



**UNIVERSITÀ POLITECNICA DELLE MARCHE**  
FACOLTÀ DI INGEGNERIA

---

Corso di Laurea triennale in Ingegneria Informatica e dell'Automazione

# **Studio e sviluppo di algoritmi per l'inseguimento di traiettorie per droni**

*Study and development of a line follower algorithm for drones*

Relatore:

**Prof. Gianluca Ippoliti**

Correlatore:

**Prof. Giuseppe Orlando**

Candidato:

**Alexia Spreccà**

Matricola 1087720

---

Anno accademico 2020-2021



## **Ringraziamenti**

Ringrazio la mia famiglia per avermi accompagnato e sostenuto.  
Un grazie speciale va a Fabrizio, grande collega di questi 3 anni.



## Sommario

Il seguente elaborato ha lo scopo di illustrare le leggi matematiche e fisiche che regolano il volo di un quadrirotore, mostrarne il modello matematico alla base e una successiva linearizzazione che ci servirà per il progetto di nuovi controllori. Dopo aver analizzato il modello a blocchi Simulink per descrivere la dinamica del quadrirotore, vengono considerati due gradi di libertà, il pitch e il roll, e progettati dei nuovi controllori sia con la sintesi in  $\omega$  che con il luogo delle radici in modo tale da permettere al drone di mantenersi in una posizione di hovering. Si sono infine valutate le prestazioni dei controllori in ParrotMinidroneCompetition.



# Indice

<b>1</b>	<b>Introduzione</b>	<b>1</b>
1.1	Quadricotteri . . . . .	3
<b>2</b>	<b>Modello matematico</b>	<b>9</b>
2.1	Sistemi di riferimento . . . . .	9
2.2	Angoli di Eulero . . . . .	10
2.3	Assunzioni basilari prima della modellazione . . . . .	14
2.4	Dinamica del quadricottero . . . . .	15
2.5	Modello Matematico totale . . . . .	19
<b>3</b>	<b>Linearizzazione</b>	<b>20</b>
3.1	Linearizzazione di una funzione reale . . . . .	20
3.2	Linearizzazione di un sistema dinamico . . . . .	20
<b>4</b>	<b>Modello Simulink</b>	<b>23</b>
4.1	Command . . . . .	24
4.2	Sensor . . . . .	26
4.3	Environment . . . . .	27
4.4	Visualization . . . . .	29
4.5	Airframe . . . . .	30
4.5.1	Modello non lineare . . . . .	30
4.6	Flight Control System . . . . .	37
<b>5</b>	<b>Calcolo della funzione di trasferimento di Pitch e Roll</b>	<b>40</b>
5.1	Funzione di trasferimento del Pitch . . . . .	41
5.2	Funzione di trasferimento per il Roll . . . . .	47
<b>6</b>	<b>Sintesi di controllori</b>	<b>48</b>
6.1	Generalità sui PID . . . . .	48
6.2	Flight Controller . . . . .	49
6.3	PID di pitch e roll . . . . .	52
6.4	Sintesi in $\omega$ . . . . .	55
6.5	Sintesi con il luogo delle radici . . . . .	59
6.6	Controllori Pitch . . . . .	61
6.6.1	Sintesi in $\omega$ . . . . .	61

6.6.2	Luogo delle radici	66
6.7	Prestazioni Pitch	69
6.8	Controllori Roll	71
6.8.1	sintesi in $\omega$	71
6.8.2	luogo delle radici	75
6.9	Prestazioni roll	78
<b>7</b>	<b>Conclusioni</b>	<b>80</b>



# Elenco delle figure

1.1	Categorie di droni . . . . .	2
1.2	Configurazione del drone a più e a croce . . . . .	3
1.3	Rotazioni . . . . .	4
1.4	Manovra di salita . . . . .	5
1.5	Hovering del drone . . . . .	5
1.6	Meccanismi di volo per quadricotteri in configurazione a più . . . . .	6
1.7	Meccanismi di volo per quadricotteri in configurazione a croce . . . . .	7
1.8	Parrot Mambo . . . . .	8
2.1	Inertial Frame e Body Fixed Frame . . . . .	10
2.2	Angoli di Eulero . . . . .	11
4.1	Modello Simulink asbQuadcopter . . . . .	23
4.2	Command . . . . .	24
4.3	Signal Editor . . . . .	25
4.4	Position/Attitude Reference . . . . .	26
4.5	Sensor . . . . .	26
4.6	Sensor System . . . . .	27
4.7	Environment . . . . .	27
4.8	Environment(Costant) . . . . .	28
4.9	Environment(Variable) . . . . .	28
4.10	Visualization . . . . .	29
4.11	Simulink 3D . . . . .	29
4.12	Airframe . . . . .	30
4.13	Nonlinear Airframe . . . . .	31
4.14	AC model . . . . .	32
4.15	Gravity Force Calculation . . . . .	32
4.16	Drag Calculation . . . . .	33
4.17	Forza aerodinamica . . . . .	33
4.18	Motor Forces and Torques . . . . .	34
4.19	MotorsToW . . . . .	34
4.20	For Each Subsystem . . . . .	34
4.21	Calcolo dell'Advance Ratio . . . . .	35
4.22	Calcolo dell'angolo di orientamento . . . . .	36
4.23	Equazioni 4.9 4.10 4.11 . . . . .	36
4.24	FCS . . . . .	37

4.25	Flight Control System	38
4.26	Estimator	39
4.27	Logging	39
5.1	Modello lineare	40
5.2	Struttura dati: linsys	41
5.3	StateName	42
5.4	Matrice A	43
5.5	InputName	43
5.6	Matrice B	44
5.7	OutputName	44
5.8	Matrice C	45
5.9	Matrice D	45
6.1	Schema di controllo in controreazione	48
6.2	Flight Controller	50
6.3	Yaw Controller	50
6.4	X-Y Controller	51
6.5	Attitude Controller	51
6.6	Gravity Feedforward/Equilibrium Thrust	52
6.7	Pitch: modello lineare con riferimento costante	52
6.8	Pitch: modello lineare con riferimento variabile	53
6.9	Pitch: modello non lineare con riferimento costante	53
6.10	Pitch: modello non lineare con riferimento variabile	53
6.11	Roll: modello lineare con riferimento costante	54
6.12	Roll: modello lineare con riferimento variabile	54
6.13	Roll: modello non lineare con riferimento costante	54
6.14	Roll: modello non lineare con riferimento variabile	55
6.15	Funzione anticipatrice del modulo	58
6.16	Funzione anticipatrice della fase	58
6.17	Asintoti al variare di n-m	60
6.18	Diagrammi di Bode di $\hat{F}(s)$	62
6.19	Diagrammi di F(s)	63
6.20	Pitch sintesi in $\omega$	64
6.21	Pitch: modello lineare, riferimento costante	64
6.22	Pitch: modello lineare, riferimento variabile	65
6.23	Pitch: modello non lineare, riferimento costante	65
6.24	Pitch: modello non lineare, riferimento variabile	65
6.25	Funzione di trasferimento del processo	66
6.26	Pitch: sisotool(P(s))	67
6.27	Pitch: sisotool(F(s))	67
6.28	Pitch: modello lineare riferimento costante	68
6.29	Pitch: modello lineare riferimento variabile	68
6.30	Pitch: modello non lineare riferimento costante	68
6.31	Pitch: modello non lineare riferimento variabile	69

6.32	Prestazioni pitch con controllore PID . . . . .	70
6.33	Prestazioni pitch con controllore sviluppato con il luogo delle radici . . . . .	70
6.34	Prestazioni pitch con controllore sviluppato con la sintesi in $\omega$ . . . . .	71
6.35	Diagrammi di Bode di $\hat{F}(s)$ . . . . .	72
6.36	Roll: sisotool(F(s)) . . . . .	73
6.37	Roll: modello lineare, riferimento costante . . . . .	74
6.38	Roll: modello lineare, riferimento variabile . . . . .	74
6.39	Roll: modello non lineare, riferimento costante . . . . .	74
6.40	Roll: modello non lineare, riferimento variabile . . . . .	75
6.41	Roll: sisotool(P(s)) . . . . .	75
6.42	Roll: sisotool(F(s)) . . . . .	76
6.43	Roll: modello lineare riferimento costante . . . . .	76
6.44	Roll: modello lineare riferimento variabile . . . . .	77
6.45	Roll: modello non lineare riferimento costante . . . . .	77
6.46	Roll: modello non lineare riferimento variabile . . . . .	77
6.47	Prestazioni roll con controllore PID . . . . .	78
6.48	Prestazioni roll con il luogo delle radici . . . . .	78
6.49	Prestazioni roll con la sintesi in $\omega$ . . . . .	79

# Capitolo 1

## Introduzione

I droni sono aeromobili a pilotaggio remoto cioè apparecchi volanti che vengono pilotati da un computer di bordo o da un pilota che li guida da remoto con un radiocomando. I primi droni erano stati principalmente utilizzati per missioni militari come per la ricognizione del territorio. Con l'andare avanti del tempo, furono impiegati anche per scopi commerciali, scientifici, ricreativi, per l'agricoltura e per altri scopi. In questi anni vengono utilizzati anche per consegne, per la registrazione e la fotografia aerea. I droni possono avere una struttura con eliche, che li rende simili a un elicottero, una struttura planare cioè con ali fisse, oppure una struttura ibrida che permette loro di viaggiare oltre che in aria anche sulla terra ferma e in acqua.

Esistono molte caratteristiche che possono variare nella struttura dei droni come il numero delle eliche, la potenza dei motori di quest'ultime, la durata delle batterie, la qualità delle telecamere, il peso e molte altre varianti.

I droni a rotore singolo si possono assimilare a dei mini elicotteri a pilotaggio semplificato. I droni multirotoresono i più comuni tra i droni. Sono classificati in base al numero di eliche possedute, a partire dal tricottero (tre eliche o rotori), quadricottero (quattro rotori), esacottero (sei rotori) e ottocottero (otto rotori). I multirotoresono facili da controllare, sono ideali per la consegna di piccoli carichi perchè in grado di decollare e atterrare verticalmente, quasi ovunque. Il volo è più stabile rispetto al drone a rotore singolo o ad ala fissa e il loro limite principale è il consumo della batteria. Il minidrone preso in considerazione in questo elaborato è il modello Mambo dell'azienda francese: Parrot SA, la quale mette a disposizione un pacchetto di supporto Simulink per i minidroni Parrot (Simulink Support Package for Parrot Minidrones). Tale pacchetto permette di progettare algoritmi di controllo del volo per i minidroni Parrot. Gli algoritmi possono accedere ai sensori di bordo, come i sensori a ultrasuoni, accelerometro, giroscopio e sensori per la pressione dell'aria, nonché alla fotocamera rivolta verso il basso. Tale pacchetto include quindi un modello Simulink che ci permette di simulare il comportamento del velivolo in varie condizioni di volo e ambientali. Partendo da tale modello si è poi sostituito il controllore di default del pitch e del roll con nuovi controllori creati con la sintesi in  $\omega$  e con il luogo delle radici e se ne sono valutate le prestazioni.



(a) Tricottero



(b) Quadricottero



(c) Esacottero



(d) Ottacottero

Figura 1.1: Categorie di droni

## 1.1 Quadricotteri

Il drone di nostro interesse, il Parrot Mambo, fa parte della categoria dei quadricotteri, i quali sono i più comuni sul mercato.

Essi hanno una base centrale da cui si estendono tanti bracci quanti sono i motori adottati, perciò quattro bracci. L'aspetto fondamentale su cui si insiste è lo stesso che in qualunque tipo di struttura: rigidità e peso. Se il telaio fosse troppo pesante il payload disponibile sarebbe notevolmente inferiore o richiederebbe una motorizzazione superiore per lo spostamento e questo influirebbe negativamente su costi e consumi.

Il funzionamento corretto di tutti gli organi è coadiuvato dal controllore del drone. Posizionato al centro del telaio, il controllore non è altro che una scheda programmabile, molto spesso di tipo Arduino. Su di essa è montato un processore per l'analisi dei dati dei vari input provenienti da piattaforme inerziali (IMU), GPS, altimetro e magnetometro.

Si possono adottare due tipi di configurazione di volo in base all'allineamento o sfasamento degli assi  $x$  e  $y$  rispetto ai bracci del quadricottero.

- Configurazione a più
- Configurazione a croce

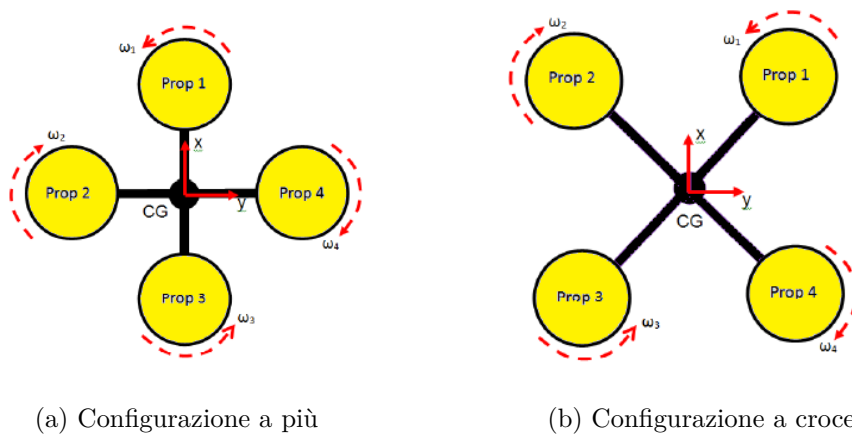


Figura 1.2: Configurazione del drone a più e a croce

Il controllo dei movimenti avviene tramite la variazione della velocità relativa di ogni rotore, in modo tale da cambiare la spinta e la coppia prodotta da ciascuno di essi. Ogni rotore produce una spinta ed una coppia motrice nel suo centro di rotazione, così come una certa resistenza fluidodinamica opposta alla direzione di volo del velivolo. La posizione del veicolo può essere facilmente descritta da tre coordinate nello spazio, ma per ottenere il controllo sul quadricottero dobbiamo anche tener conto del suo orientamento. Il quadricottero è tuttavia un sistema sotto-attuato : infatti possiede sei gradi di libertà (6 DoF), dei quali tre sono dovuti alla rotazione intorno a tre assi mentre i restanti tre al movimento in tre direzioni, ma è controllato solo attraverso quattro attuatori. Infatti, mentre il movimento lungo l'asse z e la variazione dell'angolo di Yaw sono indipendenti dagli altri gradi di libertà, un movimento frontale richiede una variazione dell'angolo di pitch così come a un movimento laterale corrisponde una variazione dell'angolo di roll.

### Gradi di libertà del drone

- spostamento lungo la direzione frontale del mezzo (x)
- spostamento lungo la direzione laterale del veicolo (y)
- spostamento lungo la direzione verticale (z)
- rotazione intorno ad un asse longitudinale del corpo (roll o rollio)
- rotazione intorno ad un asse trasversale del corpo (pitch o beccheggio)
- rotazione intorno ad un asse verticale passante per il baricentro del corpo (yaw o imbardata)

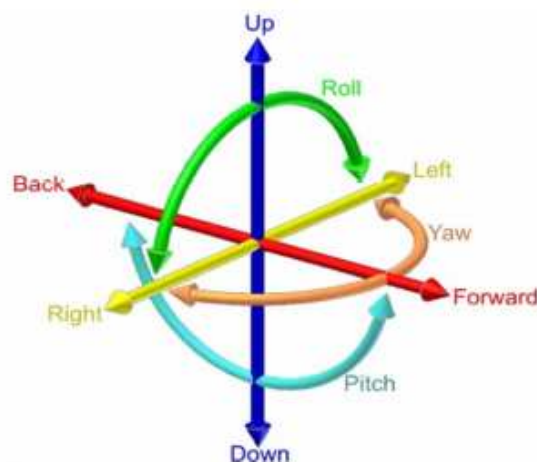


Figura 1.3: Rotazioni

Rotori opposti ruotano nella stessa direzione. In questo modo si rimuove la necessità di un rotore di coda (necessario invece nella struttura standard dell'elicottero).

Se tutti i motori girano alla stessa velocità si ottiene accelerazione angolare nulla intorno agli assi di rotazione.

Ne consegue che modificando opportunamente i giri si avrà uno scompenso di forze e/o momenti in grado di far effettuare al drone le manovre desiderate.

Sia nella configurazione a più che a croce la manovra di salita viene eseguita aumentando la spinta di tutti i rotori allo stesso modo.

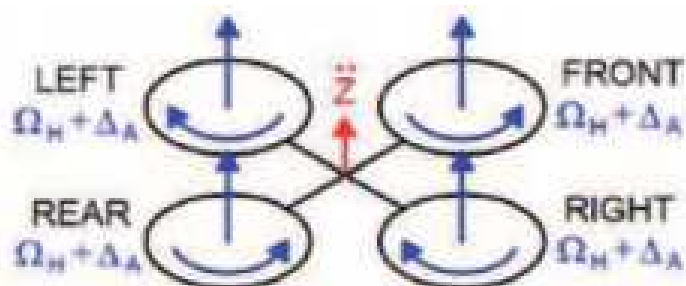


Figura 1.4: Manovra di salita

La particolare tipologia di volo che si verifica quando il drone staziona in aria a velocità nulla e quota costante, detta *hovering*, è possibile considerarla un caso particolare della manovra di salita in cui ciascun motore ha tiro pari ad un quarto del peso del drone, raggiungendo il bilancio delle forze tra spinta e forza peso.

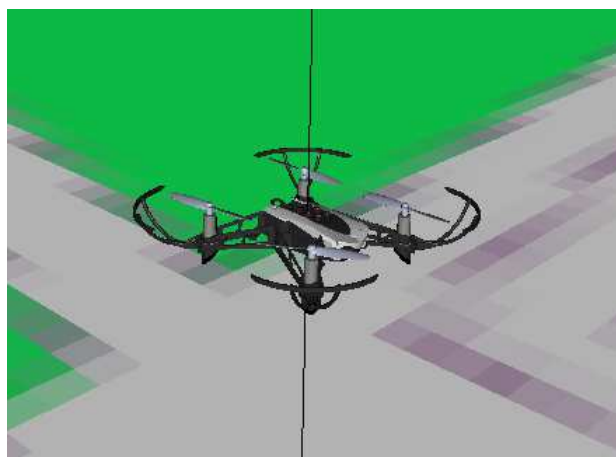


Figura 1.5: Hovering del drone



Per ottenere spostamenti lungo le 3 direzioni sopra citate o per effettuare manovre di beccheggio,rollio o imbardata bisogna modificare in maniera opportuna la potenza da dare ai 4 motori.

### Configurazione a più

Facendo riferimento alla figura 1.2 si può notare come gli assi x e y siano allineati con i bracci del drone.

In questo caso per ottenere una manovra di rollio occorrerà aumentare la velocità rotazionale del secondo rotore e diminuire quella del quarto o viceversa, a seconda che si voglia rispettivamente un'inclinazione verso sinistra o verso destra.

Per ottenere invece una manovra di beccheggio dovrà essere aumentata la velocità rotazionale del primo rotore e diminuita quella del terzo o viceversa.

Infine per ottenere una manovra di imbardata servirà aumentare la velocità rotazionale dei rotori 1 e 3 e diminuire quella dei rotori 2 e 4. In questo modo noteremo una rotazione del velivolo intorno al proprio asse verticale in senso antiorario.

Per avere una rotazione in senso orario basterà aumentare la velocità rotazionale dei motori 2 e 4 e diminuire quella dei motori 1 e 3.

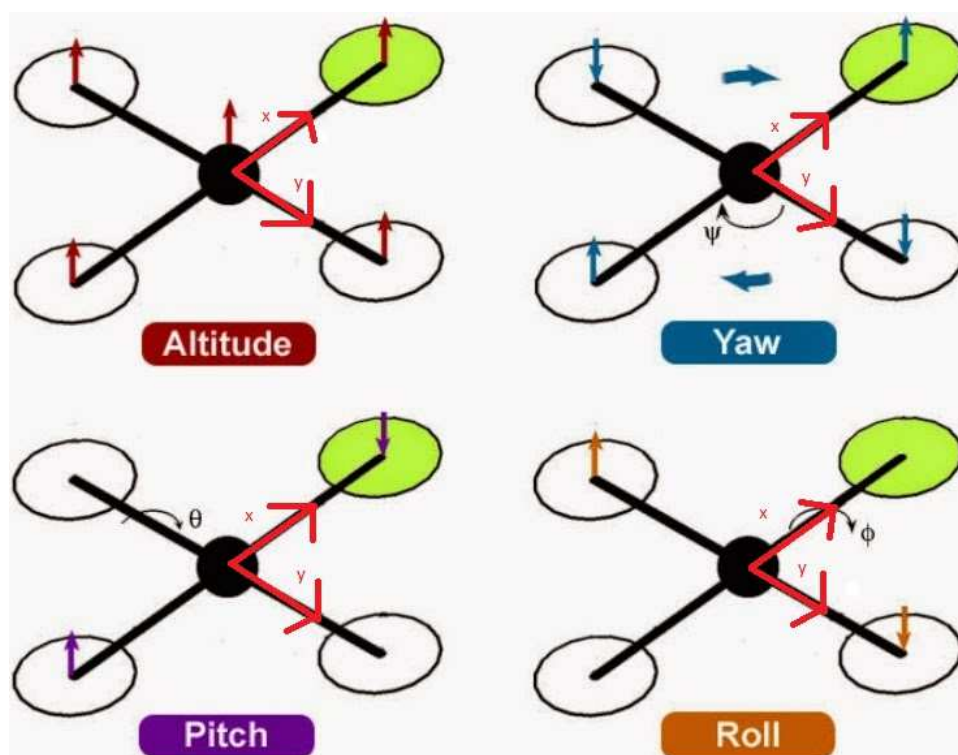


Figura 1.6: Meccanismi di volo per quadricotteri in configurazione a più

## Configurazione a croce

Facendo sempre riferimento alla figura 1.2 si può osservare come gli assi x e y siano sfalsati di  $45^\circ$  rispetto ai bracci del drone.

In questo caso per ottenere manovre di beccheggio o di rollio occorrerà modificare la velocità rotazionale di tutti e 4 i rotori e non solo di due come in precedenza.

Per ottenere una manovra di beccheggio bisognerà aumentare la velocità rotazionale dei motori 1 e 2 e diminuire quella dei motori 3 e 4 o viceversa.

Per una manovra di rollio invece occorrerà aumentare la velocità rotazionale dei motori 1 e 4 e diminuire quella dei motori 2 e 3 o viceversa.

Infine per quanto riguarda l'imbardata, come per la configurazione a più, si dovrà aumentare la velocità rotazionale di due rotori opposti e diminuire quella dell'altra coppia a seconda del verso in cui si vuole che avvenga la rotazione.

Il controllo di beccheggio e rollio è migliore di circa il 30% per la configurazione a croce poiché vengono utilizzati tutti e quattro i rotori anziché solo due.

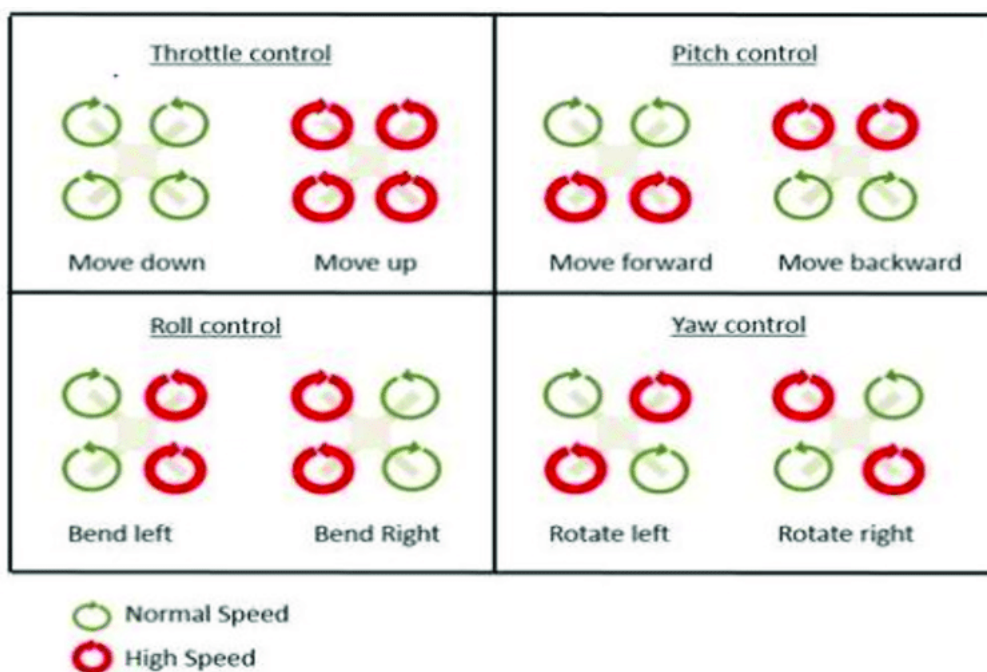


Figura 1.7: Meccanismi di volo per quadrotori in configurazione a croce

## Caratteristiche di Parrot Mambo

Parrot Mambo fa parte della categoria dei minidroni.

