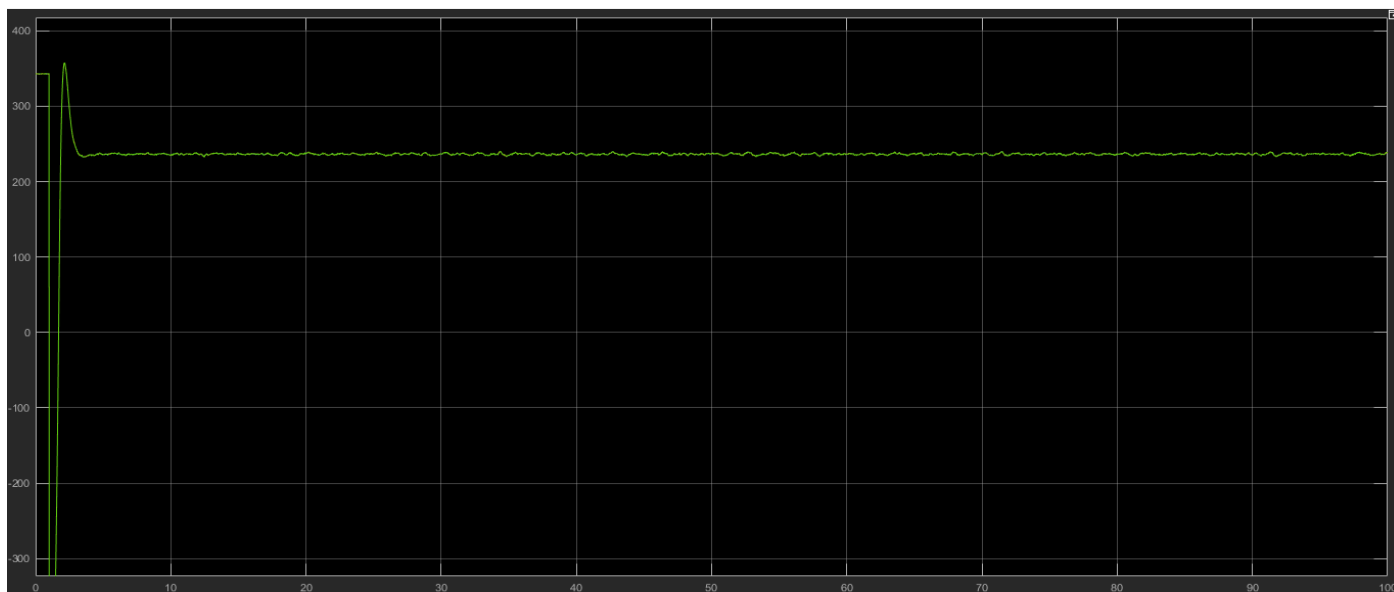


Fig. 74 – saturazione motori

In figura 74 è rappresentata la saturazione dei motori. Essi raggiungono un valore molto alto, quello di saturazione, solo nei primi istanti quando devono avere una forza tale da riuscire ad innalzare il mini-drone, dopo di che, una volta raggiunta l'altezza desiderata, i motori si stabilizzano tutti sullo stesso valore, ottenendo una condizione di hovering.

18 CONCLUSIONI

Come notiamo dai grafici dell'uscita controllata, il controllore in frequenza per l'angolo di pitch, è stato ben progettato in quanto l'orientamento viene mantenuto e siamo passati da oscillazioni dell'ordine di 10^{-3} ad oscillazioni dell'ordine 10^{-4} .

Per spiegare l'andamento delle variabili x e y , riprendiamo la figura 23 considerando la parte con lo switch:

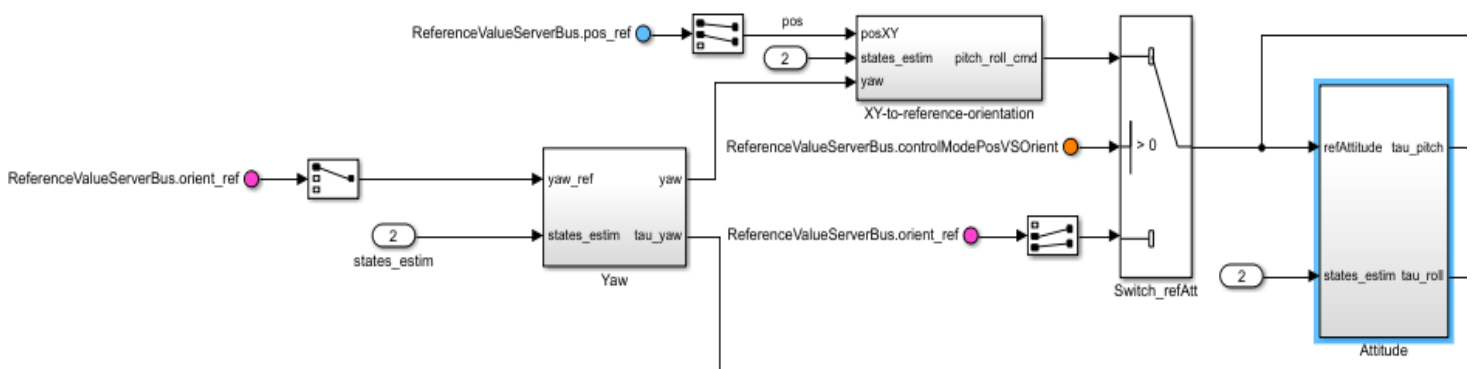


Fig. 23 – Flight Controller

Lo `switch_refAtt` è uno switch del Simulink, così funzionante: se l'ingresso 2 soddisfa il criterio di selezione, viene eseguita la condizione dell'ingresso 1; in caso contrario, viene eseguita quella dell'ingresso 3. Gli ingressi sono numerati dall'alto verso il basso. La prima e la terza porta di ingresso sono porte dati, mentre la seconda porta di ingresso è quella di controllo. I criteri per la porta di controllo 2 sono: $u_2 \geq \text{soglia}$, $u_2 > \text{soglia}$ o $u_2 \sim 0$. Nel nostro caso il valore di soglia è impostato a 0, e la condizione è che $u_2 > \text{soglia}$.