Alcune delle caratteristiche principali dei minidroni sono:

- Manovrabilità
- Resistenza
- Leggerezza

Rispetto ai droni normali, però, i minidroni hanno alcuni aspetti negativi:

- Scarsa qualità delle riprese
- Bassa durata della batteria
- Poca distanza dal telecomando

### Scheda tecnica del minidrone Mambo

Tipologia	Drone
Serie	Parrot Mambo
Versione modello	RtF
Tempo di volo	9 min
Tempo di ricarica	30 min
Risoluzione	0.3 MPixel
Fotogrammi per secondo	60
Batteria di volo	1S LiPo / 550 mAh
Larghezza	180 mm
Trasferimento immagini	WLAN su smartphone
Categoria	Quadricottero
Lunghezza - Profondità	180 mm



Figura 1.8: Parrot Mambo

# Capitolo 2

## Modello matematico

In questo capitolo si andranno ad analizzare le leggi fisiche che regolano il moto di un quadricotore per arrivare a esplicitare un modello matematico in grado di modellare in maniera corretta la dinamica del sistema.

Tale modello sarà poi ritrovato all'interno di Simulink.

### 2.1 Sistemi di riferimento

Come prima cosa, andiamo a definire i sistemi di riferimento (fisso e solidale al corpo del drone) in modo tale da ricavare posizione e orientamento del velivolo in maniera più semplice possibile. Prendiamo in considerazione 2 diversi sistemi di riferimento:

- **Inertial Frame (Sistema di riferimento terrestre)**

La posizione del veicolo può essere descritta dalle coordinate  $x$ ,  $y$  e  $z$ , attraverso un sistema di riferimento solidale con l'osservatore con origine nel baricentro del drone.

Tale sistema fisso, viene scelto in riferimento al piano tangente passante per la superficie terrestre in ogni punto: è stato chiamato  $O_E$ .

Il pedice "E" sta per Earth, da cui i tre assi riprendono le direzioni, con gli assi  $x$  ed  $y$  appartenenti al piano tangente che puntano rispettivamente verso il Nord e l'Est, mentre l'asse  $z$  ha direzione perpendicolare al piano e diretto verso il basso.

- **Body Fixed Frame**

E' conveniente esprimere le rotazioni in un sistema di coordinate con origine nel centro di massa del drone e assi allineati con i bracci del velivolo. Tale sistema è chiamato "Body Fixed Frame".

L'asse  $x$  punta verso la parte frontale del corpo, l'asse  $y$  punta verso la destra e l'asse  $z$  punta verso il basso.

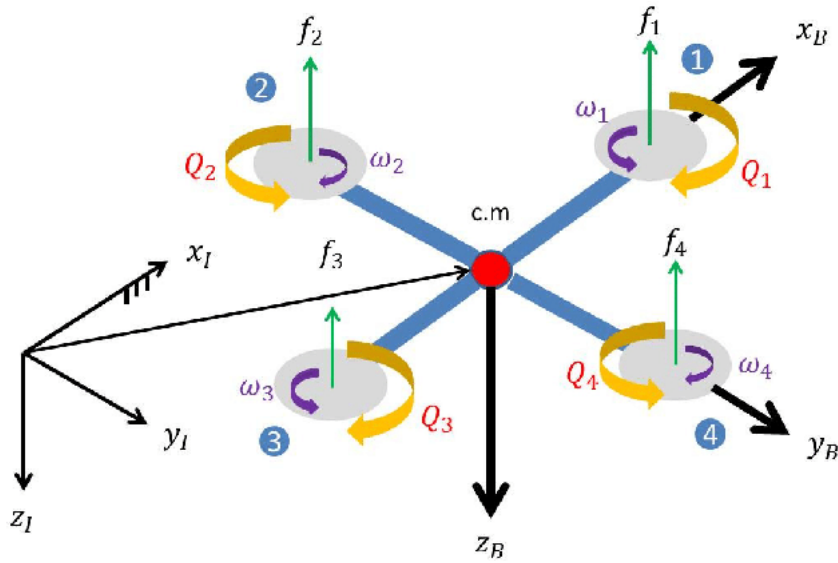


Figura 2.1: Inertial Frame e Body Fixed Frame

## 2.2 Angoli di Eulero

Gli angoli di Eulero sono stati introdotti per descrivere l'orientamento di un corpo rigido nello spazio. Un corpo rigido è un sistema di punti materiali caratterizzati dal fatto che le loro mutue distanze si mantengono costanti nel tempo, indipendentemente dalle eventuali sollecitazioni a cui è soggetto il sistema.

Gli angoli di Eulero descrivono la posizione di un sistema di riferimento XYZ solidale con un corpo rigido attraverso una serie di rotazioni a partire da un sistema di riferimento fisso xyz. I due sistemi di riferimento coincidono nell'origine.

Se i piani xy e XY sono distinti, si intersecano in una retta (passante per l'origine) detta linea dei nodi N. Se i piani coincidono, si definisce la linea dei nodi come l'asse X. Gli angoli di Eulero sono:

- $\alpha$ : angolo tra l'asse x e la linea dei nodi. Detto angolo di precessione, è definito in  $[0, 2\pi]$  oppure in  $[-\pi, \pi]$ ;
- $\beta$ : angolo tra gli assi z e Z. Detto angolo di nutazione, è definito in  $[0, \pi]$  oppure in  $[-\frac{\pi}{2}, \frac{\pi}{2}]$ ;
- $\gamma$ : angolo tra la linea dei nodi e l'asse X. Detto angolo di rotazione propria, è definito in  $[0, 2\pi]$  oppure in  $[-\pi, \pi]$ ;

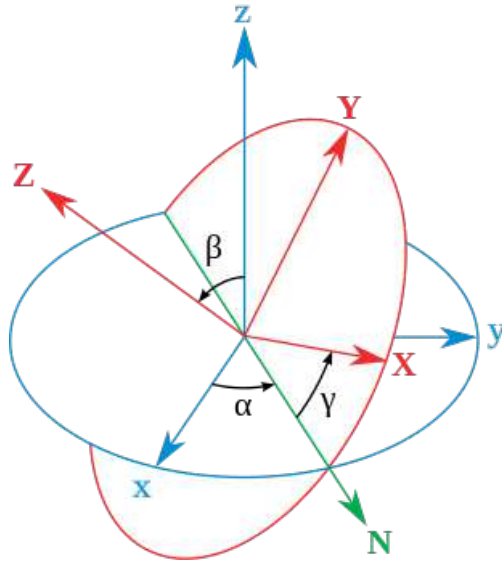


Figura 2.2: Angoli di Eulero

Il sistema fisso (xyz) è rappresentato in blu mentre il sistema ruotato (XYZ) è rappresentato in rosso. La linea dei nodi, indicata con N, è rappresentata in verde.

## Matrici di rotazione

Nel sistema di riferimento degli assi terrestri introduciamo il vettore lineare posizione ( $P_i$ ) e il vettore che rappresenta l'orientamento nello spazio del velivolo tramite gli angoli di Eulero ( $A_i$ ).

$$P_i = [x_i \ y_i \ z_i]^T$$

$$A_i = [\phi \ \theta \ \psi]^T$$

Nella terna di riferimento del sistema corpo del drone si introducono i vettori che richiamano la posizione del quadricottero nel body frame ( $P_b$ ), le velocità di traslazione ( $V_b$ ) e di rotazione ( $\omega_b$ ).

$$P_b = [x_b \ y_b \ z_b]^T$$

$$V_b = [u \ v \ w]^T$$

$$\omega_b = [p \ q \ r]^T$$

Per eseguire il passaggio dal sistema di riferimento mobile (Body Frame) a quello fisso (Inertial Frame) e viceversa intervengono tre matrici di rotazione, ciascuna corrispondente ad un angolo, e quindi ad una manovra di roll, pitch o yaw.

- Rotazione di un angolo  $\phi$  intorno all'asse x in senso antiorario.

$$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix} \quad (2.1)$$

- Rotazione di un angolo  $\theta$  intorno all'asse y in senso antiorario.

$$R_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \quad (2.2)$$

- Rotazione di un angolo  $\psi$  intorno all'asse z in senso antiorario.

$$R_z(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2.3)$$

Per poter effettuare il passaggio dal sistema di riferimento solidale al corpo del drone al sistema di riferimento terrestre, le 3 matrici devono essere moltiplicate tra loro. Il risultato è una matrice di rotazione completa (DCM : *Direct Cosine Matrix*).

$$R_b^i = R_z(\psi) \cdot R_y(\theta) \cdot R_x(\phi) \quad (2.4)$$

$$R_b^i = \begin{bmatrix} \cos(\psi) \cos(\theta) & \cos(\psi) \sin(\theta) \sin(\phi) - \sin(\psi) \cos(\phi) & \cos(\psi) \sin(\theta) \cos(\phi) + \sin(\psi) \sin(\phi) \\ \sin(\psi) \cos(\theta) & \sin(\psi) \sin(\theta) \sin(\phi) + \cos(\psi) \cos(\phi) & \sin(\psi) \sin(\theta) \cos(\phi) - \sin(\phi) \cos(\psi) \\ -\sin(\theta) & \cos(\theta) \sin(\phi) & \cos(\theta) \cos(\phi) \end{bmatrix} \quad (2.5)$$

### Passaggio di coordinate dal Body Fixed Frame all'Inertial Frame

$$\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} = R_b^i \begin{bmatrix} x_b \\ y_b \\ z_b \end{bmatrix} \quad (2.6)$$

Per poter effettuare il passaggio inverso occorrerà considerare la trasposta della matrice DCM dato che quest'ultima è una matrice ortogonale e quindi la trasposta coincide con l'inversa.

$$R_i^b = (R_b^i)^T$$

$$R_i^b = \begin{bmatrix} \cos(\psi) \cos(\theta) & \sin(\psi) \cos(\theta) & -\sin(\theta) \\ \cos(\psi) \sin(\theta) \sin(\phi) - \sin(\psi) \cos(\phi) & \sin(\psi) \sin(\theta) \sin(\phi) + \cos(\psi) \cos(\phi) & \cos(\theta) \sin(\phi) \\ \cos(\psi) \sin(\theta) \cos(\phi) + \sin(\psi) \sin(\phi) & \sin(\psi) \sin(\theta) \cos(\phi) - \sin(\phi) \cos(\psi) & \cos(\theta) \cos(\phi) \end{bmatrix} \quad (2.7)$$

### Passaggio di coordinate dall'Inertial Frame al Body Fixed Frame

$$\begin{bmatrix} x_b \\ y_b \\ z_b \end{bmatrix} = R_i^b \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} \quad (2.8)$$

### Velocità lineare dal Body Frame all'Inertial Frame

$$\dot{P}_i = R_i^b V_b \quad (2.9)$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} \cos(\psi) \cos(\theta) & \cos(\psi) \sin(\theta) \sin(\phi) - \sin(\psi) \cos(\phi) & \cos(\psi) \sin(\theta) \cos(\phi) + \sin(\psi) \sin(\phi) \\ \sin(\psi) \cos(\theta) & \sin(\psi) \sin(\theta) \sin(\phi) + \cos(\psi) \cos(\phi) & \sin(\psi) \sin(\theta) \cos(\phi) - \sin(\phi) \cos(\psi) \\ -\sin(\theta) & \cos(\theta) \sin(\phi) & \cos(\theta) \cos(\phi) \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \quad (2.10)$$

### Conversione tra le velocità angolari nel Body Frame e nell'Inertial Frame

Prima di passare alla trattazione della dinamica del velivolo andiamo a definire il vettore delle velocità angolari nei due sistemi di riferimento e le matrici di rotazione che consentono di passare dall'uno all'altro.

$$\omega_b = [p \ q \ r]^T$$

$$\omega_i = [\dot{\phi} \ \dot{\theta} \ \dot{\psi}]^T$$

### Velocità angolari dall'Inertial Frame al Body Frame

$$\omega_b = \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\sin(\theta) \\ 0 & \cos(\phi) & \cos(\theta) \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\theta) \cos(\phi) \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} \dot{\phi} - \dot{\psi} \sin(\theta) \\ \dot{\theta} \cos(\phi) + \dot{\psi} \sin(\phi) \cos(\theta) \\ -\dot{\theta} \sin(\phi) + \dot{\psi} \cos(\phi) \cos(\theta) \end{bmatrix} \quad (2.11)$$



## Velocità angolari dal Body Frame all'Inertial Frame

$$\omega_i = \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin(\phi) \tan(\theta) & \cos(\phi) \tan(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) \sec(\theta) & \cos(\phi) \sec(\theta) \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \quad (2.12)$$

$$= \begin{bmatrix} (p \cos(\theta) + q \sin(\phi) \sin(\theta) + r \cos(\phi) \sin(\theta)) / \cos(\theta) \\ q \cos(\phi) - r \sin(\phi) \\ (q \sin(\phi) + r \cos(\phi)) / \cos(\theta) \end{bmatrix} \quad (2.13)$$

La matrice dell'equazione 2.12, che chiameremo ROT, può essere ridotta a una matrice unitaria standard quando i quattro rotori sono stabilizzati in volo, l'angolo di rollio e l'angolo di beccheggio sono piccoli così come la loro velocità angolare.

$$\omega_i = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \omega_b \quad (2.14)$$

Nel set di equazioni sopra descritte il calcolo dell'angolo  $\psi$  è inscindibile dagli altri due. Nella pratica questo significa che manovre in rollio ( $\phi$ ) e/o beccheggio ( $\theta$ ) possono automaticamente causare variazioni angolari di  $\psi$  ottenendo anche una manovra di imbardata.

## 2.3 Assunzioni basilari prima della modellazione

Prima della creazione del modello dinamico non lineare del quadricottero consideriamo alcune ipotesi:

- Il velivolo è un corpo rigido. Ignorando la vibrazione elastica della lama e la deformazione, esso conserva la propria forma, oltre che il proprio volume, indipendentemente dallo stato dinamico in cui si trova.
- Non viene presa in considerazione la rotazione terrestre e il movimento di rivoluzione.
- Il quadricottero è dotato di una rigorosa struttura simmetrica, il centro di gravità può essere considerato coincidente con il centro del corpo.
- La spinta generata da un rotore è direttamente proporzionale al quadrato della velocità.
- Essendo tutti e 4 i motori uguali, è possibile prenderne in considerazione uno alla volta.

## 2.4 Dinamica del quadricottero

Per poter arrivare alle equazioni che definiscono la dinamica del quadricottero definiamo innanzitutto uno dei principi fondamentali della dinamica: *La seconda legge di Newton*. Tale principio afferma che se su un oggetto con una certa massa agisce una forza risultante, l'oggetto subisce un'accelerazione uguale alla forza risultante divisa per la massa.

$$\sum_i^N \vec{F}_i = m \vec{a}$$

Somma delle forze
massa del corpo
accelerazione

Inoltre, in meccanica classica, la rotazione di un corpo rigido è descritta dalle equazioni di Eulero, usando un sistema di riferimento rotante con gli assi fissati nel corpo e paralleli agli assi principali di inerzia del corpo.

La forma vettoriale generale è :

$$I\dot{\omega} + \omega \times (I\omega) = \tau \quad (2.15)$$

dove  $\tau$  è il momento meccanico applicato,  $I$  è la matrice di inerzia e  $\omega$  è la velocità angolare intorno agli assi principali.

Considerando che un quadricottero è un corpo rigido simmetrico e che gli assi di rotazione coincidono con gli assi principali, il momento d'inerzia può essere espresso come:

$$I = \begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix} \quad (2.16)$$

Applicando le due leggi appena descritte al modello del quadricottero si ottiene:

$$m\ddot{P}_i = \sum_{i=1}^N F_i \quad (2.17)$$

$$I\ddot{A}_i = \sum_{i=1}^N \tau_i \quad (2.18)$$

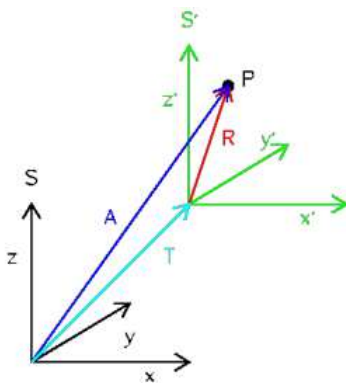
Riprendendo l'equazione 2.17 e riscrivendola utilizzando le derivate prime della traslazione si ottiene:

$$m \frac{\partial}{\partial t} (R_b^i \cdot V_b) = R_b^i \cdot F_b \quad (2.19)$$

Ricordando che la derivata del prodotto di due funzioni derivabili è uguale alla derivata della prima funzione per la seconda più la prima funzione per la derivata della seconda, avremo:

$$m \cdot (\dot{R}_b^i \cdot V_b + R_b^i \cdot \dot{V}_b) = R_b^i \cdot F_b \quad (2.20)$$

Se si prende come sistema di riferimento una terna mobile, velocità e accelerazione sono la somma di due componenti: una componente di trascinamento (velocità della terna rispetto al sistema assoluto) e una componente relativa (velocità del punto rispetto alla terna mobile).



Tornando alla 2.20 si può quindi esprimere il prodotto  $\dot{R}_b^i \cdot V_b$  come :  $R_b^i \cdot \omega_b \times V_b$

$$m \cdot R_b^i \cdot (\dot{V}_b + \omega_b \times V_b) = R_b^i \cdot F_b \quad (2.21)$$

Dividendo tale equazione per la matrice di rotazione  $R_b^i$  si avrà:

$$m \cdot (\dot{V}_b + \omega_b \times V_b) = F_b \quad (2.22)$$

Tale equazione vettoriale può essere espressa in 3 equazioni scalari:

$$F_b = \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = m \begin{bmatrix} \dot{u} + qw - rv \\ \dot{v} + ru - pw \\ \dot{w} + pv - qu \end{bmatrix} \quad (2.23)$$

Si vanno ora a considerare le forze esterne agenti sul velivolo, fin qui trascurate. Possiamo scomporle in forza peso, dovuta all'attrazione gravitazionale della Terra, e in "forza dei motori", definita spinta.

Per la forza peso si utilizza la formulazione classica: il prodotto della massa per la gravità. La direzione di questo vettore, ovviamente, sarà sempre verticale e diretta verso il basso indipendentemente dall'orientamento del corpo. Per la spinta dei motori si considera una modellazione semplificata dipendente dal quadrato delle velocità di rotazione di ciascun rotore:

$$T = b(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) \quad (2.24)$$

dove b rappresenta il coefficiente di spinta dei motori.

Essendo il tutto calcolato in assi corpo, la spinta sarà sempre diretta lungo l'asse z, mentre il vettore peso avrà componenti diverse dipendenti dall'assetto del quadrirotore. Il vettore delle forze esterne può essere quindi scritto come:

$$\begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = F_e = DCM \cdot \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix} \quad (2.25)$$

Si ottengono così le seguenti equazioni:

$$\begin{cases} \dot{u} = rv - qw - g \sin(\theta) \\ \dot{v} = pw - ru + g \cos(\theta) \sin(\phi) \\ \dot{w} = qu - pv + g \cos(\phi) \cos(\theta) - \frac{b}{m}(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) \end{cases} \quad (2.26)$$

Partendo invece dalla 2.18 ed effettuando gli stessi passaggi sopra riportati per la seconda legge di Newton, si avrà:

$$\begin{aligned} I \frac{\partial}{\partial t} (ROT \cdot \omega_b) &= ROT \cdot \tau_b \\ I(\dot{ROT} \cdot \omega_b + ROT \cdot \dot{\omega}_b) &= ROT \cdot \tau_b \\ I \cdot ROT \cdot \dot{\omega}_b + ROT \cdot \omega_b \times I \cdot \omega_b &= ROT \cdot \tau_b \\ I \cdot (\dot{\omega}_b + \omega_b \times I \cdot \omega_b) &= \tau_b \end{aligned} \quad (2.27)$$

Tale equazione vettoriale si può scomporre nelle 3 equazioni scalari:

$$\tau_b = \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \begin{bmatrix} \dot{p}I_x + qr(I_z - I_y) \\ \dot{q}I_y + pr(I_x - I_z) \\ \dot{r}I_z + pq(I_y - I_x) \end{bmatrix} \quad (2.28)$$

Si prendono ora in considerazione i momenti esterni agenti sul quadricottero. Il quadricottero infatti per poter effettuare delle manovre di rollio, beccheggio o imbardata deve essere sottoposto a determinate coppie di rotazione che si ottengono moltiplicando le forze per il braccio agente, ovvero la distanza tra il punto di applicazione della forza e il baricentro del corpo.

$$\begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \tau_e = \begin{bmatrix} l(T_4 + T_1 - T_2 - T_3) \\ l(-T_4 + T_1 + T_2 - T_3) \\ d(\omega_1^2 + \omega_3^2 - \omega_2^2 - \omega_4^2) \end{bmatrix} = \begin{bmatrix} lb(\omega_4^2 + \omega_1^2 - \omega_2^2 - \omega_3^2) \\ lb(-\omega_4^2 + \omega_1^2 + \omega_2^2 - \omega_3^2) \\ d(\omega_1^2 + \omega_3^2 - \omega_2^2 - \omega_4^2) \end{bmatrix} \quad (2.29)$$

dove  $l$  è il braccio,  $d$  è il *Drag Factor* ovvero il fattore di trascinamento delle eliche,  $b$  è la costante di coppia della motorizzazione scelta e  $T_i$  è la spinta generata dall' $i$ -esimo rotore. Si può notare la dipendenza quadratica del valore delle coppie al variare del numero dei giri.

Per semplicità di notazione si può scrivere:

$$\begin{aligned} U_1 &= lb(\omega_4^2 + \omega_1^2 - \omega_2^2 - \omega_3^2) \\ U_2 &= lb(-\omega_4^2 + \omega_1^2 + \omega_2^2 - \omega_3^2) \\ U_3 &= d(\omega_1^2 + \omega_3^2 - \omega_2^2 - \omega_4^2) \end{aligned}$$

Possiamo quindi sostituire la 2.28 nella 2.29 ottenendo:

$$\begin{bmatrix} \dot{p}I_x + qr(I_z - I_y) \\ \dot{q}I_y + pr(I_x - I_z) \\ \dot{r}I_z + pq(I_y - I_x) \end{bmatrix} = \begin{bmatrix} U_1 \\ U_2 \\ U_3 \end{bmatrix} \quad (2.30)$$

Isolando le accelerazioni angolari nel Body Fixed Frame si avrà:

$$\begin{cases} \dot{p} = \frac{U_1}{I_x} - qr \frac{I_z - I_y}{I_x} \\ \dot{q} = \frac{U_2}{I_y} - pr \frac{I_x - I_z}{I_y} \\ \dot{r} = \frac{U_3}{I_z} \end{cases} \quad (2.31)$$

## 2.5 Modello Matematico totale

Mettendo insieme tutte le equazioni sopra trovate si ottiene:

$$\left\{ \begin{array}{l}
 \dot{x} = (\cos \psi \cos \theta)u + (\cos \psi \sin \theta \sin \phi - \sin \psi \cos \phi)v + (\cos \psi \sin \theta \cos \phi + \sin \psi \sin \phi)w \\
 \dot{y} = (\sin \psi \cos \theta)u + (\sin \psi \sin \theta \sin \phi + \cos \psi \cos \phi)v + (\sin \psi \sin \theta \cos \phi - \sin \phi \cos \psi)w \\
 \dot{z} = -\sin(\theta)u + (\cos(\theta) \sin(\phi))v + (\cos(\theta) \cos(\phi))w \\
 \dot{\phi} = p + \sin(\phi) \tan(\theta)q + r \cos(\phi) \tan(\theta) \\
 \dot{\theta} = q \cos(\phi) - r \sin(\phi) \\
 \dot{\psi} = q \sin(\phi) \sec(\theta) + r \cos(\phi) \sec(\theta) \\
 \dot{u} = rv - qw - g \sin(\theta) \\
 \dot{v} = pw - ru + g \cos(\theta) \sin(\phi) \\
 \dot{w} = qu - pv + g \cos(\phi) \cos(\theta) - \frac{b}{m}(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) \\
 \dot{p} = \frac{U_1}{I_x} - qr \frac{I_z - I_y}{I_x} \\
 \dot{q} = \frac{U_2}{I_y} - pr \frac{I_x - I_z}{I_y} \\
 \dot{r} = U_3 / I_z
 \end{array} \right. \quad (2.32)$$

Le prime tre equazioni sono prese dalla 2.10 e rappresentano le velocità lineari nell'Inertial Frame.

La quarta, quinta e sesta equazione non sono altro che le velocità angolari nel sistema di riferimento terrestre (è possibile ritrovarle nella 2.12).

La settima, ottava e nona equazione stanno ad indicare le accelerazioni lineari nel sistema corpo del drone (equazione: 2.26)

Infine le ultime 3 equazioni vengono riprese dalla 2.31 ed indicano le accelerazioni angolari nel Body Fixed Frame.

Il sistema presenta delle componenti non lineari. Sarà quindi necessaria una linearizzazione in un punto di lavoro per poter applicare delle tecniche di controllo lineare.

# Capitolo 3

## Linearizzazione

Gli elementi di non linearità presenti nel modello matematico totale del quadricottero rendono più difficile lo studio di tale modello e la progettazione di controllori lineari. E' quindi necessario introdurre una linearizzazione di tale modello che approssimi il comportamento del modello non lineare.

### 3.1 Linearizzazione di una funzione reale

Una funzione  $f(x): \mathbb{R} \rightarrow \mathbb{R}$  può essere sviluppata in serie di Taylor in un intorno di ampiezza  $\partial x = x - x_0$  di un qualsiasi valore  $x_0$  della variabile reale  $x$  come:

$$f(x) = f(x_0 + \partial x) =$$

$$= f(x_0) + \left(\frac{d}{dx}f(x)\right)_{|x=x_0}\partial x + \left(\frac{1}{2!}\frac{d^2}{dx^2}f(x)\right)_{|x=x_0}\partial x^2 + .. \quad (3.1)$$

La funzione  $f(x)$  può essere approssimata in tale intorno mediante il troncamento  $h(x)$  dello sviluppo in serie di Taylor arrestato al termine di primo grado:

$$f(x) = f(x_0 + \partial x) \simeq f(x_0) + \left(\frac{d}{dx}f(x)\right)_{|x=x_0}\partial x = h(x) \quad (3.2)$$

L'approssimazione  $h(x)$  è tanto migliore quanto più è piccolo l'intorno  $\partial x$  del punto di linearizzazione  $x_0$ .

### 3.2 Linearizzazione di un sistema dinamico

I sistemi dinamici reali non sono mai perfettamente lineari, ma possono essere approssimati nell'intorno di ogni prefissato movimento (come un punto di equilibrio) mediante opportuni modelli lineari, detti *modelli linearizzati*.

**Obiettivo:** costruire un modello dinamico lineare che approssimi bene il comportamento del sistema dinamico non lineare nell'intorno di un prefissato movimento "nominale".

Dato un sistema dinamico, a dimensione finita, MIMO, a tempo continuo, non lineare, stazionario.

$$\begin{cases} \dot{x}(t) = f(x(t), u(t)) \\ \dot{y}(t) = g(x(t), u(t)) \end{cases} \quad (3.3)$$

se ne considerino due diverse evoluzioni temporali:

- Un movimento "nominale"  $x_n(t)$  ottenuto applicando un ingresso "nominale"  $u_n(t)$  al sistema posto in uno stato iniziale "nominale"  $x_{n0}$ , cui corrisponde una uscita "nominale"  $y_n(t)$ .

$x_n(t)$  e  $y_n(t)$  soddisfano il seguente sistema di equazioni:

$$\begin{cases} \dot{x}_n(t) = f(x_n(t), u_n(t)), x_n(t=0) = x_{n0} \\ \dot{y}_n(t) = g(x_n(t), u_n(t)) \end{cases} \quad (3.4)$$

- Un movimento "perturbato"  $x(t)$  ottenuto applicando un ingresso "perturbato"  $u(t)$  al sistema posto in uno stato iniziale "perturbato"  $x_0$  cui corrisponde un'uscita "perturbata"  $y(t)$ .

$x(t)$  e  $y(t)$  soddisfano il seguente sistema di equazioni:

$$\begin{cases} \dot{x}(t) = f(x(t), u(t)), x(t=0) = x_0 \\ \dot{y}(t) = g(x(t), u(t)) \end{cases} \quad (3.5)$$

Le differenze tra queste 2 diverse evoluzioni temporali rappresentano le perturbazioni del sistema:

- Perturbazione sullo stato  
 $\partial x(t) = x(t) - x_n(t)$   
 $x(t) = x_n(t) + \partial x(t)$
- Perturbazione sull'ingresso  
 $\partial u(t) = u(t) - u_n(t)$   
 $u(t) = u_n(t) + \partial u(t)$
- Perturbazione sull'uscita  
 $\partial y(t) = y(t) - y_n(t)$   
 $y(t) = y_n(t) + \partial y(t)$

L'evoluzione temporale della perturbazione sullo stato  $\partial x(t)$  è soluzione dell'equazione differenziale:

$$\partial \dot{x}(t) = \frac{d}{dt} \partial x(t) = \frac{d}{dt} (x(t) - x_n(t)) = \dot{x}(t) - \dot{x}_n(t) \quad (3.6)$$

Quindi:

$$\partial \dot{x}(t) = \dot{x}(t) - \dot{x}_n(t) = f(x(t), u(t)) - f(x_n(t), u_n(t)) \quad (3.7)$$



La funzione  $f(x(t), u(t))$  può essere sviluppata in serie di Taylor in un intorno di  $x_n(t)$  e  $u_n(t)$  come:

$$\begin{aligned} f(x(t), u(t)) &= f(x_n(t) + \partial x(t), u_n(t) + \partial u(t)) = \\ &f(x_n(t), u_n(t)) + \frac{\partial f(x, u)}{\partial x} \Big|_{x=x_n, u=u_n} \partial x(t) + \frac{\partial f(x, u)}{\partial u} \Big|_{x=x_n, u=u_n} \partial u(t) + \dots \end{aligned} \quad (3.8)$$

e può essere approssimata mediante il troncamento di tale sviluppo in serie arrestato al termine lineare:

$$f(x(t), u(t)) \simeq f(x_n(t), u_n(t)) + \frac{\partial f(x, u)}{\partial x} \Big|_{x=x_n, u=u_n} \partial x(t) + \frac{\partial f(x, u)}{\partial u} \Big|_{x=x_n, u=u_n} \partial u(t) \quad (3.9)$$

Quindi:

$$\begin{aligned} \partial x(t) &= f(x(t), u(t)) - f(x_n(t), u_n(t)) \simeq \\ &\simeq \frac{\partial f(x, u)}{\partial x} \Big|_{x=x_n, u=u_n} \partial x(t) + \frac{\partial f(x, u)}{\partial u} \Big|_{x=x_n, u=u_n} \partial u(t) = \\ &= A(t)\partial x(t) + B(t)\partial u(t) \end{aligned} \quad (3.10)$$

dove:

$$A(t) = \frac{\partial f(x, u)}{\partial x} \Big|_{x=x_n, u=u_n} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \dots & \frac{\partial f_1}{\partial x_n} \\ \dots & \dots & \dots \\ \frac{\partial f_n}{\partial x_1} & \dots & \frac{\partial f_n}{\partial x_n} \end{bmatrix} \Big|_{x=x_n, u=u_n} \quad (3.11)$$

è detta: *Jacobiano di f rispetto ad x*

mentre:

$$B(t) = \frac{\partial f(x, u)}{\partial u} \Big|_{x=x_n, u=u_n} = \begin{bmatrix} \frac{\partial f_1}{\partial u_1} & \dots & \frac{\partial f_1}{\partial u_n} \\ \dots & \dots & \dots \\ \frac{\partial f_n}{\partial u_1} & \dots & \frac{\partial f_n}{\partial u_n} \end{bmatrix} \Big|_{x=x_n, u=u_n} \quad (3.12)$$

è detta: *Jacobiano di f rispetto ad u*

Procedendo allo stesso modo per  $\partial y(t)$ , si ottiene:

$$\begin{aligned} \partial y(t) &= y(t) - y_n(t) = g(x(t), u(t)) - g(x_n(t), u_n(t)) \simeq \\ &\simeq \frac{\partial g(x, u)}{\partial x} \Big|_{x=x_n, u=u_n} \partial x(t) + \frac{\partial g(x, u)}{\partial u} \Big|_{x=x_n, u=u_n} \partial u(t) = \\ &= C(t)\partial x(t) + D(t)\partial u(t) \end{aligned} \quad (3.13)$$

dove:

$$C(t) = \frac{\partial g(x, u)}{\partial x} \Big|_{x=x_n, u=u_n} = \begin{bmatrix} \frac{\partial g_1}{\partial x_1} & \dots & \frac{\partial g_1}{\partial x_n} \\ \dots & \dots & \dots \\ \frac{\partial g_q}{\partial x_1} & \dots & \frac{\partial g_q}{\partial x_n} \end{bmatrix} \Big|_{x=x_n, u=u_n} \quad (3.14)$$

è detta: *Jacobiano di g rispetto ad x*

e:

$$D(t) = \frac{\partial g(x, u)}{\partial u} \Big|_{x=x_n, u=u_n} = \begin{bmatrix} \frac{\partial g_1}{\partial u_1} & \dots & \frac{\partial g_1}{\partial u_p} \\ \dots & \dots & \dots \\ \frac{\partial g_q}{\partial u_1} & \dots & \frac{\partial g_q}{\partial u_p} \end{bmatrix} \Big|_{x=x_n, u=u_n} \quad (3.15)$$

è detta: *Jacobiano di g rispetto ad u*

# Capitolo 4

## Modello Simulink

Dopo aver scaricato il pacchetto di supporto Simulink per i minidroni Parrot (*Simulink Support Package for Parrot Minidrones*), scrivendo sulla Command Window di MATLAB il comando: `asbQuadcopterStart` si ottiene questa schermata:

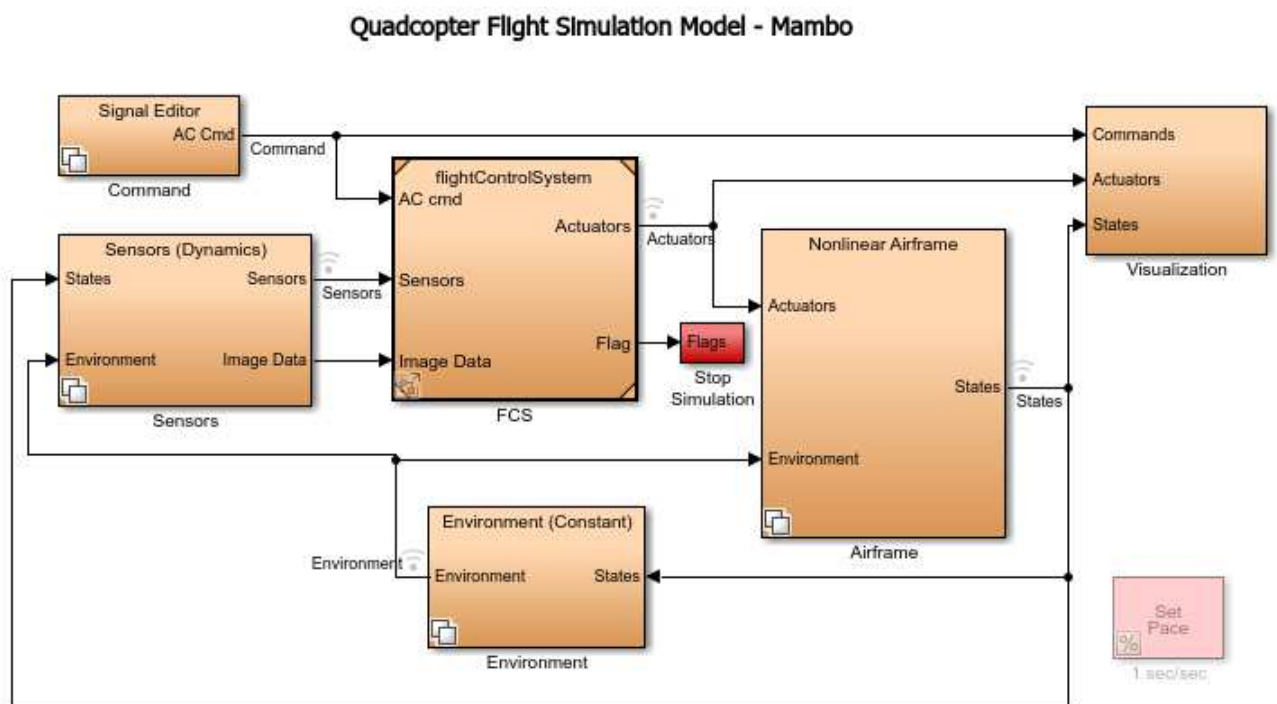


Figura 4.1: Modello Simulink asbQuadcopter

Si può notare la presenza di più sottosistemi:

- *Command*: generatore di segnali di riferimento i quali entrano in ingresso al controllore che li utilizza per mantenere il drone in posizione di *hovering*;
- *Sensors*: Tale blocco contiene le costanti di trasduzione e le relazioni tra i diversi sensori presenti a bordo;
- *Environment*: al suo interno viene costruito un vettore contenente tutte le variabili ambientali che agiscono durante il volo del quadricottero;
- *Airframe*: modello del processo;
- *Visualization*: con il seguente blocco è possibile effettuare una simulazione del volo del drone;
- *flightControlSystem* : blocco di controllo che contiene i PID di default per ogni grado di libertà. Tale blocco viene considerato per ultimo per poi passare alla progettazione di nuovi controllori per il pitch e per il roll;

Si considera ora ciascun blocco per darne una descrizione dettagliata.

## 4.1 Command

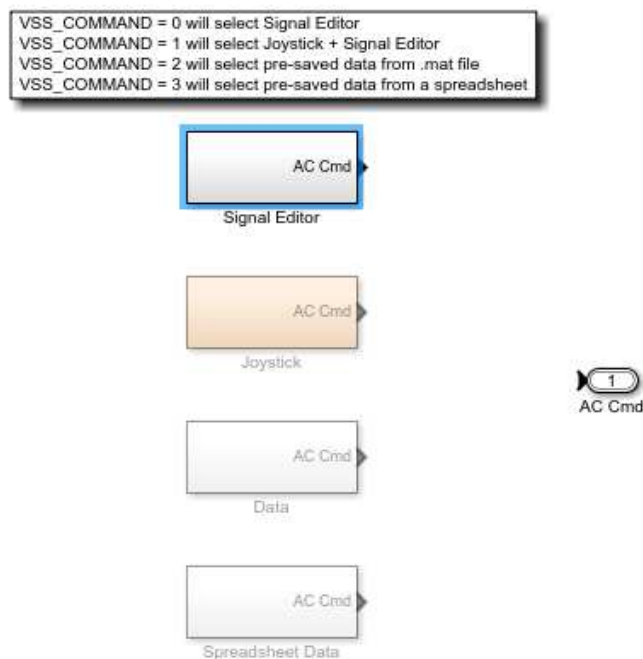


Figura 4.2: Command

Accedendo a tale blocco si possono notare 4 sotto-blocchi. E' possibile selezionarne uno solo alla volta attraverso la variabile **VSS\_COMMAND**:

- Signal Editor : attraverso il comando  $VSS\_COMMAND=0$  si seleziona il costruttore di segnali di default;
- Joystick : attraverso il comando  $VSS\_COMMAND=1$  si può gestire manualmente il comportamento del veicolo;
- Data : attraverso il comando  $VSS\_COMMAND=2$  si possono selezionare segnali direttamente da un file .mat;
- Spreadsheet Data : attraverso il comando  $VSS\_COMMAND=3$  si possono selezionare segnali da un foglio di calcolo;

Accedendo al blocco Signal Editor:

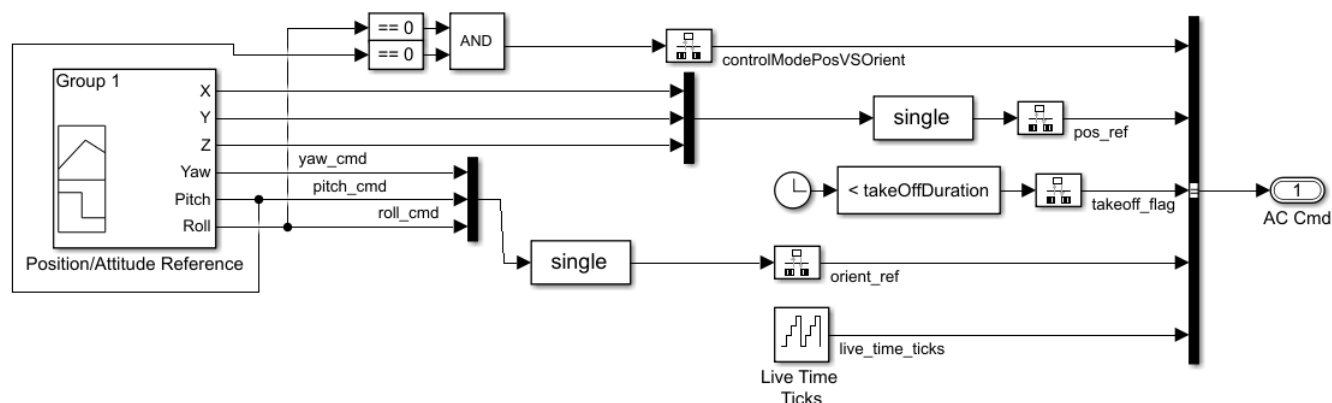


Figura 4.3: Signal Editor

si può notare che i segnali X,Y,Z vanno a rappresentare il riferimento di posizione del quadrirotore (*pos\_ref*) mentre i segnali Yaw, Pitch e Roll costituiscono il vettore di orientamento del drone (*orient\_ref*).

Inoltre una porta AND prende in ingresso i valori del pitch e del roll. Se questi sono entrambi nulli viene selezionata la modalità di controllo della posizione. Al contrario, se almeno uno dei valori è diverso da zero viene selezionata la modalità di controllo dell'orientamento.

All'interno del blocco *Position/Attitude Reference* è possibile visualizzare e modificare l'andamento delle variabili X,Y,Z,Yaw, Pitch e Roll.

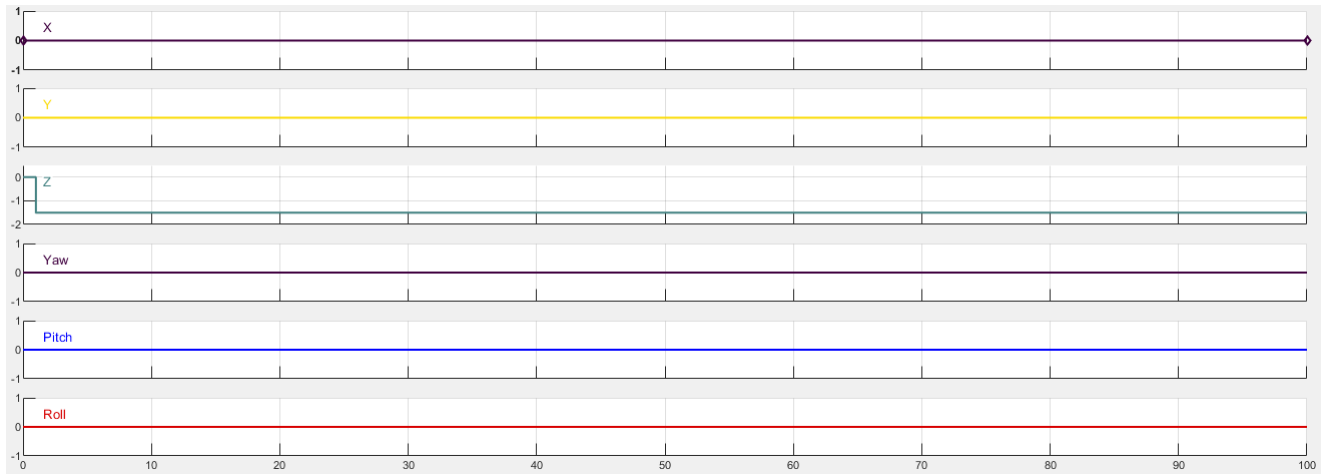


Figura 4.4: Position/Attitude Reference

## 4.2 Sensor

Tale blocco prende come variabili in ingresso il vettore degli stati e le variabili d'ambiente e modella il comportamento dei vari sensori.

E' possibile, attraverso la variabile **VSS\_SENSOR**, scegliere fra due modelli, il primo con sensori di tipo ideale( $VSS\_SENSOR=0$ ), mentre il secondo con sensori più vicini alla realtà ovvero che presentano rumore e imprecisione( $VSS\_SENSOR=1$ ).

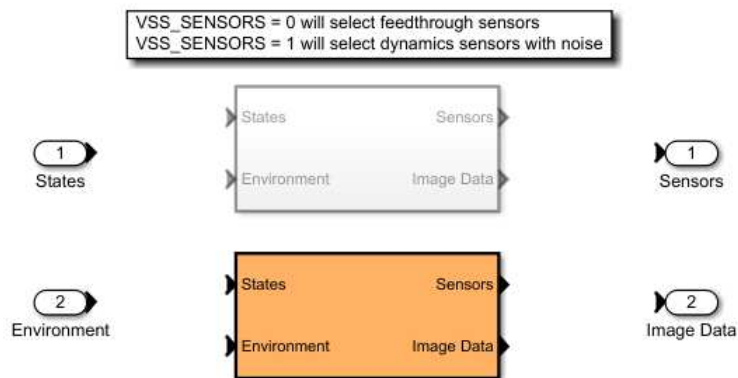


Figura 4.5: Sensor

All'interno di *Sensors Dynamics*, cliccando su *Sensor System* si ritrovano due sottoblocchi di cui uno si occupa di elaborare le immagini della fotocamera (Camera), mentre il secondo elabora i dati provenienti dal sensore IMU (Inertial Measurement Unit) ovvero da un sistema avionico che implementa il sistema di navigazione inerziale di un aeromobile.

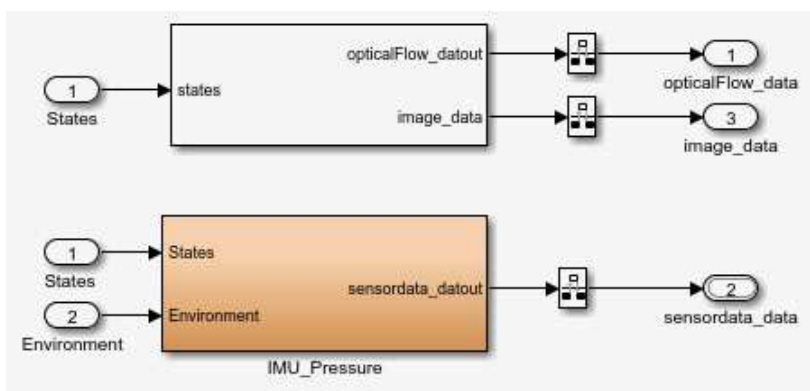


Figura 4.6: Sensor System

### 4.3 Environment

In tale blocco vengono definiti i dati riguardanti l'ambiente con cui il drone viene a contatto.

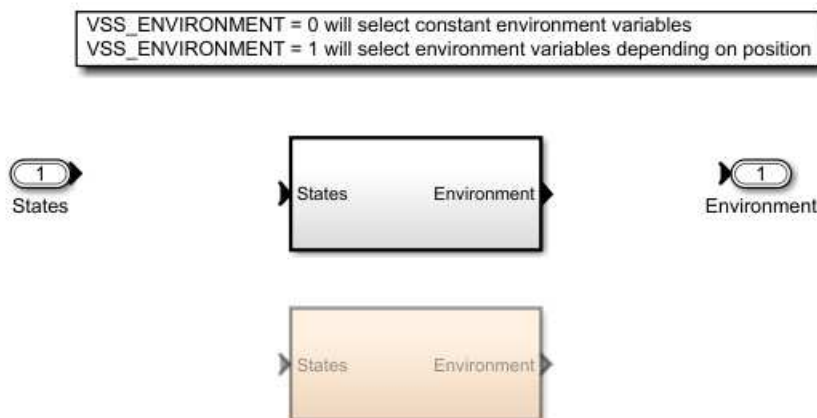


Figura 4.7: Environment

Attraverso la variabile **VSS\_ENVIRONMENT** si possono selezionare due modalità:

- $VSS\_ENVIRONMENT = 0$ : permette di selezionare variabili d'ambiente costanti;
- $VSS\_ENVIRONMENT = 1$ : permette di selezionare variabili d'ambiente dipendenti dalla posizione;

Il primo sotto-blocco costruisce un vettore costituito da:

- la costante di gravità;
- le costanti atmosferiche come: densità e temperatura dell'aria, velocità del suono, pressione atmosferica;
- il campo magnetico;

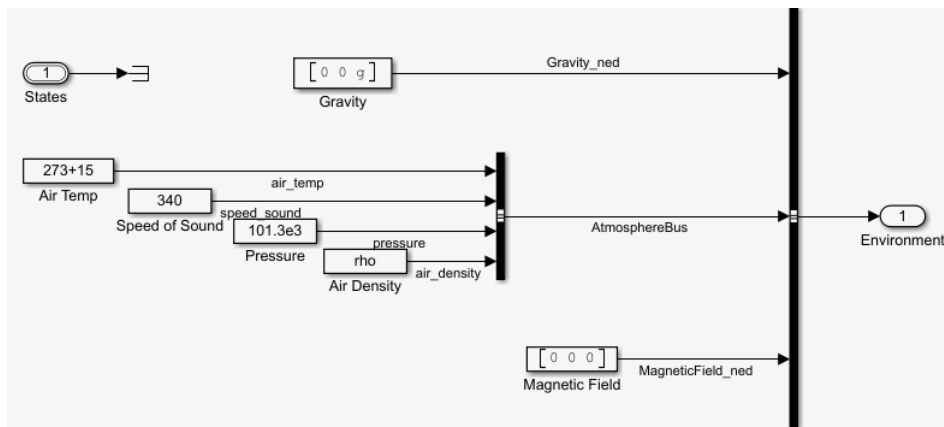


Figura 4.8: Environment(Costant)

Cliccando sul secondo sotto-blocco si avrà invece:

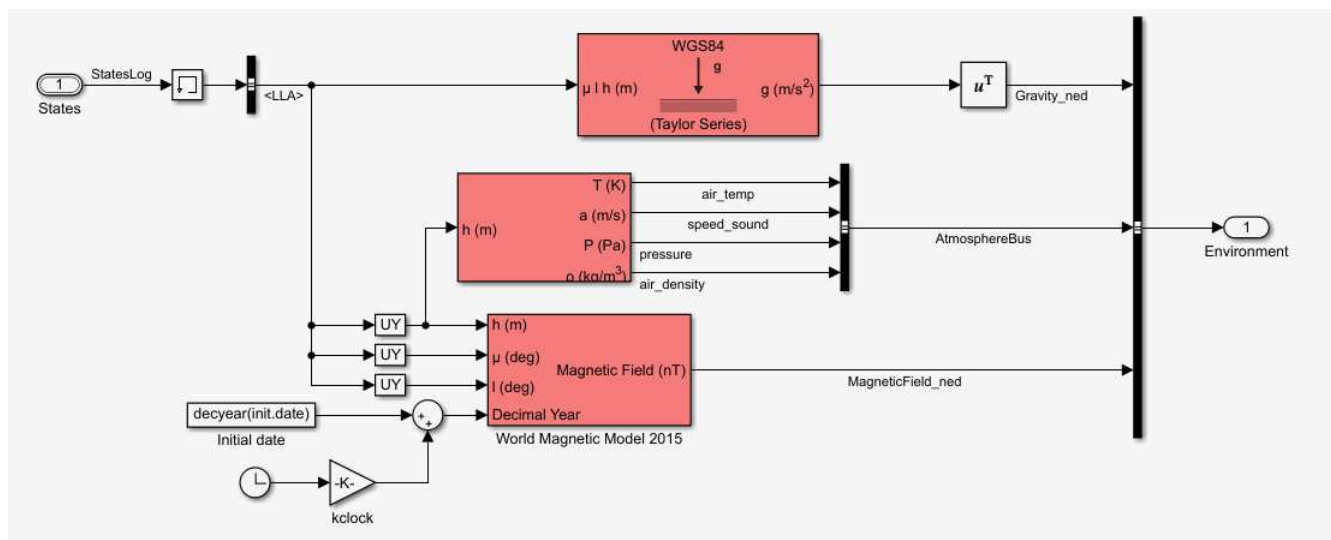


Figura 4.9: Environment(Variable)

## 4.4 Visualization

Con il seguente blocco si ha la possibilità di visualizzare le variabili di stato del drone e le velocità angolari degli attuatori al fine di ottenere una simulazione di volo del drone.