Ora abbiamo bisogno di riprendere un'altra figura, già spiegata nell'elaborato, per poter capire da dove provengono i valori in entrata alle porte dello switch e come viene valutato il criterio di selezione.

Riprendiamo quindi la figura 43, in cui però è stata modificata una componente: il blocchetto in alto a sinistra, che riceve in ingresso il segnale di pitch, è stato cambiato da "==" a "~=".

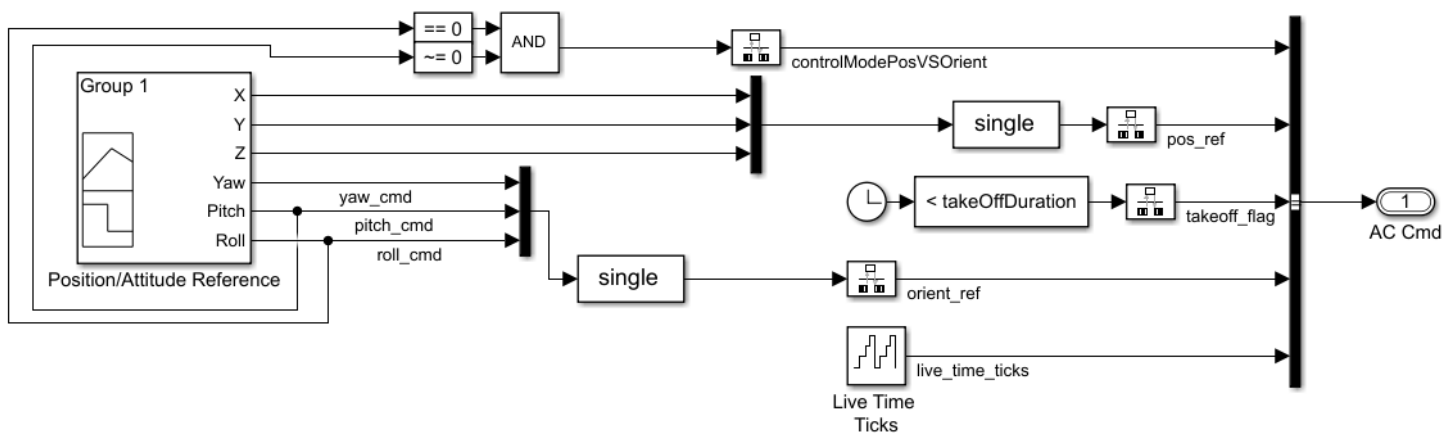


Fig. 43 – Riferimenti iniziali

L'uscita 1 "AC Cmd" entra nel blocco "flighControlSystem" (FCS) del modello a blocchi iniziale mostrato in figura 9. Successivamente il segnale va in ingresso sia ad "estimator" che a "controller". Consideriamo ora solo l'entrata in "controller", trovandoci quindi nella figura 23, riportata sopra.

La condizione dello switch, che vediamo indicata con il pallino arancione, si basa su ciò che esce dalla figura 43. Vediamo ora come, ma prima spieghiamo perché il blocchetto a sinistra è stato cambiato da "==" a "~=". Il segnale di riferimento del pitch e del roll è settato a 0, quindi essendo vera la condizione, da "AND" uscirebbe *true*, il cui valore è 1. Esso va confrontato con la soglia dello switch, la cui condizione è: se u2 (che nel caso appena considerato è 1) è > della soglia (che è settata a 0), allora esegui la condizione 1. Essendo verificata la condizione, viene presa in considerazione la porta di ingresso 1: è indicata con il pallino blu e prevede che venga fatto un controllo sulla posizione, come viene indicato anche dal nome "ReferenceValueServerBus.pos_ref". Il controllore da me progettato, prevede che venga fatto un controllo sull'orientamento, cioè un controllo sull'angolo di pitch e non sulla posizione. Quindi, lasciando i riferimenti di roll e pitch settati a 0, e cambiando il blocchetto da "==" a "~=", la condizione in uscita da "AND" è *false*, restituendo il valore 0. Esso viene confrontato con la condizione dello switch: se u2 (che in questo caso è 0) è > della soglia (che è settata a 0), allora esegui la condizione 1. Siccome il criterio di selezione non è

soddisfatto, viene eseguita la condizione della porta di ingresso 3, indicata in figura con il pallino viola con il nome "ReferenceValueServerBus.orient_ref". Esso va ad eseguire un controllo sull'orientamento, quindi va a controllare gli angoli.

Questa è la causa dello scostamento delle variabili di x e y dai valori di riferimento settati a 0; il controllore in frequenza è stato progettato per un controllo sull'orientamento, ma non sulla posizione. Infatti, come notiamo dai grafici rilevati, l'uscita da me controllata, il pitch, è stabile e segue il valore di riferimento.

Riferimenti bibliografici

- [1] A. Isidori, Sistemi di controllo
- [2] Samir Bouabdallah, Design and control of quadrotors with application to autonomous flying
- [3] Francesco Sabatino, Quadrotor control: modeling, nonlinear control design, and simulation
- [4] Fum, Wei Zhong, Implementation of Simulink controller design on Iris+ quadrotor
- [5] G. P. Carratelli, M. Del Duca, Controllo di un quadcopter