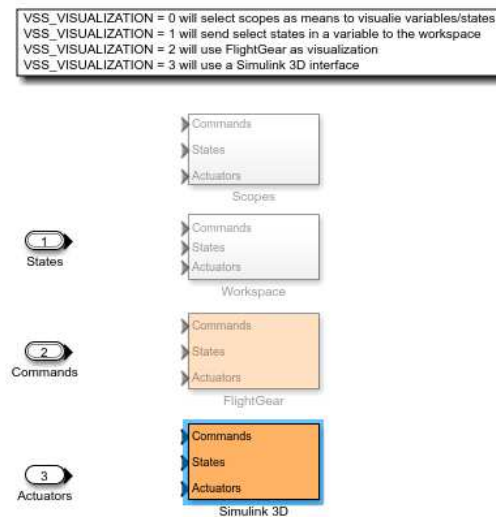


Figura 4.10: Visualization

Scrivendo nella Command Window di MATLAB il seguente comando: **VSS\_VISUALIZATION=3**, verrà selezionato il quarto sotto-blocco: *Simulink 3D*, il quale permette di visualizzare il volo in 3D del quadrirotore.

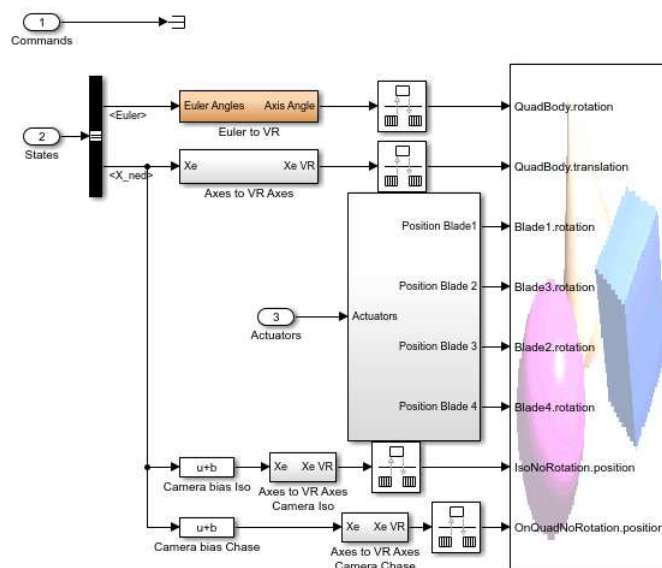


Figura 4.11: Simulink 3D



## 4.5 Airframe

Tale blocco rappresenta il modello del processo e contiene al suo interno le equazioni matematiche trovate nel capitolo 2.

Cliccando sul blocco in questione ci troviamo di fronte alla schermata:

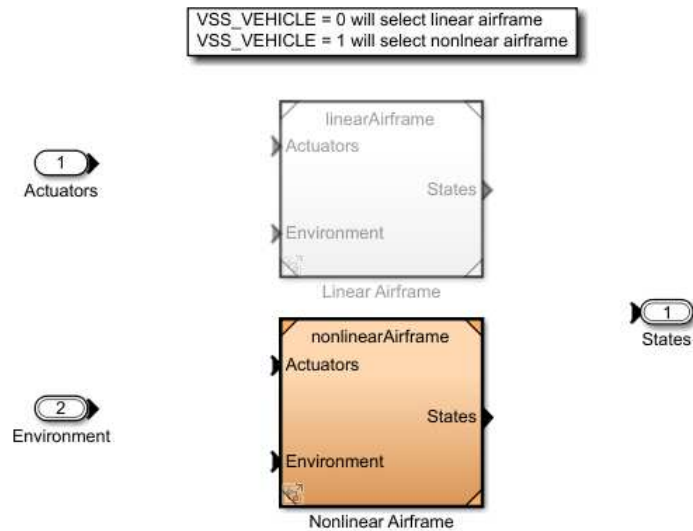


Figura 4.12: Airframe

In base a come viene settata la variabile **VSS\_VEHICLE** si può accedere al primo o al secondo sotto-blocco.

- $VSS\_VEHICLE=0$ : viene selezionato il modello lineare;
- $VSS\_VEHICLE=1$ : viene selezionato il modello non lineare;

Verrà ora considerato il modello non lineare del quadricottero lasciando in seguito la descrizione del modello lineare, che servirà per ottenere le funzioni di trasferimento relative ai due gradi di libertà del pitch e del roll e di conseguenza alla costruzione di nuovi controllori.

### 4.5.1 Modello non lineare

Il blocco *Nonlinear Airframe* è costituito a sua volta da 3 sotto-blocchi:

- AC model
- 6DOF (Quaternion)
- Bus setup

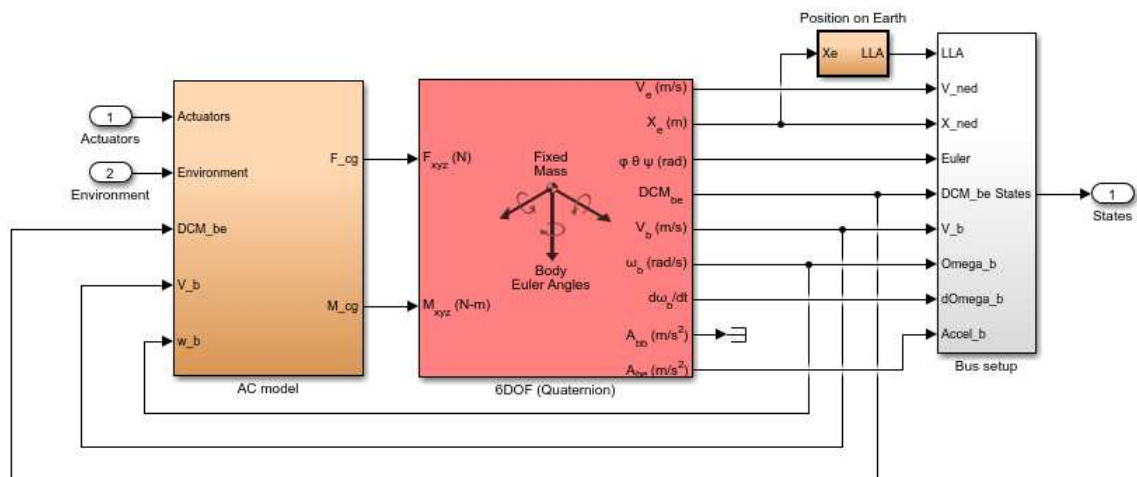


Figura 4.13: Nonlinear Airframe

## AC model

Tale blocco prende in ingresso:

- l'azione di spinta degli attuatori;
- le variabili provenienti dal blocco *Environment*;
- la matrice di rotazione DCM\_be che permette di passare dal Body Fixed Frame all'Inertial Frame;
- la velocità lineare  $V_b$ , misurata nel Body Fixed Frame;
- la velocità angolare  $\omega_b$ , misurata nel sistema di riferimento del corpo;

In uscita si avranno le forze e i momenti meccanici calcolati rispetto al centro di gravità del velivolo.

Al suo interno si può notare la presenza di quattro sotto-blocchi:

- Gravity Force Calculation;
- Drag calculation;
- Motor Forces and Torques;
- Applied Force Calculation;

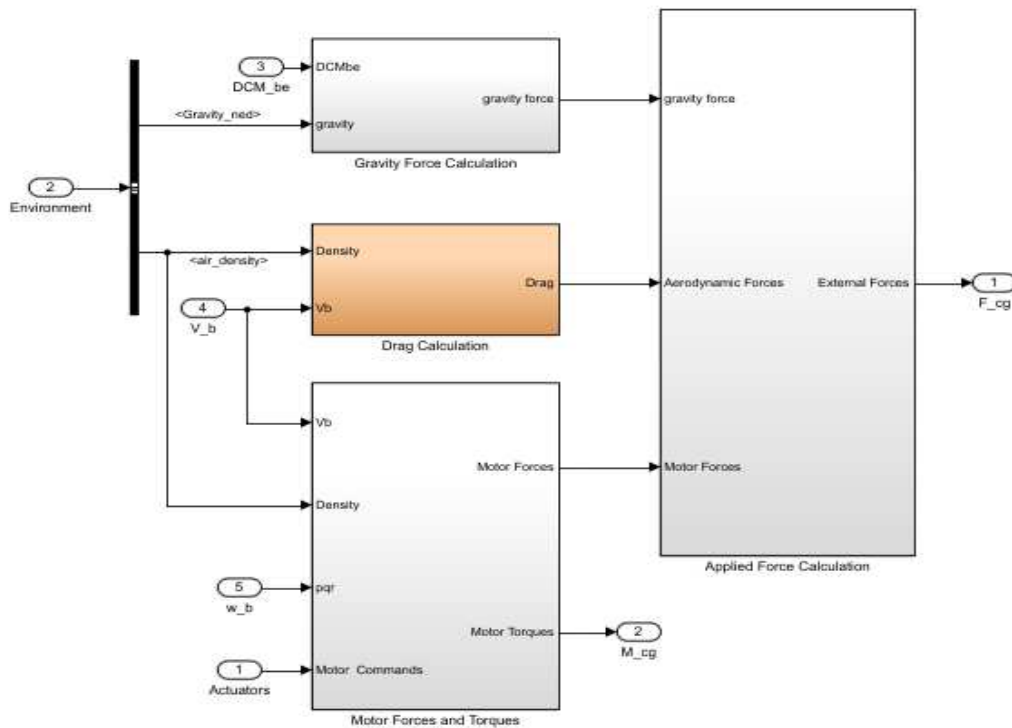


Figura 4.14: AC model

## Gravity Force Calculation

Tale blocco si occupa di calcolare la forza di gravità nel sistema corpo del drone. Come si nota nella figura sotto riportata, viene effettuata una trasformazione di coordinate dal sistema di riferimento solidale con il corpo del drone al sistema di riferimento terrestre utilizzando la matrice di rotazione DCM\_be (indicata come  $R_b^i$  nel capitolo 2).

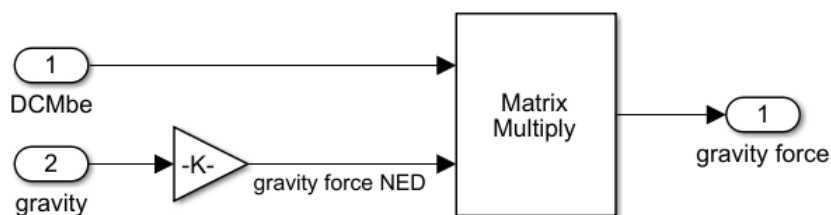


Figura 4.15: Gravity Force Calculation

## Drag Calculation

Nel modello Simulink è possibile notare la presenza di forze aerodinamiche le quali erano state trascurate nel capitolo 2 avendo considerato piccoli spostamenti intorno ad un punto di equilibrio.

Tali forze sono espresse come :

$$\vec{F}_D = \frac{1}{2}\rho C_d A \vec{V}_b |\vec{V}_b| \quad (4.1)$$

dove  $\rho$  rappresenta la densità dell'aria,  $A$  è la superficie impattata,  $\vec{V}_b$  è la velocità nel Body Fixed Frame e  $C_d$  è il coefficiente di drag.

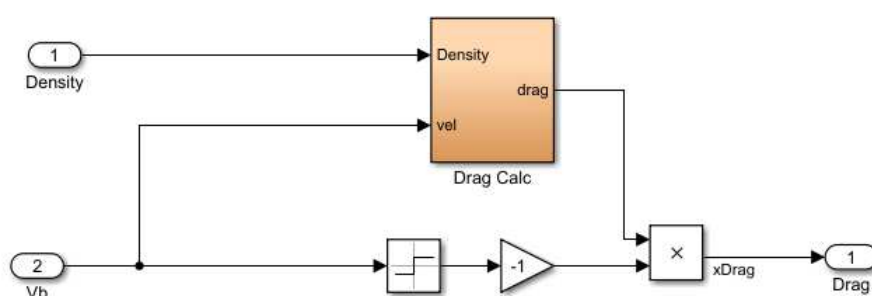


Figura 4.16: Drag Calculation

La velocità  $V_b$  viene moltiplicata per -1 dato che  $\vec{F}_D$  è una forza di attrito viscoso. Nella figura sottostante viene riportato il calcolo di  $\vec{F}_D$ .

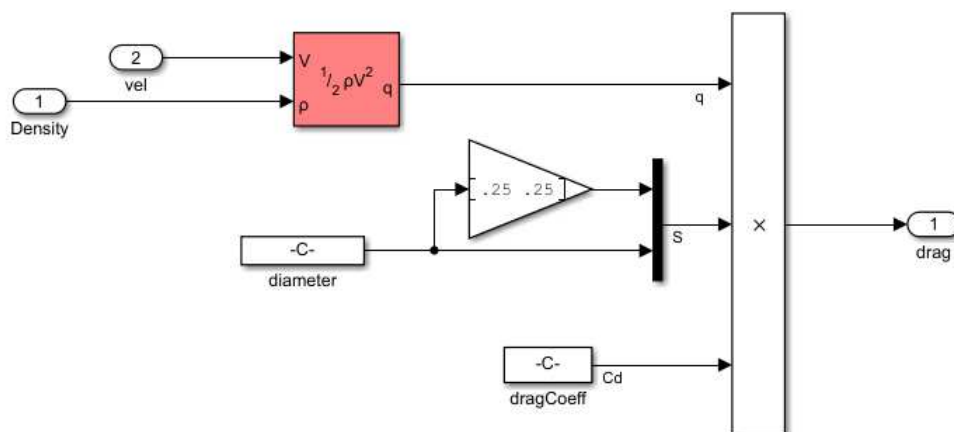


Figura 4.17: Forza aerodinamica

## Motor Forces and Torques

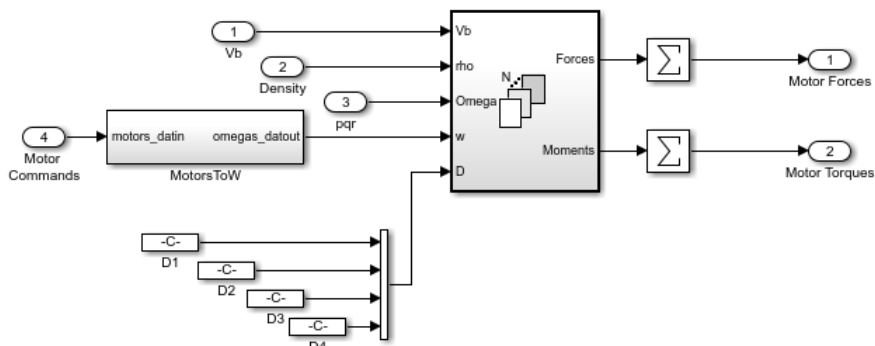


Figura 4.18: Motor Forces and Torques

Il blocco *MotorsToW* prende in ingresso il valore degli attuatori e li converte in segnali da dare in ingresso ai motori.

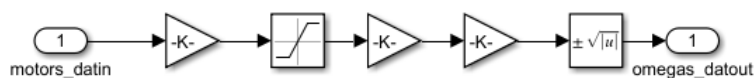


Figura 4.19: MotorsToW

Mentre il blocco *For Each Subsystem* prende in ingresso le velocità lineari nel Body Fixed Frame ( $V_b$ ), le velocità angolari nel sistema di riferimento solidale al corpo del drone ( $\omega_b$ ), la densità dell'aria  $\rho$ , l'uscita del blocco *MotorsToW* e il vettore  $D$  che rappresenta la posizione rispetto al centro di massa dei rotori. Restituisce, in uscita, le forze e i momenti.

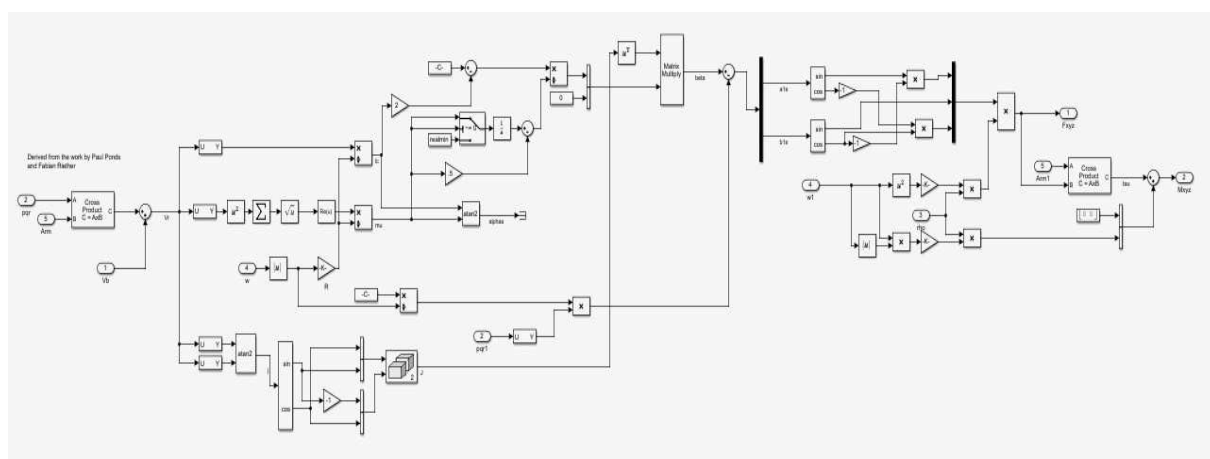


Figura 4.20: For Each Subsystem

Per poter dare una descrizione più accurata di questo sotto-blocco sarà necessario considerare dei fenomeni che sono stati fin qui trascurati per semplicità.

Quando il velivolo si muove in una certa direzione l'efficienza delle eliche anteriori e posteriori non è la stessa, si viene quindi a creare una differenza tra l'innalzamento delle lame anteriori e posteriori, provocando al quadricottero una perdita di stabilità. Tale fenomeno prende il nome di *Blade Flipping*.

Per modellare tale fenomeno è utile calcolare il cosiddetto *Advance Ratio* ovvero il rapporto di avanzamento ( $\mu_i$ ):

$$\mu_i = \frac{\|v_{(1,2)}\|}{\omega_i r} \quad (4.2)$$

dove del vettore  $v$  definito come:  $v = v_b + \omega_b \times d_i$  se ne prendono le prime due componenti (longitudinale, trasversale). Al denominatore invece vi è la velocità angolare dell' $i$ -esimo rotore  $\omega_i$  e il raggio rotorico  $r$ . Il vettore  $d_i$  tiene conto non solo dello scostamento longitudinale dei motori ma anche dello spostamento verticale rispetto al centro di gravità del corpo. Può essere espresso come:

$$\vec{d}_N = [0 \quad d \quad h]^T \quad (4.3)$$

$$\vec{d}_S = [0 \quad -d \quad h]^T \quad (4.4)$$

$$\vec{d}_E = [d \quad 0 \quad h]^T \quad (4.5)$$

$$\vec{d}_W = [-d \quad 0 \quad h]^T \quad (4.6)$$

dove  $h$  è l'altezza dei rotori rispetto al centro di gravità del velivolo e  $d$  è la lunghezza del braccio.

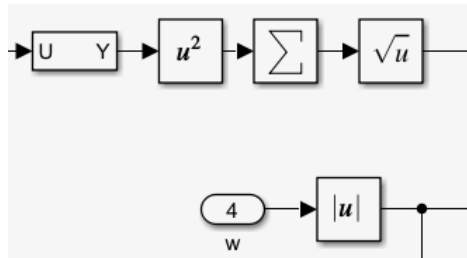


Figura 4.21: Calcolo dell'Advance Ratio

Gli angoli di orientamento rispetto alla direzione di avanzamento del drone sono calcolati tramite la seguente equazione:

$$\mu_i = \frac{\mu_i(4\theta_t - 2\lambda^2)}{1 - \frac{\mu_i^2}{2}} \quad (4.7)$$

$$v_i = \frac{\frac{8\gamma C_T \mu_i}{9\alpha\theta} + \frac{C_T}{2\mu_i}}{1 + \frac{\mu_i^2}{2}} \quad (4.8)$$

dove  $\lambda$  rappresenta il flusso di aria in ingresso alle eliche,  $\alpha$  è l'angolo di salita,  $\theta_t$  è l'angolo di pitch rispetto alla direzione di traslazione,  $C_T$  è il coefficiente di spinta mentre  $\gamma$  dipende dalla geometria del drone.

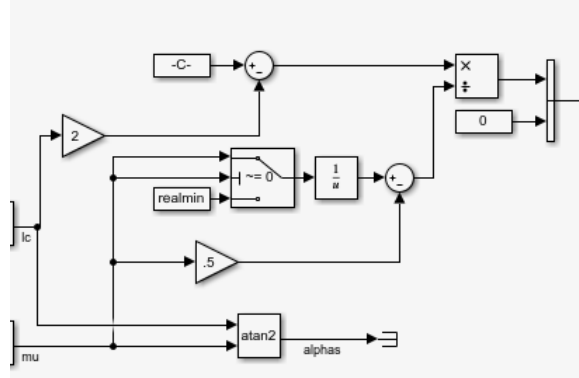


Figura 4.22: Calcolo dell'angolo di orientamento

Inoltre, indicando con  $a_i$  e  $b_i$  gli angoli formati rispettivamente dagli assi longitudinale e latitudinale del drone rispetto alla direzione di avanzamento, si può esprimere la spinta verticale come:

$$T_i = C_T \rho A r^2 \omega_i^2 \begin{bmatrix} -\sin(a_i) \\ \cos(a_i) \sin(b_i) \\ -\cos(b_i) \cos(a_i) \end{bmatrix} \quad (4.9)$$

Per quanto riguarda invece il calcolo dei momenti, dopo aver definito il vettore  $d_i$ , si può scrivere:

$$\tau_i = T_i \times d_i \quad (4.10)$$

Infine, nell'estrema destra di questo sotto-blocco, si può notare anche il contributo dovuto al drag moment ( $dm_i$ ) direttamente proporzionale al drag factor  $C_q$ .

$$dm_i = C_q \rho A r^2 \omega_i^2 \cdot r \quad (4.11)$$

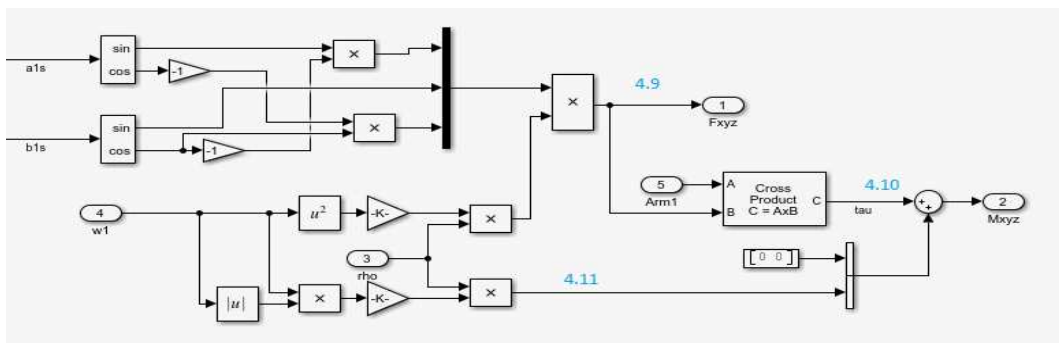


Figura 4.23: Equazioni 4.9 4.10 4.11

## 6DOF (Quaternion)

Tale blocco prende in ingresso le forze e i momenti generati dall'AC model e calcola varie grandezze tra cui: la posizione del velivolo nel sistema di riferimento terrestre, la matrice di trasformazione delle coordinate, la velocità di variazione dell'angolo di incidenza e di scivolata, le velocità e le accelerazioni angolari nel Body Fixed Frame e nell'Inertial Frame.

## Bus Setup

Crea il vettore di stato con le seguenti variabili:

- LLA: Latitudine, longitudine, altitudine;
- $V_{ned}$ : Velocità lineare nell'Inertial Frame;
- $X_{ned}$ : Posizione nell'Inertial Frame;
- Euler: Angoli di Eulero;
- $DCM_{be}$ : Matrice di rotazione dal Body Fixed Frame all'Inertial Frame;
- $V_b$ : Velocità lineare nel sistema corpo del drone;
- $\Omega_b$ : Velocità angolare nel Body Frame;
- $d\Omega_b$ : Accelerazione angolare nel Body Frame;
- $Accel_b$ : Accelerazione lineare nell'Inertial Frame;

## 4.6 Flight Control System

Il *Flight Control System* è il blocco su cui si concentrerà maggiore attenzione in quanto al suo interno sono presenti i controllori di default dei 6 gradi di libertà.

Prende in ingresso i segnali di riferimento provenienti dal *Signal Editor*, i dati elaborati dai sensori di bordo e i dati riguardanti le immagini sviluppate dalla videocamera, prima processati dall'*Image Processing System*.

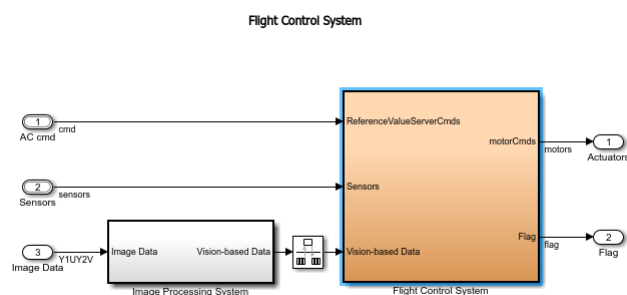


Figura 4.24: FCS



Accedendo al blocco ci si trova davanti questa schermata:

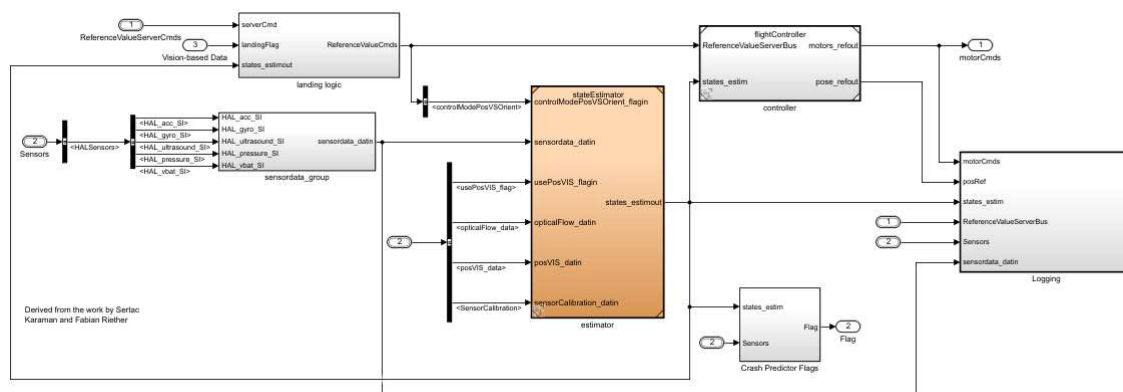


Figura 4.25: Flight Control System

Si può notare la presenza di 4 sotto-blocchi:

- Estimator: si occupa di fornire agli altri blocchi i valori dello stato del sistema;
- Crash Predictor Flags: Attraverso la variabile binaria *flag* si può fermare la simulazione nel caso di condizioni anomale di volo;
- Logging: Tale blocco permette di visualizzare l'andamento di tutte le variabili;
- Controller: Contiene i PID di tutti i gradi di libertà. Tale blocco verrà analizzato in dettaglio nel capitolo successivo dove verranno sostituiti i PID relativi al pitch e al roll con nuovi controllori;

## Estimator

Tale blocco prende in ingresso i segnali provenienti dai diversi sensori e restituisce in uscita una stima delle variabili di stato.

Al suo interno si possono notare 4 sotto-blocchi:

- Sensor Processing: Come indicato nel nome, tale blocco si occupa del pre-processamento dei dati provenienti dall'IMU;
- Complementary Filter: Riceve in ingresso i dati provenienti dal Sensor Processing e calcola una stima degli angoli di yaw, pitch e roll e le velocità angolari nel sistema di riferimento solidale al corpo del drone ( $p,q,r$ );
- Estimator Altitude: Si occupa di determinare una stima della quota e della velocità verticale del quadricottero;
- Estimator XY Position: Determina una stima della posizione del drone (X,Y) e della relativa velocità longitudinale e latitudinale ( $dX,dY$ );

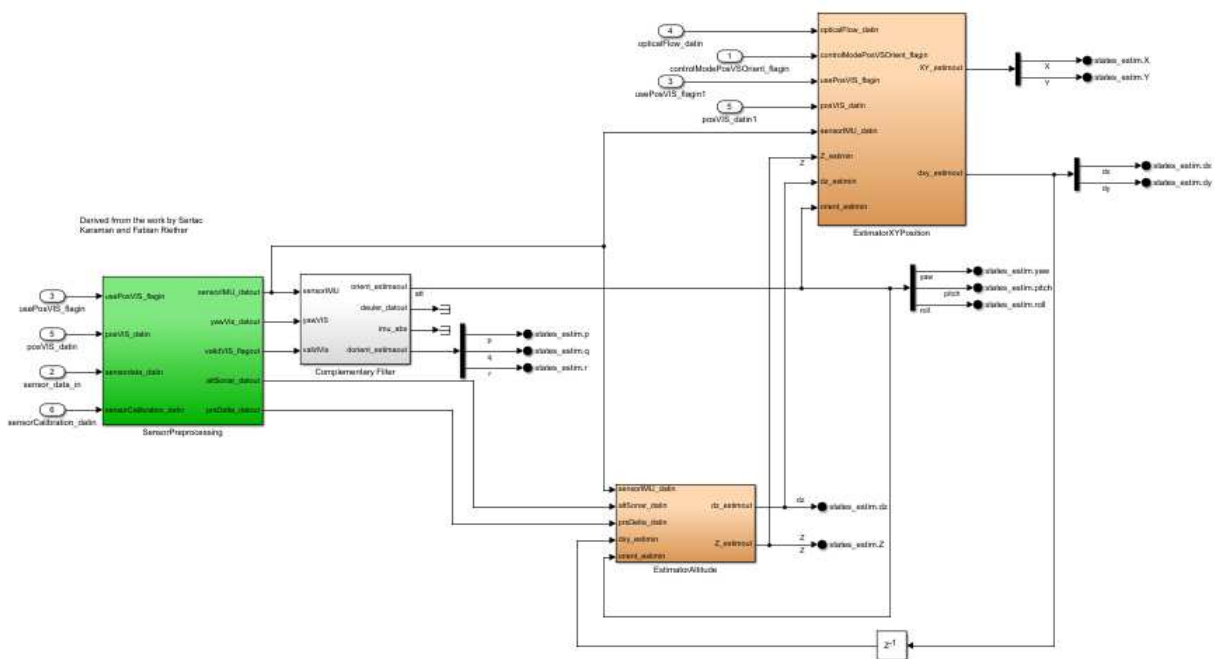


Figura 4.26: Estimator

Tali variabili verranno fornite in ingresso al *Controller*.

## Logging

All'interno di *Logging*, attraverso i blocchi *Scope*, è possibile visualizzare l'andamento delle variabili X,Y,Z,Yaw,Pitch e Roll.

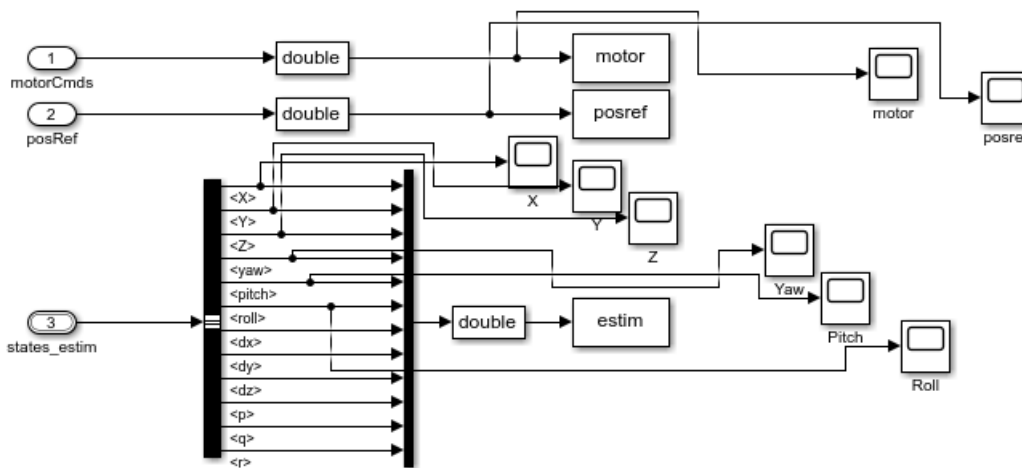


Figura 4.27: Logging

# Capitolo 5

## Calcolo della funzione di trasferimento di Pitch e Roll

Per poter sviluppare dei nuovi controllori sui due gradi di libertà del Pitch e del Roll, sarà necessario considerare il modello lineare presente all'interno di Simulink, al fine di isolare la dinamica relativa a questi 2 gradi di libertà.

Si potrà, dopo aver trovato le matrici A,B,C,D, ricavare le funzioni di trasferimento del pitch e del roll, attraverso cui, in seguito, si progetteranno dei controllori con la sintesi in  $\omega$  e con il luogo delle radici, ovvero tecniche di controllo lineare.

Come già detto in precedenza, è possibile ottenere il modello lineare settando la variabile **VSS\_VEHICLE** a 0.

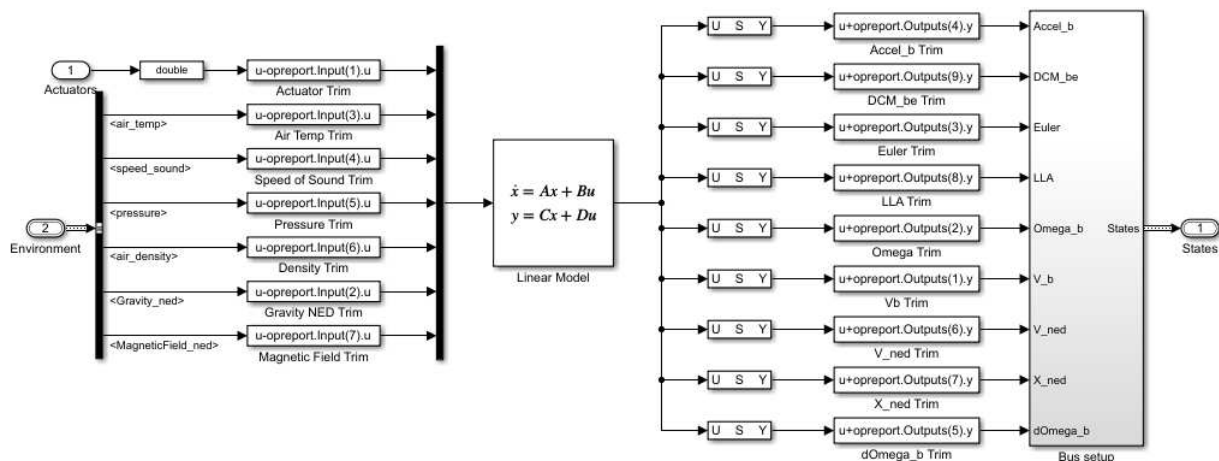
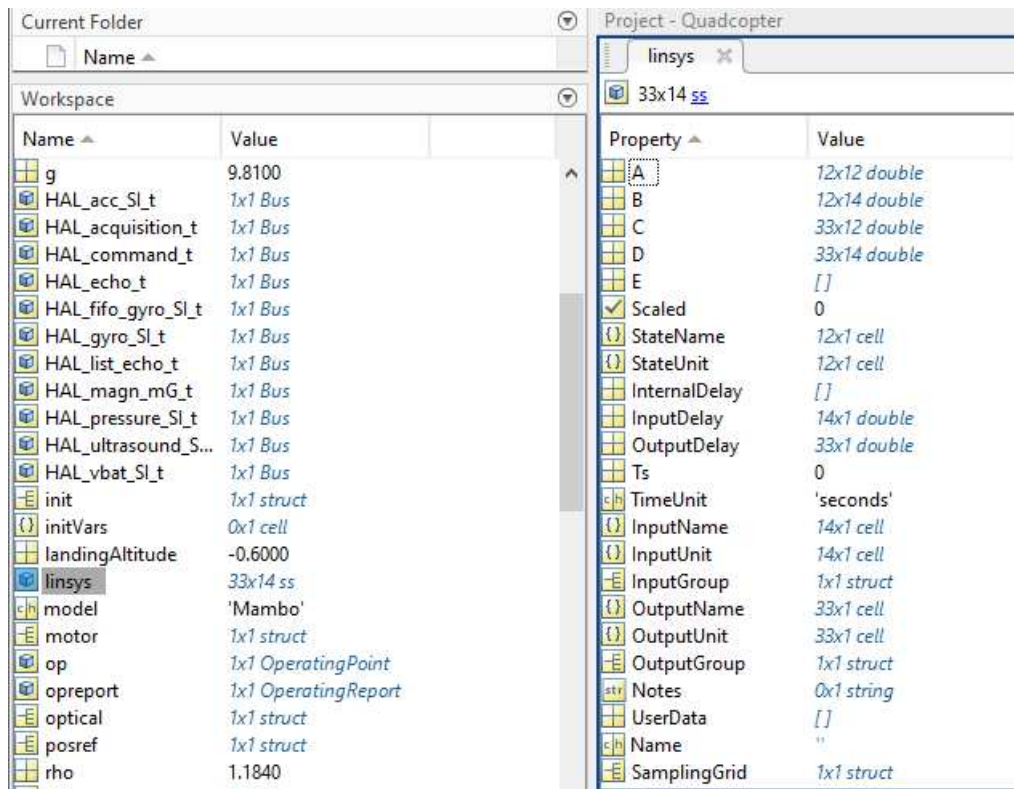


Figura 5.1: Modello lineare

Tale modello (ottenuto tramite la funzione *trimLinearizeOpPoint* presente all'interno del pacchetto *Simulink Control Design*) prende in ingresso la spinta degli attuatori (*Actuators*), la temperatura e la densità dell'aria, la velocità del suono, la pressione atmosferica, il vettore gravità e il vettore campo magnetico calcolati nel sistema di riferimento inerziale. D'altra parte le uscite sono le stesse del sistema non lineare.

## 5.1 Funzione di trasferimento del Pitch

Per poter trovare la funzione di trasferimento relativa al grado di libertà del Pitch dovremo accedere ad una struttura dati presente nella *Workspace* di MATLAB: *linsys*. Tale struttura sarà accessibile solo dopo aver avviato per la prima volta la simulazione.



The screenshot shows the MATLAB workspace and variable browser. The workspace contains various variables, with 'linsys' selected. The variable browser shows the properties of 'linsys', which is a 33x14 state-space system.

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

Figura 5.2: Struttura dati: linsys

E' possibile notare al suo interno:

- La matrice A: matrice della dinamica  $12 \times 12$ ;
- La matrice B: matrice degli ingressi  $12 \times 14$ ;
- La matrice C: matrice delle uscite  $33 \times 12$ ;
- La matrice D: matrice  $33 \times 14$  che descrive il legame ingresso-uscita;
- StateName: Vettore delle variabili di stato  $12 \times 1$ ;
- InputName: Vettore delle variabili in ingresso  $14 \times 1$ ;
- OutputName: Vettore delle variabili di uscita  $33 \times 1$ ;

Prima di passare al calcolo delle varie matrici si ricorda che un sistema lineare, stazionario e tempo invariante assume la seguente forma:

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t) \\ y(t) = Cx(t) + Du(t) \end{cases} \quad (5.1)$$

Scegliendo come vettore di stato:

$$x(t) = \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} \quad (5.2)$$

dove  $\theta$  è l'angolo di pitch e  $\dot{\theta}$ , per le ipotesi di linearizzazione, coincide con  $q$ , ovvero la velocità angolare del pitch nel Body Fixed Frame.

Si può dunque riscrivere la 5.1 nel seguente modo:

$$\begin{cases} \begin{bmatrix} \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = A \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} + BU_2 \\ y(t) = C \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} + DU_2 \end{cases} \quad (5.3)$$

dove  $U_2$  viene definito nel capitolo 2 come :  $\text{lb}(-\omega_4^2 + \omega_1^2 + \omega_2^2 - \omega_3^2)$ .

Passiamo ora al calcolo delle varie matrici.

## Calcolo della matrice A

Per il calcolo della matrice della dinamica si considerano nel vettore delle variabili di stato la seconda componente (che rappresenta l'angolo di pitch  $\theta$ ) e l'undicesima (la velocità angolare relativa al pitch calcolata nel Body Fixed Frame,  $q$ ).

linsys.StateName	
	1
1	phi theta psi(1)
2	phi theta psi(2)
3	phi theta psi(3)
4	ub,vb,wb(1)
5	ub,vb,wb(2)
6	ub,vb,wb(3)
7	xe,ye,ze(1)
8	xe,ye,ze(2)
9	xe,ye,ze(3)
10	p,q,r(1)
11	p,q,r(2)
12	p,q,r(3)

Figura 5.3: StateName

Quindi per poter isolare la sola dinamica relativa al pitch si considerano nella matrice A la seconda e l'undicesima riga e la seconda e undicesima colonna.

linsys.A													
	1	2	3	4	5	6	7	8	9	10	11	12	
1	0	3.6611e-17	0	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 5.4: Matrice A

## Calcolo della matrice B

Per quanto invece riguarda il vettore degli ingressi si considera il contributo dato dalla spinta dei 4 attuatori (*Actuators*). Questo perchè la dinamica relativa al pitch è completamente dipendente dagli ingressi.

linsys.InputName	
	1
1	Actuators(1)
2	Actuators(2)
3	Actuators(3)
4	Actuators(4)
5	AtmosphereBus.air_density
6	AtmosphereBus.air_temp
7	AtmosphereBus.pressure
8	AtmosphereBus.speed_sound
9	Gravity_ned(1)
10	Gravity_ned(2)
11	Gravity_ned(3)
12	MagneticField_ned(1)
13	MagneticField_ned(2)
14	MagneticField_ned(3)

Figura 5.5: InputName

Per il calcolo della matrice B bisognerà quindi considerare la seconda riga (riga nulla) e l'undicesima (dove per quanto detto sopra prenderemo il coefficiente, presente nelle prime 4 colonne, che andrà a moltiplicare la spinta dei 4 attuatori).

linsys.B												
	1	2	3	4	5	6	7	8	9	10	11	12
1	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
3	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
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
6	-0.0096	0.0096	-0.0096	0.0096	-8.2855	0	0	0	0	0	1.0000	0
7	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
9	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
11	0.3133	-0.3133	-0.3133	0.3133	1.8856e-14	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

Figura 5.6: Matrice B

### Calcolo della matrice C

Del vettore delle uscite sotto riportato, si considera solo la quattordicesima riga ovvero quella relativa al pitch.

linsys.OutputName	
1	
1	Accel_body(1)
2	Accel_body(2)
3	Accel_body(3)
4	DCM_be(1)
5	DCM_be(2)
6	DCM_be(3)
7	DCM_be(4)
8	DCM_be(5)
9	DCM_be(6)
10	DCM_be(7)
11	DCM_be(8)
12	DCM_be(9)
13	Euler(1)
14	Euler(2)
15	Euler(3)
16	LLA(1)
17	LLA(2)
18	LLA(3)
19	Omega_body(1)
20	Omega_body(2)
21	Omega_body(3)
22	V_body(1)
23	V_body(2)
24	V_body(3)
25	V_ned(1)
26	V_ned(2)
27	V_ned(3)
28	X_ned(1)
29	X_ned(2)
30	X_ned(3)
31	dOmega_body(1)
32	dOmega_body(2)
33	dOmega_body(3)

Figura 5.7: OutputName

Per costruire la matrice C si sceglie quindi la quattordicesima riga e le colonne 2 e 11.

### Calcolo della matrice D

La matrice D ha lo stesso numero di righe dell'uscita (33) e un numero di colonne pari al numero di righe degli ingressi (12). Anche in questo caso viene considerata quindi la quattordicesima riga che è composta da tutti 0. Possiamo dunque scrivere:  $D=0$ .

linsys.C												
	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	-5.2296e-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 5.8: Matrice C

linsys.D												
	1	2	3	4	5	6	7	8	9	10	11	12
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
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
3	-0.0096	0.0096	-0.0096	0.0096	-8.2855	0	0	0	0	0	1.0000	0
4	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
6	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
8	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
10	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
12	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
14	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
16	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
18	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
20	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
22	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
24	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
26	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
28	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
30	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
32	0.3133	-0.3133	-0.3133	0.3133	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

Figura 5.9: Matrice D



Il sistema 5.1 può essere ora riscritto come:

$$\begin{cases} \begin{bmatrix} \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & -1.6192 \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ 0.3133 \end{bmatrix} U_2 \\ y(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} \end{cases} \quad (5.4)$$

E quindi:

$$\begin{cases} \dot{\theta} = \dot{\theta} \\ \ddot{\theta} = -1.6192\dot{\theta} + 0.3133U_2 \\ y(t) = \theta \end{cases} \quad (5.5)$$

Applicando le trasformate di Laplace si ottiene:

$$\begin{cases} s^2\theta(s) = -1.6192s\theta(s) + 0.3133U(s) \\ Y(s) = \theta(s) \end{cases} \quad (5.6)$$

Da cui:

$$\begin{aligned} (s^2 + 1.6192s)\theta(s) &= 0.3133U(s) \\ \frac{\theta(s)}{U(s)} &= \frac{0.3133}{s^2 + 1.6192s} \end{aligned} \quad (5.7)$$

Per costruire il legame ingresso/uscita dobbiamo però inserire due costanti moltiplicative.

Scrivendo sulla Command Window di MATLAB il comando: *Controler.Q2Ts* si ottiene una matrice dove la terza colonna fornisce il coefficiente ( $L_1$ ) relativo al momento di pitch: 5.6659.

La seconda costante riguarda il coefficiente di spinta dei motori (ottenuto tramite il comando *Vehicle.Motor.thrustToMotorCommand*). Tale costante  $L_2$  vale:  $1.5307e^{03}$ . Si ottiene così la funzione di trasferimento relativa al pitch:

$$\begin{aligned} P(s) &= L_1 L_2 \frac{0.3133}{s^2 + 1.6192s} \\ P(s) &= \frac{2717}{s^2 + 1.6192s} \end{aligned} \quad (5.8)$$

## 5.2 Funzione di trasferimento per il Roll

Effettuando gli stessi passaggi sopra riportati e quindi andando a considerare nel vettore degli stati la prima e la decima componente ( $\phi$  e  $p$ ), nel vettore degli ingressi il contributo dovuto alla spinta dei 4 attuatori e nel vettore delle uscite la tredicesima componente, si ottengono le seguenti matrici:

$$\begin{aligned} A &= \begin{bmatrix} 0 & 1 \\ 0 & -2.1715 \end{bmatrix} \\ B &= \begin{bmatrix} 0 \\ 0.4202 \end{bmatrix} \\ C &= [1 \ 0] \\ D &= 0 \end{aligned} \tag{5.9}$$

Da cui:

$$\left\{ \begin{array}{l} \begin{bmatrix} \dot{\phi} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & -2.1715 \end{bmatrix} \begin{bmatrix} \phi \\ \dot{\phi} \end{bmatrix} + \begin{bmatrix} 0 \\ 0.4202 \end{bmatrix} U_1 \\ y(t) = [1 \ 0] \begin{bmatrix} \phi \\ \dot{\phi} \end{bmatrix} \end{array} \right. \tag{5.10}$$

Applicando poi le trasformate di Laplace e moltiplicando per le costanti sopra citate, si ottiene la funzione di trasferimento per il roll:

$$P(s) = \frac{3644}{s^2 + 2.171s} \tag{5.11}$$

# Capitolo 6

## Sintesi di controllori

### 6.1 Generalità sui PID

Il controllore PID è un algoritmo di controllo dotato di una struttura predefinita. Grazie alla sua semplicità di utilizzo è l'algoritmo di controllo più utilizzato nelle applicazioni industriali.

Dal punto di vista matematico, il regolatore PID è un sistema dinamico che prende in ingresso l'errore, ovvero la differenza fra il riferimento  $r(t)$  e la variabile controllata  $y(t)$  ( $e(t) = r(t) - y(t)$ ) e lo elabora restituendo in uscita lo sforzo di controllo  $u(t)$ .

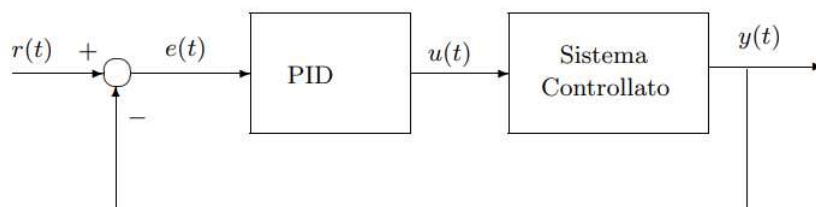


Figura 6.1: Schema di controllo in controeazione

Il controllore PID ha la seguente forma:

$$u(t) = K_p \left[ e(t) + \frac{1}{K_I} \int_0^t e(\tau) d\tau + \tau_D \frac{de(t)}{dt} \right] \quad (6.1)$$

E' quindi la somma di 3 contributi:

- Azione Proporzionale: dipende dal *guadagno proporzionale*  $K_p$  viene introdotta per due motivi principali: Il primo è che un elevato guadagno in catena diretta diminuisce l'errore di regime permanente. Il secondo è che aumentando tale guadagno aumenta la

pulsazione di attraversamento e di conseguenza la banda passante a ciclo chiuso. Diminuisce quindi il tempo di salita e il sistema è più veloce.

- Azione Derivativa: dipendente dalla costante  $\tau_D$  presenta un grande vantaggio. Nel dominio di Laplace, infatti, l'introduzione di uno zero porta un anticipo di fase di  $\frac{\pi}{2}$ . Aumenta quindi il margine di fase e migliora la stabilità.
- Azione integrale: dipendente dalla costante  $K_I$ , serve a migliorare la precisione di regime permanente. Infatti, nel dominio di Laplace l'integrale corrisponde ad un polo nell'origine ( $\frac{1}{s}$ ). Si ha cioè un sistema di tipo 1 e quindi errore a regime permanente nullo per un ingresso a gradino. Inoltre si rende il sistema astatico rispetto a disturbi che agiscono sull'uscita.

## 6.2 Flight Controller

Tale blocco prende in ingresso le variabili stimate dal blocco *Estimator* ed è costituito da 4 blocchi principali che contengono i controllori per i 6 gradi di libertà e da 3 blocchi ausiliari tra cui:

- Switch Ref Att: Permette di scegliere quale tipo di segnale mandare in ingresso al controllore del Pitch e del Roll (segnale di riferimento rispetto alla posizione oppure segnale di riferimento rispetto all'orientamento).
- Control Mixer
- Thrusts to Motor Command: insieme al control Mixer fornisce le costanti moltiplicative relative all'azione degli attuatori.

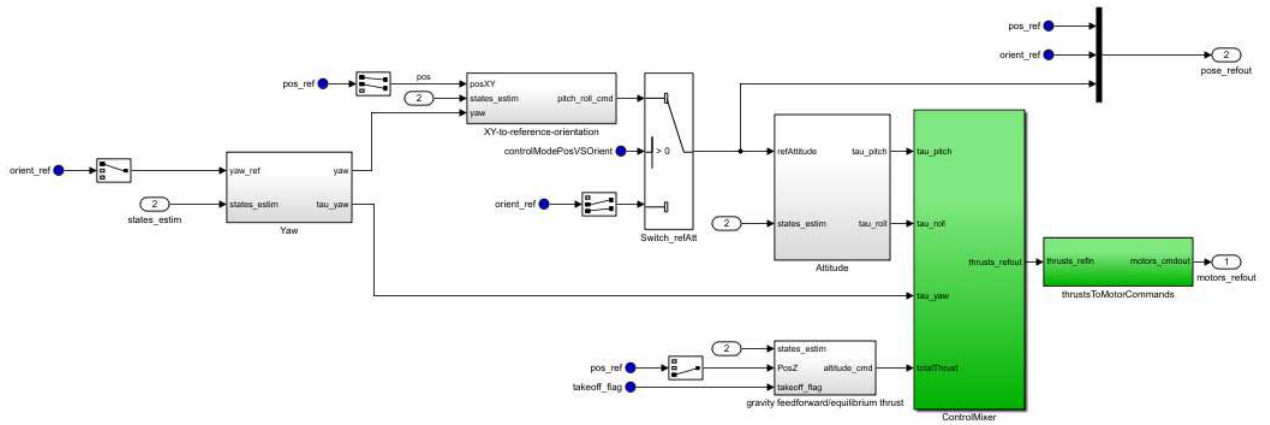


Figura 6.2: Flight Controller

Si analizzano ora i 4 blocchi principali relativi ai vari PID.

- Yaw Controller: Tale controllore è un PD, ovvero presenta soltanto l'azione derivativa e proporzionale. Come si può notare, prende in ingresso il segnale di riferimento per lo Yaw ( $yaw\_ref$ ) a cui viene sottratto il valore dello yaw proveniente dall'ingresso  $states\_estim$ . Viene generato così il segnale di errore che verrà moltiplicato per una costante  $K_P$ . A tale azione proporzionale viene sottratta l'azione derivativa (ottenuta moltiplicando la variabile  $r$  proveniente da  $states\_estim$  per la costante derivativa). Tale controllore permette di conoscere l'esatta posizione del drone.

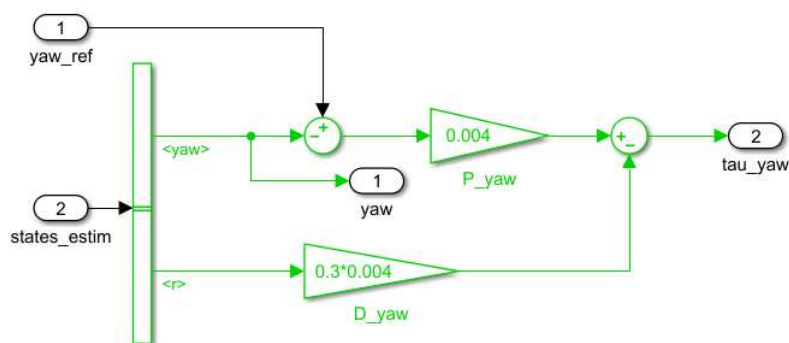


Figura 6.3: Yaw Controller

- X-Y-to-reference-orientation: Essendo il drone un sistema sotto-attuato, tale blocco fornisce dei segnali di riferimento variabili che

non saranno utilizzati direttamente per il controllo dei motori ma saranno forniti ai controllori del Pitch e del Roll.

Anche questo è un controllore Proporzionale-Derivativo. La parte superiore tiene conto del fatto che una variazione dello yaw provoca anche una variazione dello spostamento longitudinale e latitudinale.

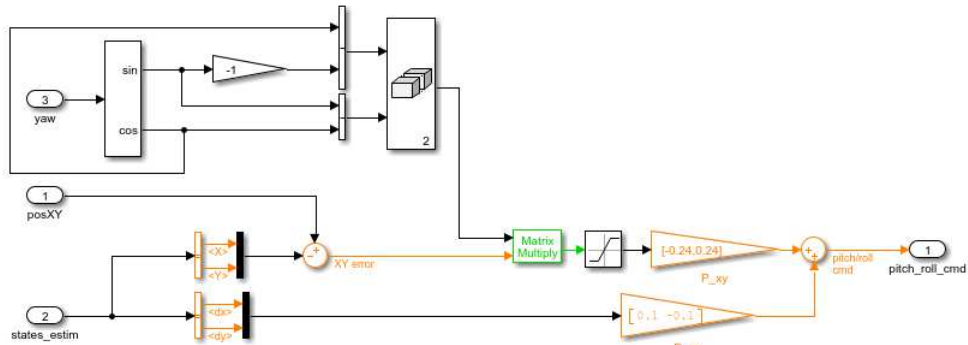


Figura 6.4: X-Y Controller

- Attitude Controller: Tale blocco prende in ingresso il segnale variabile proveniente dal precedente blocco oppure, modificando opportunamente lo switch, un segnale costante proveniente dal blocco *Signal\_Editor*. Il PID contenuto all'interno presenta sia un'azione proporzionale che derivativa e integrale.

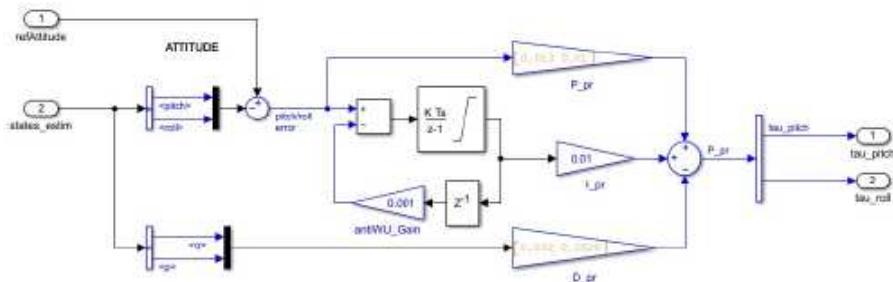


Figura 6.5: Attitude Controller

- Gravity Feedforward/Equilibrium Thrust: Tale controllore è l'unico ad essere disaccoppiato dagli altri. E' un controllore Proporzionale-Derivativo. Tramite la variabile *takeoff\_flag* è possibile dividere la fase di decollo, che richiede uno sforzo maggiore, con la fase di volo stazionario.

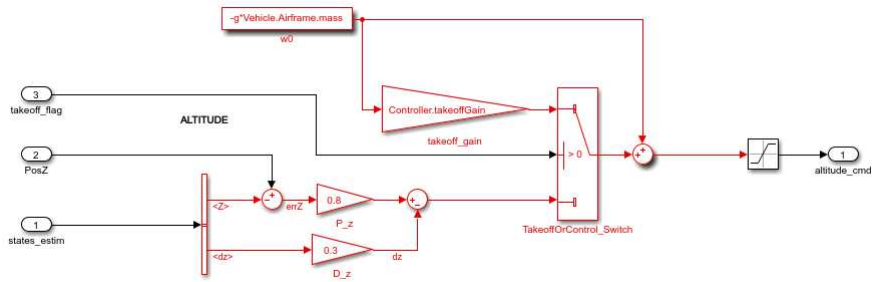


Figura 6.6: Gravity Feedforward/Equilibrium Thrust

### 6.3 PID di pitch e roll

All'interno di questa sezione, verrà mostrato l'andamento delle variabili di pitch e roll, controllate tramite regolatori PID. Verranno presi in considerazione due tipi di riferimento:

- Riferimento costante proveniente dal blocco *Signal Editor*;
- Riferimento variabile derivante dal blocco *XY-To-Reference-Orientation*;

Inoltre, verrà studiato l'andamento di tali variabili, sia nel modello lineare che non lineare.

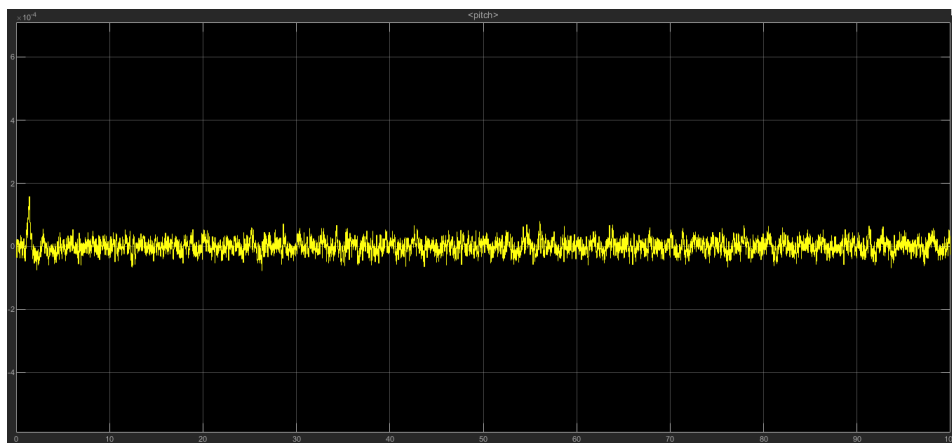


Figura 6.7: Pitch: modello lineare con riferimento costante

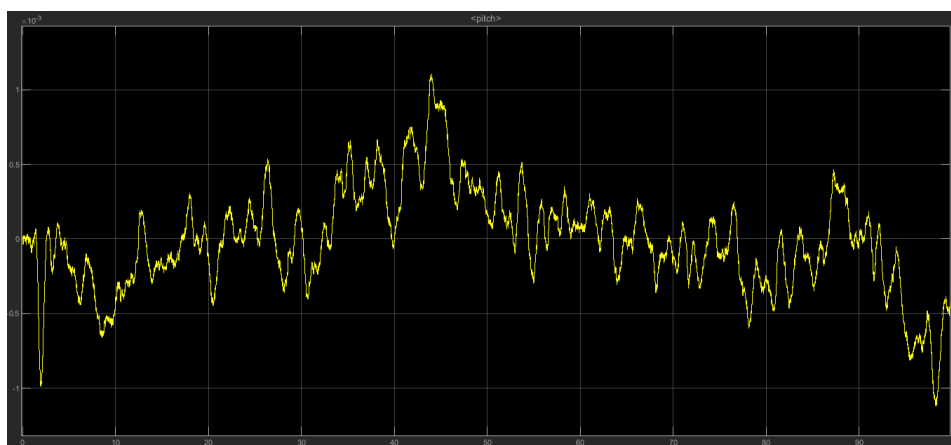


Figura 6.8: Pitch: modello lineare con riferimento variabile

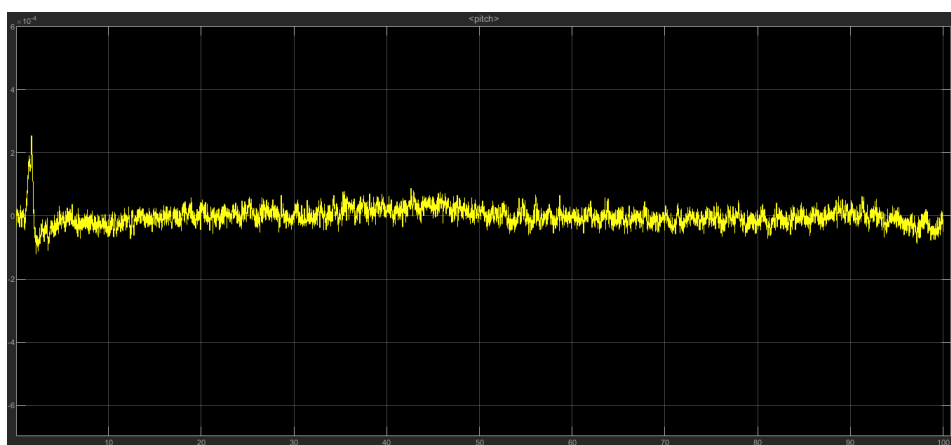


Figura 6.9: Pitch: modello non lineare con riferimento costante

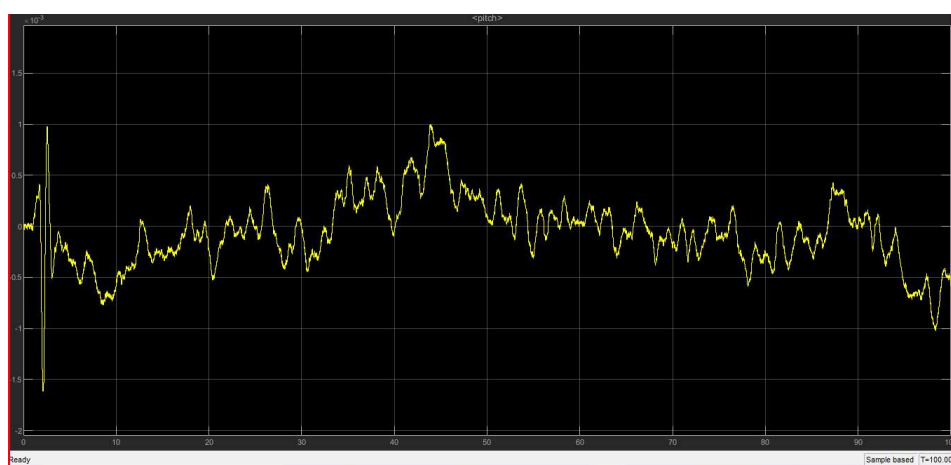


Figura 6.10: Pitch: modello non lineare con riferimento variabile



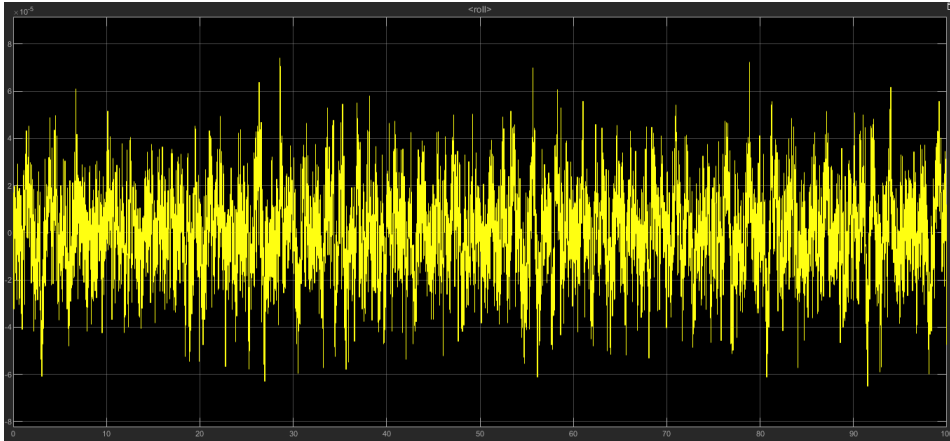


Figura 6.11: Roll: modello lineare con riferimento costante

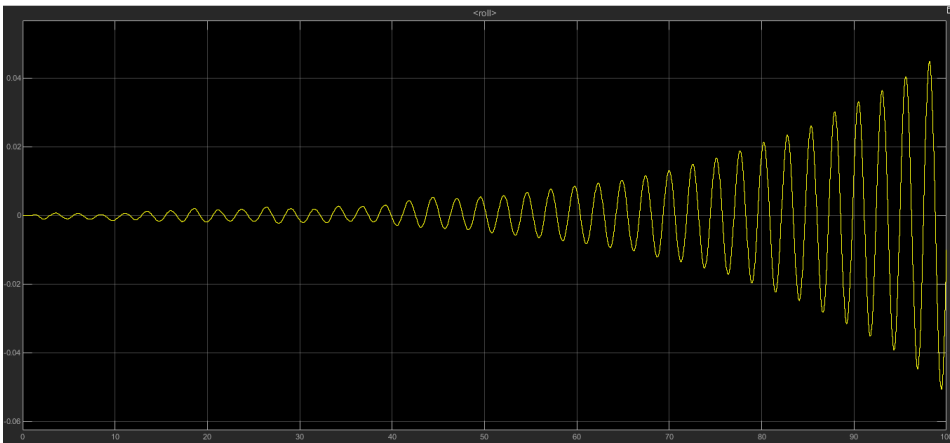


Figura 6.12: Roll: modello lineare con riferimento variabile

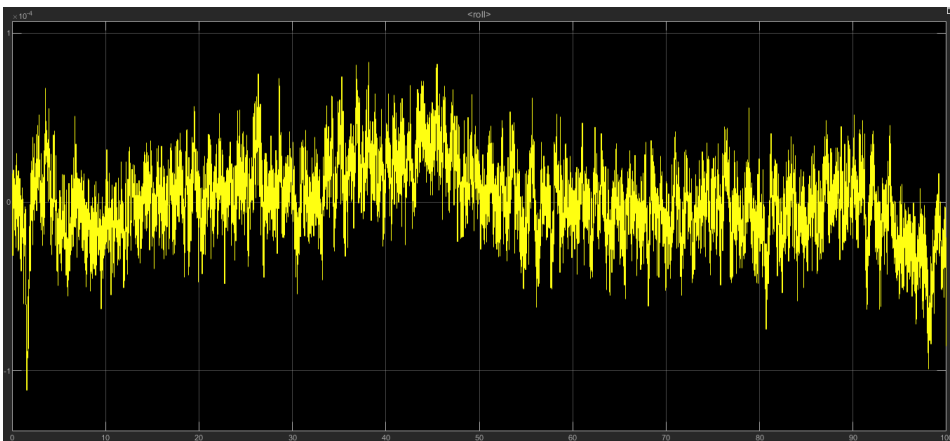


Figura 6.13: Roll: modello non lineare con riferimento costante

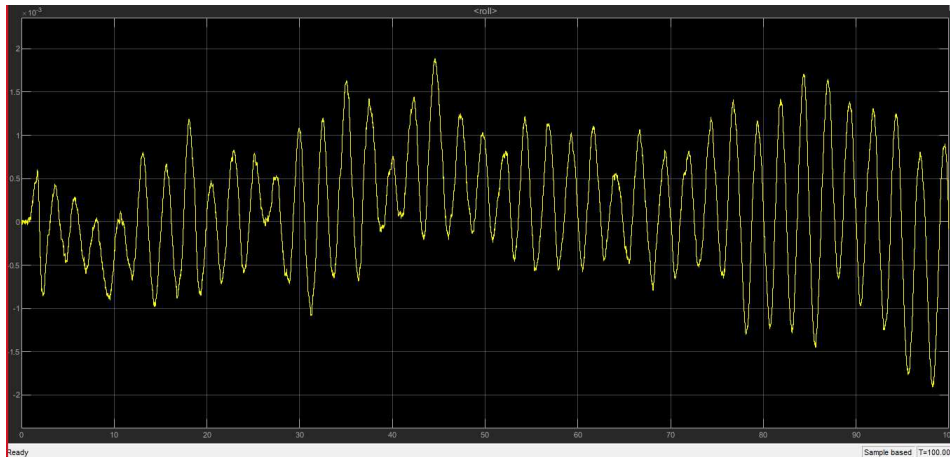


Figura 6.14: Roll: modello non lineare con riferimento variabile

Come si può notare dai vari grafici, sia nel modello lineare che non lineare, le oscillazioni intorno al valore di riferimento, sono molto basse. I modelli lineare e non lineare con riferimento costante presentano oscillazioni dell'ordine di  $10^{-4}$  mentre gli stessi, con riferimento variabile, hanno oscillazioni poco più grandi (dell'ordine di  $10^{-3}$ ). Si può osservare però un'eccezione nell'andamento del roll, nel modello lineare, con riferimento variabile. Infatti qui le oscillazioni crescono all'aumentare del tempo rendendo quindi il sistema instabile.

Prima di passare alla progettazione di nuovi controllori si effettua una trattazione generale delle due tecniche di controllo lineare utilizzate: sintesi in  $\omega$  e luogo delle radici.

## 6.4 Sintesi in $\omega$

Effettuare una sintesi significa scegliere la funzione di trasferimento del controllore  $G(s)$  in modo tale che il sistema in catena chiusa abbia le proprietà che gli sono state assegnate in fase di progetto; ovvero si vuole che l'uscita abbia le caratteristiche desiderate, dette *specifiche*. Ne esistono di due tipi:

- univoche: chiamate così in quanto ottenibili in un unico modo, esse riguardano la struttura del sistema e sono ad esempio: stabilità a ciclo chiuso, tipo del sistema (numero di poli in  $s=0$ ), errore di regime permanente, astatismo rispetto ai disturbi;

- non univoche (lasche): possono essere imposte in vari modi. Tra queste vi è, ad esempio, il margine di fase, la banda passante, la forma di  $F(j\omega)$ ;

Il primo passo è quindi quello di cercare un controllore che soddisfi le specifiche univoche (anche detto controllore di primo tentativo).

Tale controllore avrà la seguente forma:  $\frac{K_g}{s^r}$  dove  $k_g$  indica il guadagno del controllore e  $r$  è il numero di poli nell'origine che ci serve, ad esempio, per garantire l'astatismo ai disturbi in catena diretta, o il tipo del sistema.

In seguito si verifica se tale controllore soddisfa anche le specifiche lasche. Se così non fosse, si progettano funzioni anticipatrici o attenuatrici al fine di costruire un controllore che le soddisfi tutte.

Per poter comprendere fino in fondo tale tecnica di progettazione, si definiscono alcuni parametri fondamentali, sia nel dominio della frequenza che nel dominio del tempo. Inoltre si distinguono anche i parametri in catena chiusa da quelli in catena aperta:

- sovraelongazione  $\hat{s} = \frac{y_{max} - y_{des}}{y_{des}}$ : massimo scostamento della risposta dal valore di regime (parametro definito in catena chiusa, nel dominio del tempo);
- tempo di salita  $t_s$ : tempo che l'uscita impiega per raggiungere per la prima volta il valore di regime. E' quindi una misura della prontezza del sistema (come  $\hat{s}$  anche  $t_s$  è definito in catena chiusa e nel dominio del tempo);
- tempo di assestamento  $t_a$ : tempo che l'uscita impiega per entrare definitivamente nella fascia di assestamento per non uscirne più (definito in catena chiusa, nel dominio del tempo);
- modulo alla risonanza  $M_r = \frac{|W(j\omega)|_{max}}{|W(j0)|}$ : se il modulo di  $W(j0)$  fosse uguale a 1, ovvero 0 dB, il modulo alla risonanza coinciderebbe con il massimo valore assunto dalla funzione di trasferimento in catena chiusa ( $M_r$  è un parametro definito in catena chiusa, nel dominio della frequenza);
- banda passante  $B_3$ : pulsazione in corrispondenza della quale il modulo di  $W(j\omega)$  si attenua di 3 dB rispetto al modulo di  $W(j0)$  (come

$M_r$  anche  $B_3$  è un parametro in catena chiusa, nel dominio della frequenza);

- pulsazione di attraversamento  $\omega_t$ : pulsazione in corrispondenza della quale il diagramma di Bode del modulo di  $F(j\omega)$  interseca l'asse a 0 dB (tale parametro è definito in catena aperta, nel dominio della frequenza);
- margine di fase  $m\varphi = \angle F(j\omega_t) + 180^\circ$  : parametro in catena aperta nel dominio della frequenza. Se il margine di fase è positivo, il sistema è stabile;

Tali indici sono relazionati tra loro nel seguente modo:

$$\frac{1 + \hat{s}}{M_r} \simeq 0.85 \quad (6.2)$$

$$B_3 t_s \simeq 3 \quad (6.3)$$

Come detto in precedenza, se il controllore di primo tentativo non soddisfa le specifiche lasche si devono progettare funzioni compensatrici elementari. Ne esistono due tipi:

- funzione anticipatrice: permette di aumentare il modulo e la fase della funzione di trasferimento in catena aperta;
- funzione ritardatrice o attenuatrice: attenua il modulo e la fase di  $F(j\omega)$ ;

### Rete anticipatrice

$$Ra(s) = \frac{1 + \frac{s}{\omega_a}}{1 + \frac{s}{m_a\omega_a}} = \frac{1 + s\tau_a}{1 + \frac{\tau_a s}{m_a}} \quad (6.4)$$

dove  $\omega_a$  è la pulsazione di rottura dello zero,  $m_a\omega_a$  è la pulsazione di rottura del polo e  $\tau_a$  è l'inverso di  $\omega_a$ . Essendo  $m_a > 1$ ,  $m_a\omega_a > \omega_a$  e quindi si ha aumento di modulo e fase.

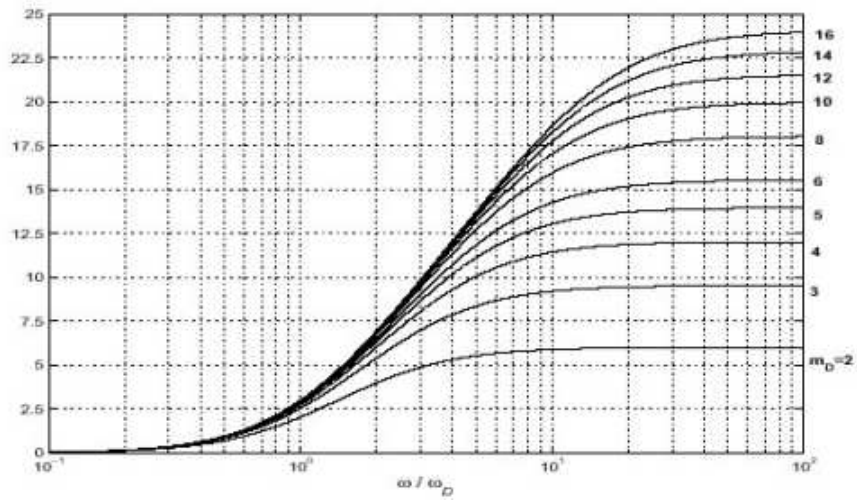


Figura 6.15: Funzione anticipatrice del modulo

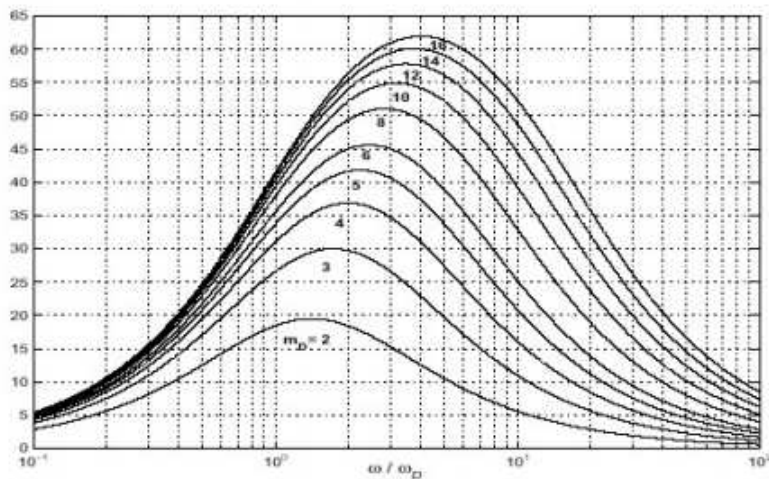


Figura 6.16: Funzione anticipatrice della fase

## Rete attenuatrice

$$Ri(s) = \frac{1 + \frac{s}{m_i\omega_i}}{1 + \frac{s}{\omega_i}} = \frac{1 + \frac{\tau_i s}{m_i}}{1 + \tau_i s} \quad (6.5)$$

dove  $m_i\omega_i$  è la pulsazione di rottura dello zero,  $\omega_i$  è la pulsazione di rottura del polo e  $\tau_i$  è l'inverso di  $\omega_i$ . Essendo  $m_i > 1$ ,  $\omega_i < m_i\omega_i$  e quindi il modulo e la fase vengono attenuati. I grafici sono gli stessi della funzione anticipatrice cambiata di segno.

## 6.5 Sintesi con il luogo delle radici

La sintesi con il luogo delle radici è una sintesi per tentativi così come la sintesi in  $\omega$  ma a differenza di quest'ultima, possiamo applicarla anche a un processo fisico che non è a fase minima. Lo svantaggio è che il modello matematico di  $P(s)$  deve essere necessariamente noto.

Sia la funzione di trasferimento in catena aperta così definita:

$$F(s) = k \frac{\prod_{i=1}^m (s - z_i)}{\prod_{i=1}^n (s - p_i)} \quad (6.6)$$

dove  $m$  è il numero degli zeri,  $n$  il numero dei poli e  $k$  è il coefficiente di guadagno (per ipotesi  $n > m$ ).

Si ottiene così la funzione di trasferimento in catena chiusa:

$$W(s) = \frac{F(s)}{1 + F(s)} = k \frac{\prod_{i=1}^m (s - z_i)}{\prod_{i=1}^n (s - p_i) + k \prod_{i=1}^m (s - z_i)} \quad (6.7)$$

Il luogo delle radici è il luogo dei punti percorsi dalle radici dell'equazione:

$$f(s, k) = \prod_{i=1}^n (s - p_i) + k \prod_{i=1}^m (s - z_i) = 0 \quad (6.8)$$

Per  $k > 0$  si ha luogo positivo, per  $k < 0$  si ha luogo negativo mentre per  $k = 0$  i poli in catena chiusa coincidono con i poli in catena aperta. Altre importanti proprietà sono:

- il luogo delle radici è simmetrico rispetto all'asse reale;
- tutto l'asse reale appartiene al luogo. Una parte appartiene al luogo positivo e l'altra a quello negativo. Appartengono al luogo positivo tutte le porzioni di asse reale che lasciano alla propria destra un numero dispari di poli e zeri di  $F(s)$ , contati con la loro molteplicità;
- i punti singolari sono le radici multiple dell'equazione  $f(s, k)$ .  
Per trovare tali punti basta imporre le seguenti condizioni:

$$\begin{cases} f(s, k) = 0 \\ \frac{\partial f(s, k)}{\partial s} = 0 \end{cases} \quad (6.9)$$

Se si è in presenza di un punto singolare di molteplicità  $\mu$ , le tangenti al luogo, nel punto singolare, dividono l'angolo giro in  $2\mu$  parti uguali.

- Per  $k \rightarrow \infty$   $m$  rami convergono sugli zeri, gli altri  $n-m$  rami divergono all'infinito lungo  $n-m$  semirette (asintoti del luogo) che partono da un punto sull'asse reale:  $S_0$  (centro degli asintoti).

$$S_0 = \frac{\sum_{i=1}^n P_i - \sum_{i=1}^m Z_i}{n - m} \quad (6.10)$$

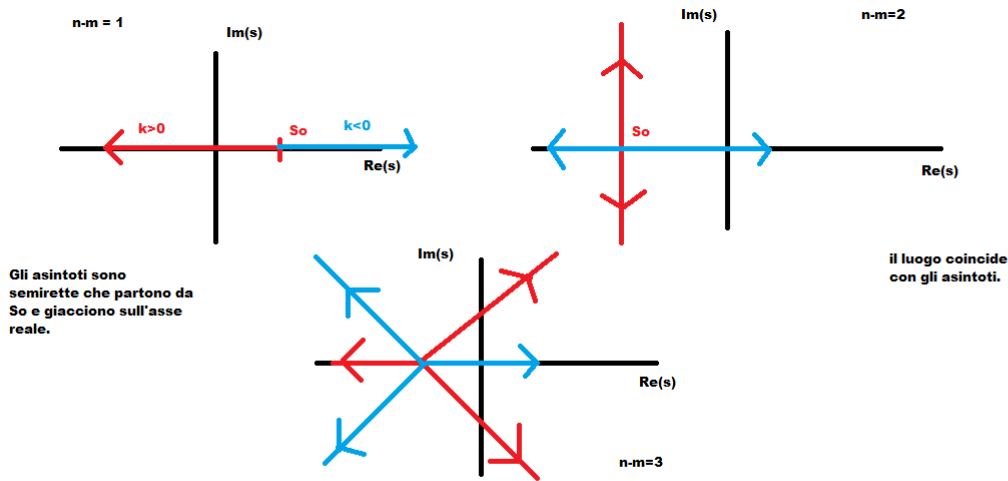


Figura 6.17: Asintoti al variare di  $n-m$

Per trovare l'angolo formato dagli asintoti si utilizzano le seguenti espressioni:

$$\varphi_+ = \frac{(2h + 1)\pi}{n - m} \quad (6.11)$$

$$\varphi_- = \frac{2h\pi}{n - m} \quad (6.12)$$

Per la sintesi con il luogo delle radici si procede allo stesso modo visto per la sintesi in  $\omega$ , ovvero si progetta un controllore di primo tentativo che serve a garantire le specifiche univoche:  $G(\hat{s}) = \frac{K_g}{s^r}$ . Si controlla se tale controllore soddisfa anche le specifiche lasche. In caso contrario si progetta una funzione  $R(s)$ , formata da coppie zero-polo, ottenendo così un nuovo controllore che soddisfa tutte le specifiche:  $G(s) = \frac{K_g}{s^r} R(s)$ . Non si possono cancellare poli o zeri nella regione non di specifica. In

tal caso, infatti, il sistema complessivo in spazio di stato non è completamente raggiungibile o osservabile. Dal punto di vista pratico, non si riuscirà mai a cancellare uno zero mettendo un polo esattamente nella stessa posizione.

Inoltre, il grado del denominatore di  $G(s)$  deve essere maggiore o uguale del grado del numeratore di  $G(s)$  (non si possono avere più zeri che poli altrimenti il sistema non è fisicamente realizzabile). E' conveniente ottenere un  $n-m$  più piccolo possibile.

Indicando con  $m_G$  e  $n_G$  rispettivamente il numero di zeri e poli di  $G(s)$  e con  $m_P$  e  $n_P$  il numero di zeri e poli di  $P(s)$ , si può scrivere  $n-m$  come :  $n_G + n_P - m_G - m_P$ . Essendo  $n_G \geq m_G$  la situazione più favorevole è per  $n_G = m_G$ . Quindi per  $n-m = n_P - m_P$ .

Passiamo ora alla progettazione dei controllori per i due gradi di libertà del pitch e del roll.

## 6.6 Controllori Pitch

### 6.6.1 Sintesi in $\omega$

Funzione di trasferimento del Pitch:

$$P(s) = \frac{2717}{s^2 + 1.619s} \quad (6.13)$$

Specifiche da soddisfare:

- sistema di tipo 1;
- errore di regime permanente  $< 0.36$ ;
- tempo di salita di circa un secondo;
- $M_r < 2\text{dB}$ ;



Traduzione delle specifiche:

$$\begin{aligned}
 [B_3]_{rad/s} &\simeq \frac{3}{t_s} \simeq 3rad/s \\
 [B_3]_{Hz} &\simeq \frac{3}{2\pi} \simeq 0.4777Hz \\
 \omega_t &\simeq [3 : 5][B_3]_{Hz} \simeq 1.95rad/s \\
 \tilde{e} &= \frac{K_d^2}{K_f} = \frac{1}{K_g K_p} = 0.36 \\
 K_g &= \frac{1}{0.36 \cdot 1678.196} \simeq 0.001665 \\
 Mr < 2dB &\rightarrow m_\varphi > 47^\circ
 \end{aligned} \tag{6.14}$$

Controllore di primo tentativo:  $\hat{G}(s) = 0.001665$

Quindi  $\hat{F}(s) = \hat{G}(s) \cdot P(s)$

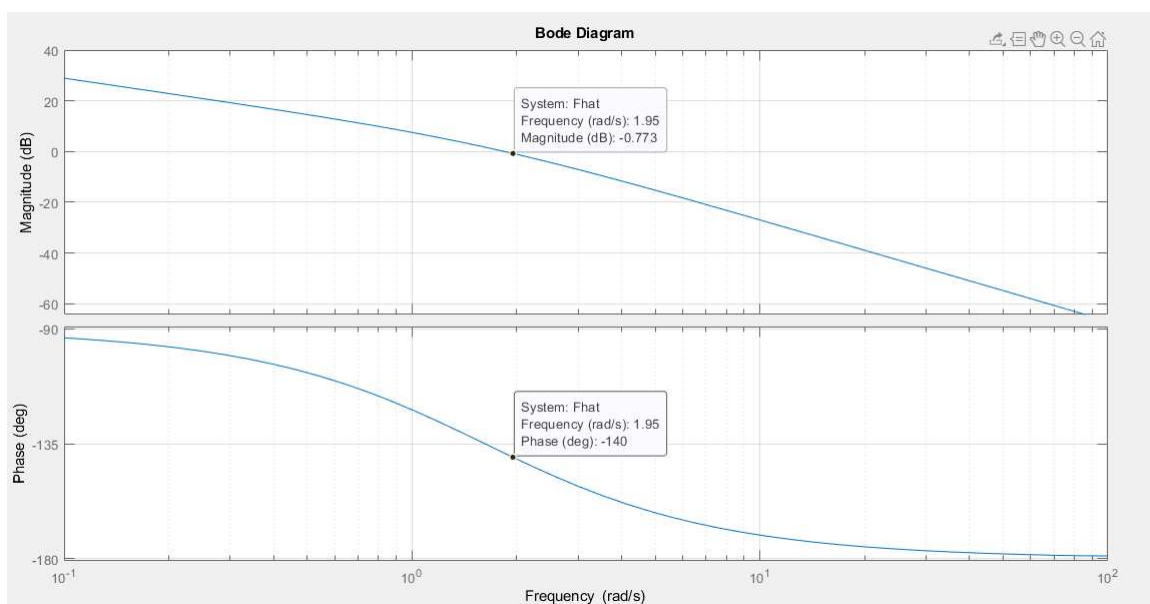


Figura 6.18: Diagrammi di Bode di  $\hat{F}(s)$

Come si può notare dai diagrammi di Bode:

- $|\hat{F}(1.95j)| = -0.8dB$
- $\angle\hat{F}(1.95j) = -140^\circ \rightarrow m_\varphi = 40^\circ$

Bisogna quindi progettare una funzione anticipatrice che produca un aumento di modulo e fase. Si è scelto  $\omega\tau_a = 0.45$  e  $m_a=7$ .

Si ottiene così:

$$Ra(s) = \frac{1 + \frac{s}{4.33}}{1 + \frac{s}{30.3}} \quad (6.15)$$

Il nuovo controllore avrà quindi la seguente forma:

$$G(s) = 0.001665 \frac{1 + \frac{s}{4.33}}{1 + \frac{s}{30.3}} = 0.011651 \frac{(s + 4.33)}{(s + 30.3)} \quad (6.16)$$

$$F(s) = G(s) \cdot P(s)$$

Attraverso il toolbox di MATLAB *sisotool* è possibile visualizzare i diagrammi di Bode di  $F(s)$ , il luogo delle radici e la risposta al gradino.

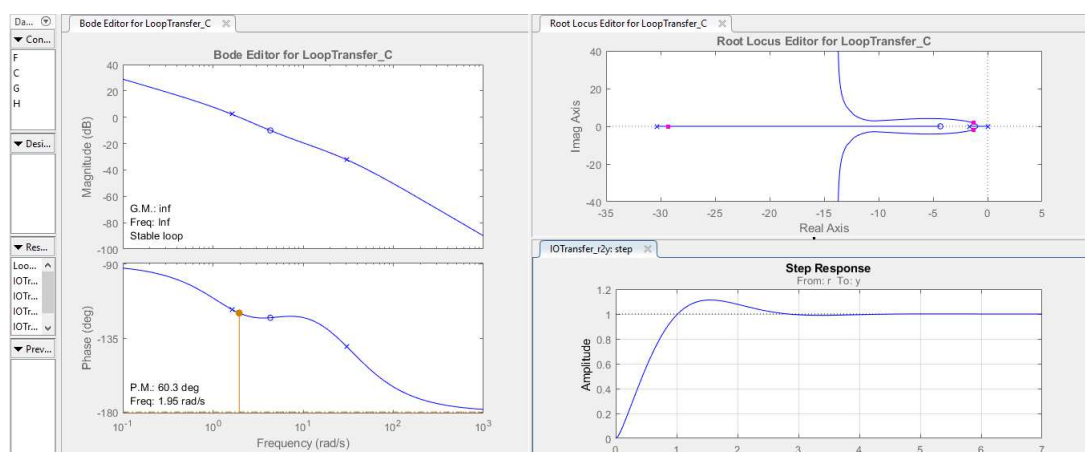


Figura 6.19: Diagrammi di  $F(s)$

Si può notare come il margine di fase sia superiore a  $47^\circ$  e il tempo di salita nella risposta a gradino sia di circa 1 secondo.

Utilizzando il comando `c2d(Gs,0.005,'zoh')` di MATLAB si ottiene lo stesso controllore a tempo discreto:

$$G(z) = 0.0011651 \frac{(z - 0.9799)}{(z - 0.8594)} \quad (6.17)$$

Tale controllore viene quindi inserito all'interno dell'Attitude Controller nel Flight Control System.

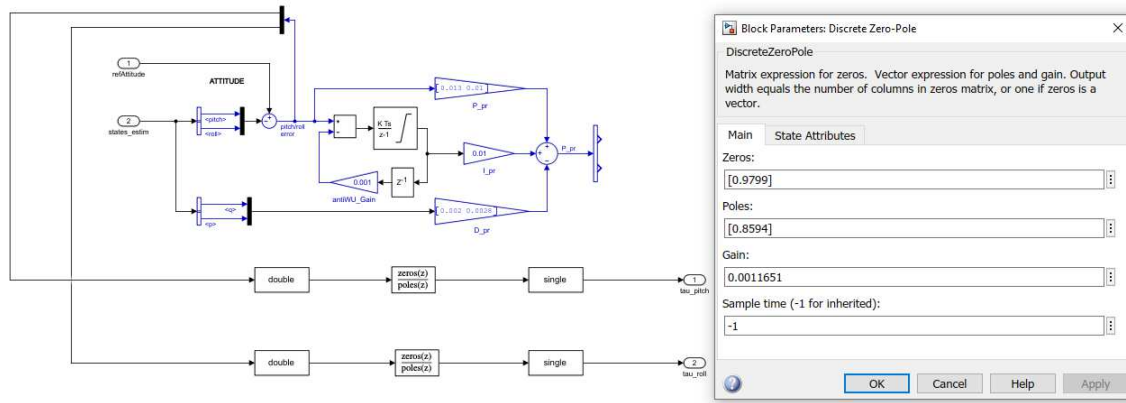


Figura 6.20: Pitch sintesi in  $\omega$

Si può quindi verificare l'andamento della variabile di Pitch attraverso gli scope presenti nel blocco Logging.

Come si può notare dai grafici sottostanti, sia nel modello lineare che non lineare, le oscillazioni intorno al valore di riferimento, si mantengono molto basse (dell'ordine di  $10^{-3}$ ).

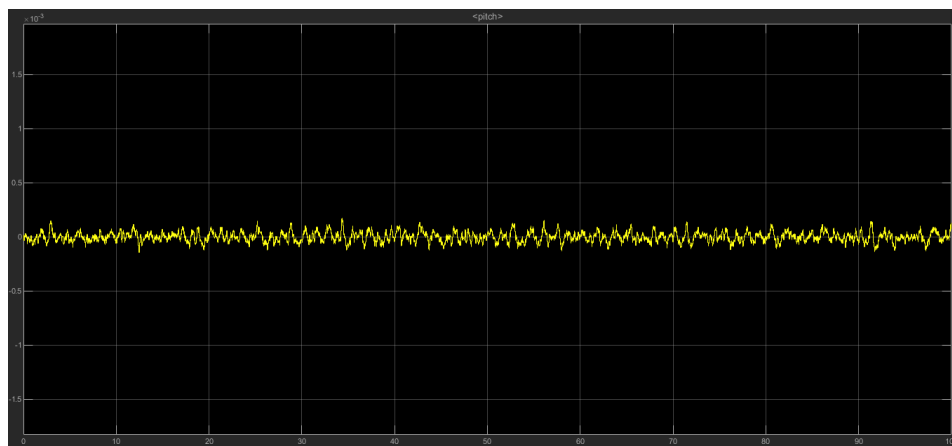


Figura 6.21: Pitch: modello lineare, riferimento costante

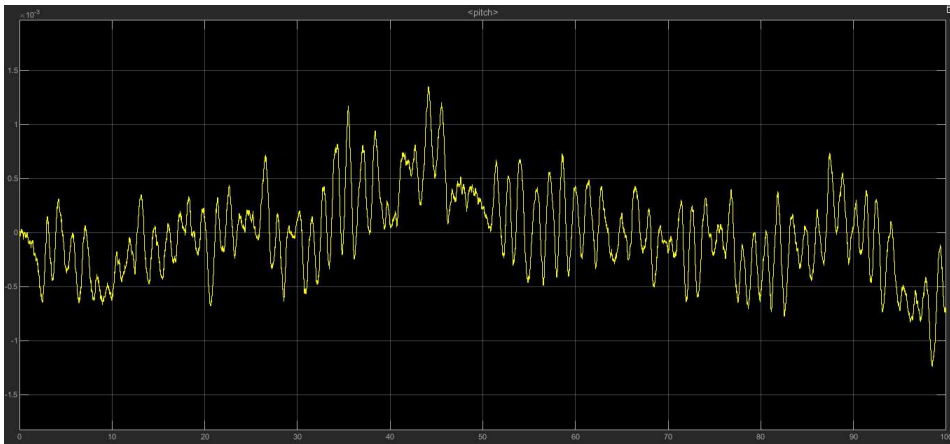


Figura 6.22: Pitch: modello lineare, riferimento variabile

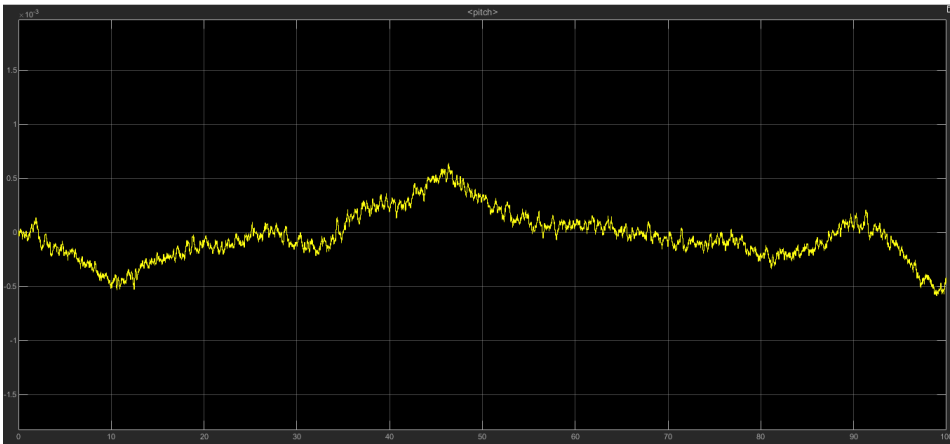


Figura 6.23: Pitch: modello non lineare, riferimento costante

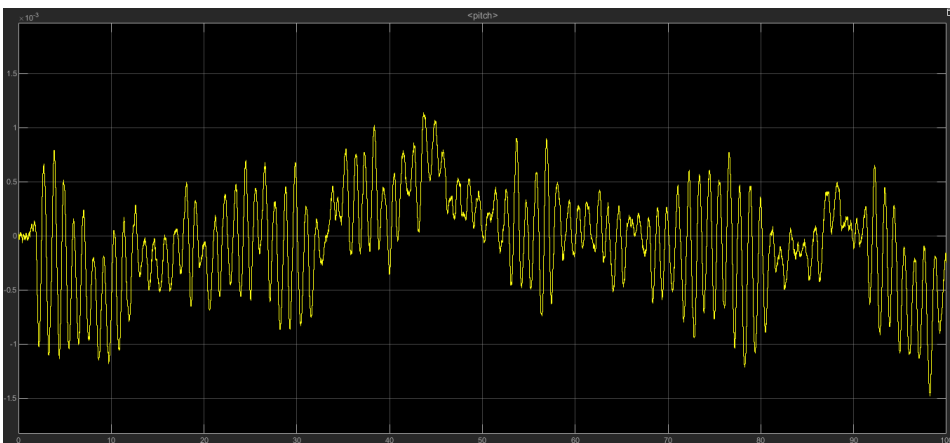


Figura 6.24: Pitch: modello non lineare, riferimento variabile

## 6.6.2 Luogo delle radici

Per progettare un controllore con il luogo delle radici è possibile utilizzare il tool *Control System Designer* a cui è possibile accedere digitando sulla Command Window di MATLAB il comando: `sisotool(Ps)`, dove `Ps` è la funzione di trasferimento del processo.

```
>> Ps = tf([2717],[1 1.6192 0])  
  
Ps =  
  
      2717  
-----  
s^2 + 1.619 s  
  
Continuous-time transfer function.  
  
>> sisotool(Ps)
```

Figura 6.25: Funzione di trasferimento del processo

Come è possibile vedere da tale funzione di trasferimento, il sistema è già a fase minima. Un sistema si dice *a fase minima* se ha tutti gli zeri nel semipiano sinistro del piano complesso (nel caso preso in considerazione si hanno solo due poli e nessuno zero). Si ha quindi la certezza che  $m$  rami del luogo positivo, tenderanno a spostarsi nel semipiano sinistro del piano complesso per  $K \rightarrow \infty$ .

E' possibile notare all'interno del tool i diagrammi di Bode di `Ps`, la risposta a gradino, i diagrammi di Nichols e di Nyquist e il luogo delle radici su cui si focalizza ora maggiore attenzione. E' quindi possibile aggiungere a mano poli e zeri e vedere, in tempo reale, come variano i diagrammi di Bode e la risposta al gradino.

Si è, in questo modo, arrivati al seguente controllore:

$$G(s) = 0.022242 \frac{(s + 4.611)}{(s + 49.18)} \quad (6.18)$$

Si ottiene la funzione di trasferimento in catena aperta:  $F(s) = G(s) \cdot P(s)$ , mentre a tempo discreto si avrà:

$$G(z) = 0.022242 \frac{(z - 0.9796)}{(z - 0.782)} \quad (6.19)$$

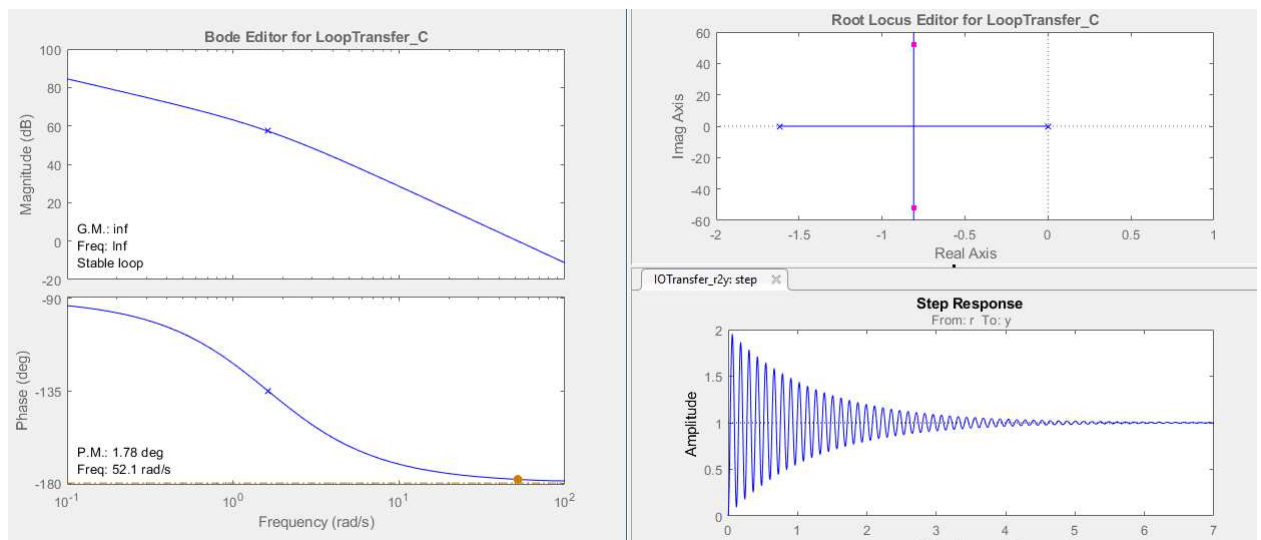


Figura 6.26: Pitch:  $\text{sisotool}(P(s))$

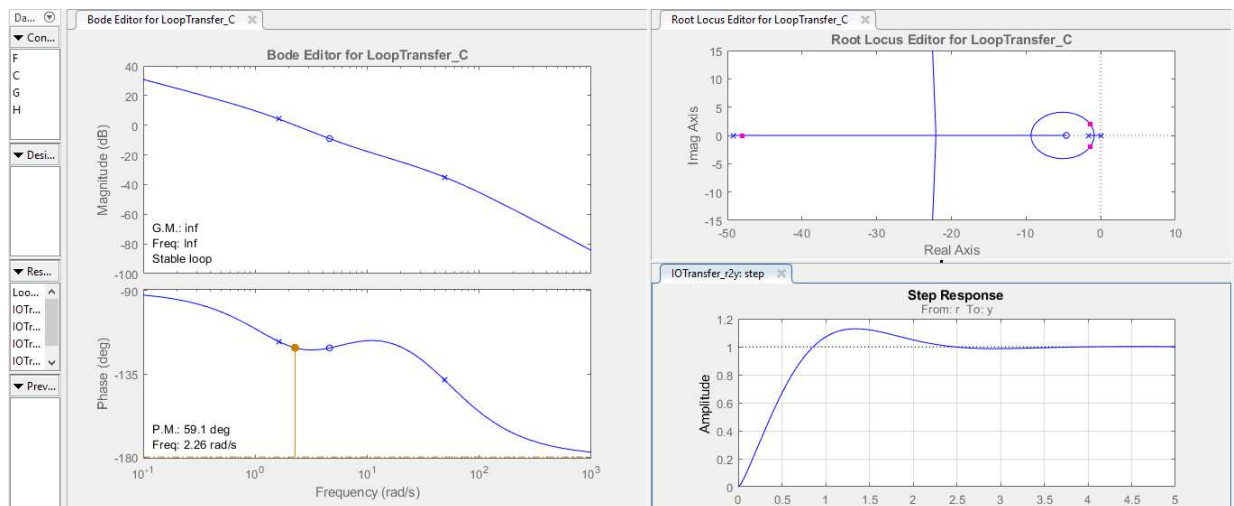


Figura 6.27: Pitch:  $\text{sisotool}(F(s))$

Dopo aver inserito tali dati all'interno di Simulink si otterranno i diagrammi relativi alla dinamica del pitch. Anche in questo caso si noteranno oscillazioni molto basse.

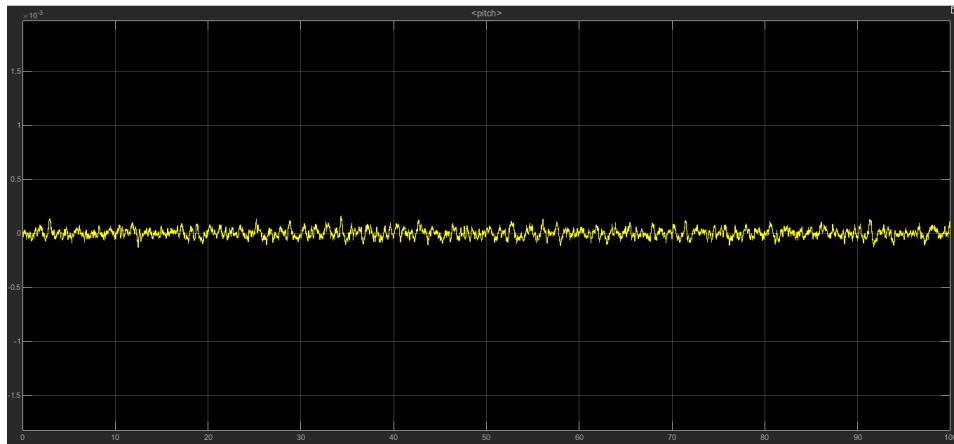


Figura 6.28: Pitch: modello lineare riferimento costante

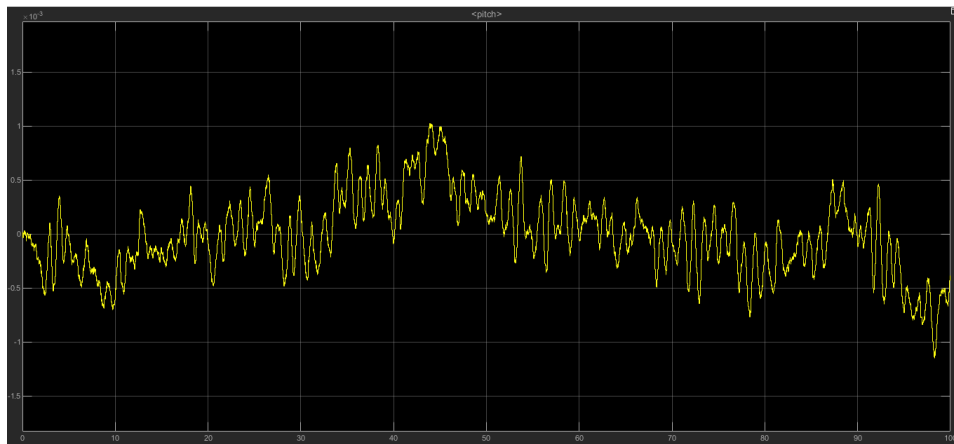


Figura 6.29: Pitch: modello lineare riferimento variabile

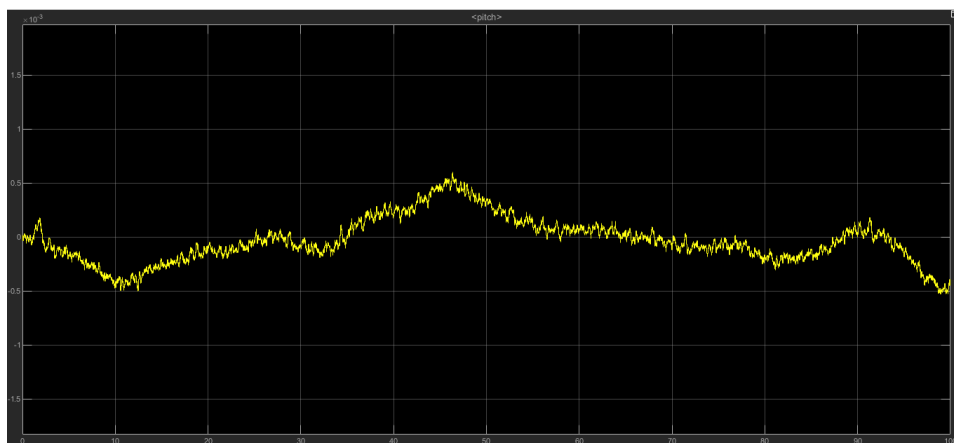


Figura 6.30: Pitch: modello non lineare riferimento costante

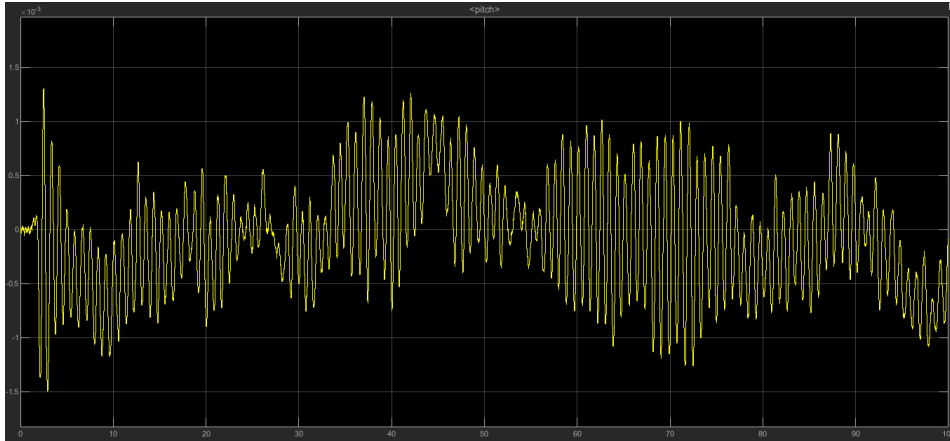


Figura 6.31: Pitch: modello non lineare riferimento variabile

## 6.7 Prestazioni Pitch

Per valutare in maniera oggettiva le prestazioni del sistema vengono utilizzati tre indici:

- IAE (Integral Absolute Error) : integra nel tempo l'errore in valore assoluto

$$IAE = \int_0^T |e(t)| dt \quad (6.20)$$

- ISE (Integral Square Error) : integra nel tempo il quadrato dell'errore

$$ISE = \int_0^T e(t)^2 dt \quad (6.21)$$

- ITAE (Integral Time Absolute Error): integra il valore assoluto dell'errore ma a differenza dell'IAE il tempo funge da peso

$$ITAE = \int_0^T t |e(t)| dt \quad (6.22)$$

Inoltre viene riportato il tempo di simulazione impiegato dal drone a percorrere una traiettoria presente all'interno di un progetto di *parrot-MinidroneCompetition*. Come è possibile notare dalle immagini, entrambi i controllori presentano degli indici di prestazione con valori più bassi rispetto al PID. Per quanto riguarda il tempo di simulazione, il controllore sviluppato con il luogo delle radici migliora il PID di default di circa un secondo mentre quello progettato con la sintesi in frequenza lo peggiora di poco. Viene inoltre riportato il grafico dell'errore che, come si può osservare, si mantiene nell'intorno dello zero.



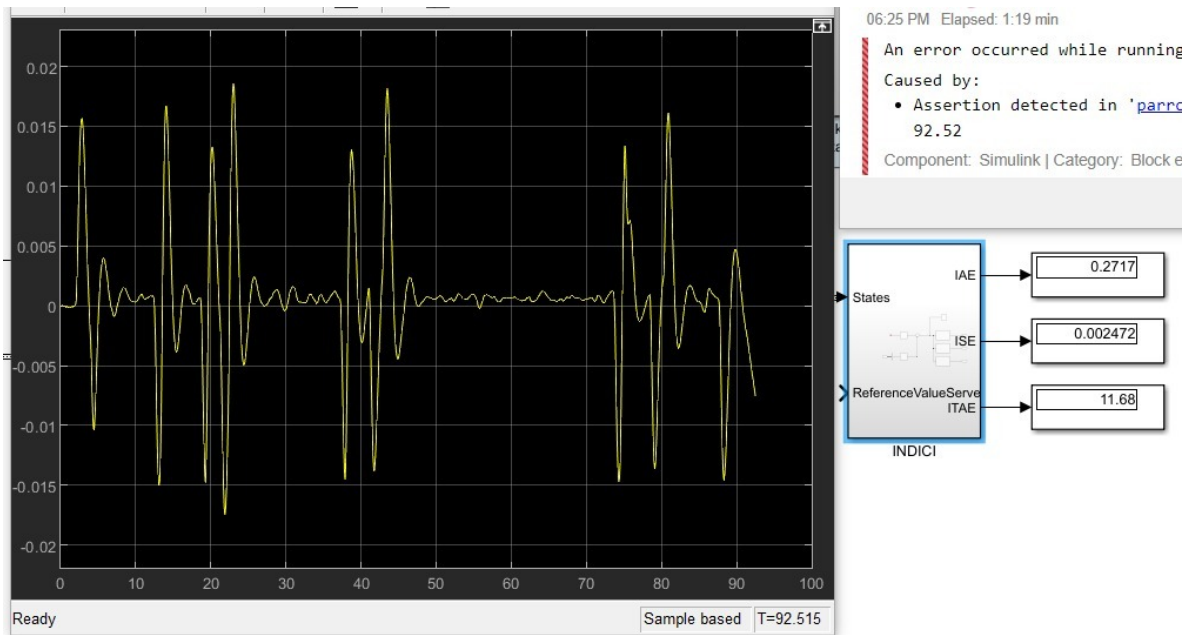


Figura 6.32: Prestazioni pitch con controllore PID

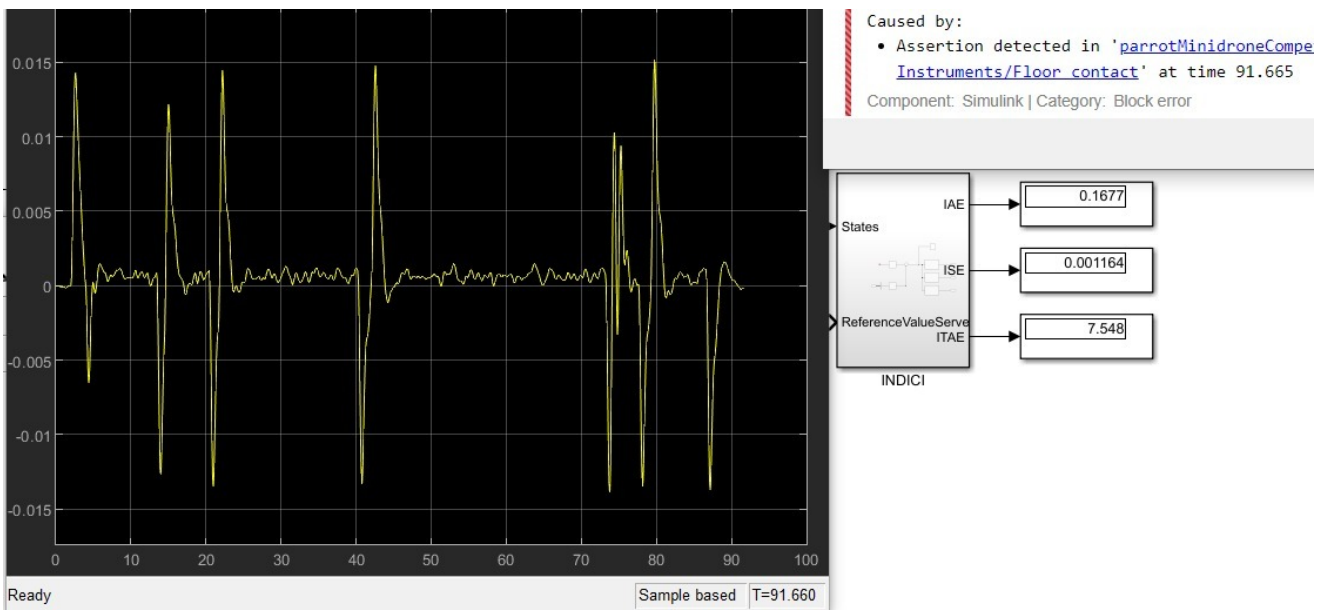


Figura 6.33: Prestazioni pitch con controllore sviluppato con il luogo delle radici

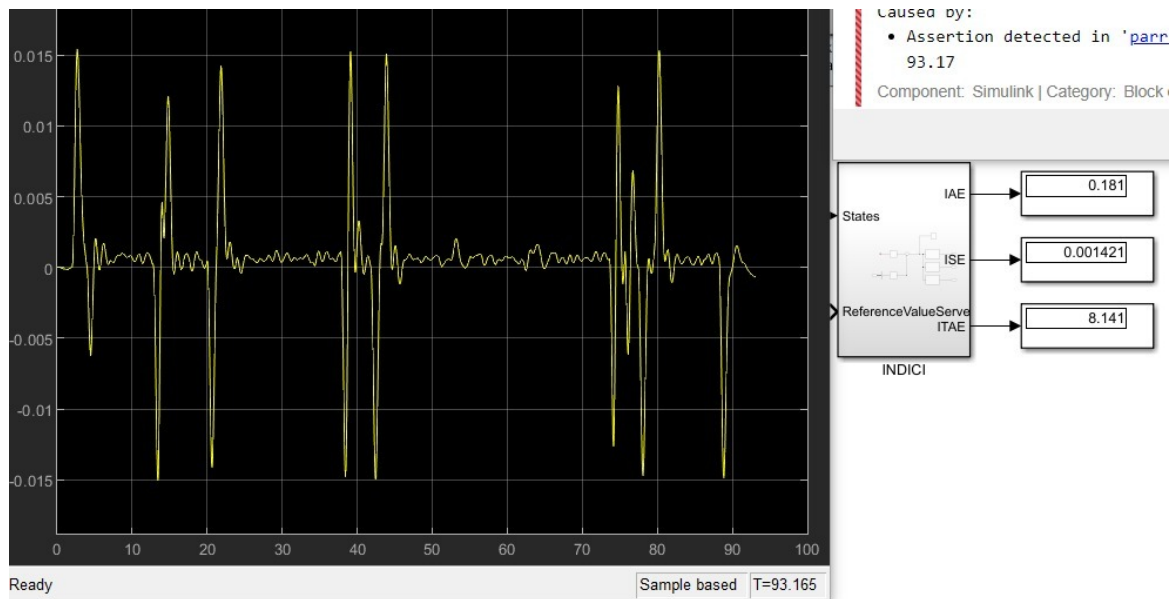


Figura 6.34: Prestazioni pitch con controllore sviluppato con la sintesi in  $\omega$

## 6.8 Controllori Roll

### 6.8.1 sintesi in $\omega$

Funzione di trasferimento del Roll:

$$P(s) = \frac{3644}{s^2 + 2.171s} \quad (6.23)$$

Specifiche da soddisfare:

- sistema di tipo 1;
- $t_s \simeq 0.9s$ ;
- $\tilde{e} \leq 0.32$ ;
- $M_r \leq 2\text{dB}$ ;

Traduzione specifiche:

$$\begin{aligned}
 [B_3]_{rad/s} &\simeq \frac{3}{t_s} \simeq 3.33 rad/s \\
 [B_3]_{Hz} &\simeq \frac{3.33}{2\pi} \simeq 0.5308 Hz \\
 \omega_t &\simeq [3 : 5][B_3]_{Hz} \simeq 2.3 rad/s \\
 \tilde{e} &= \frac{K_d^2}{K_f} = \frac{1}{K_g K_p} = 0.32 \\
 K_g &= \frac{1}{0.32 \cdot 1678.489} \simeq 0.0018618 \\
 Mr < 2dB &\rightarrow m_\varphi > 47^\circ
 \end{aligned} \tag{6.24}$$

Controllore di primo tentativo  $\hat{G}(s) = 0.0018618$

Quindi:  $\hat{F}(s) = 0.0018618 \cdot P(s)$

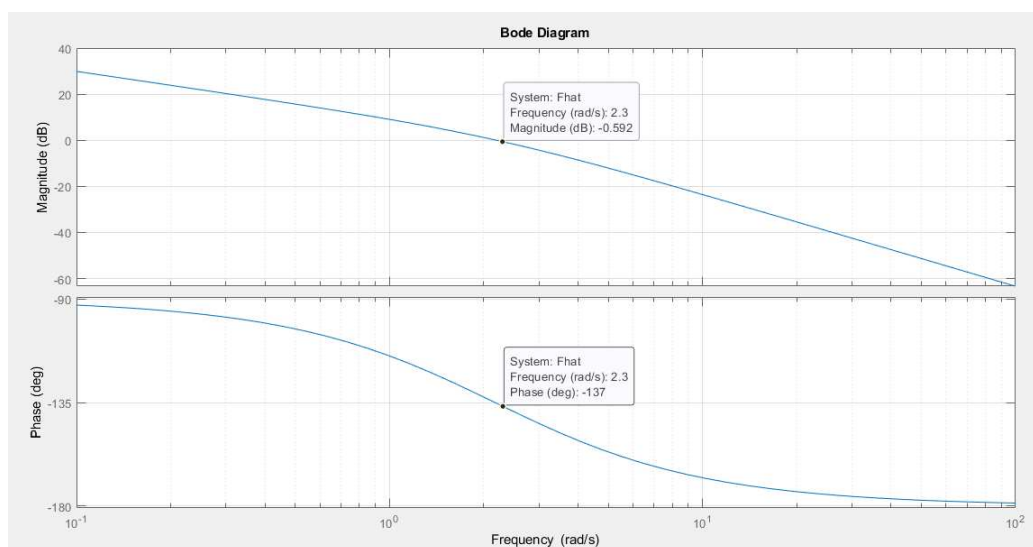


Figura 6.35: Diagrammi di Bode di  $\hat{F}(s)$

Come si può notare dai diagrammi di Bode:

- $|\hat{F}(2.3j)| = -0.592 \text{ dB};$
- $\angle F(2.3j) = -137^\circ \rightarrow m_\varphi = 43^\circ$

Bisogna quindi progettare una funzione anticipatrice. Si sceglie  $\omega\tau_a = 0.45$  (dove  $\omega$  rappresenta la pulsazione di attraversamento) e  $m_a = 10$ .

Si ottiene così:

$$Ra(s) = \frac{(1 + \frac{s}{5})}{(1 + \frac{s}{50})} \quad (6.25)$$

Il controllore risultante sarà quindi:

$$G(s) = 0.0018618 \frac{(1 + \frac{s}{5})}{(1 + \frac{s}{50})} \quad (6.26)$$

$$G(s) = 0.018618 \frac{(s + 5)}{(s + 50)}$$

Si ottiene così:  $F(s) = G(s) \cdot P(s)$

Digitando sulla Command Window di MATLAB il comando  $sisotool(F(s))$ , si ottengono i seguenti diagrammi:

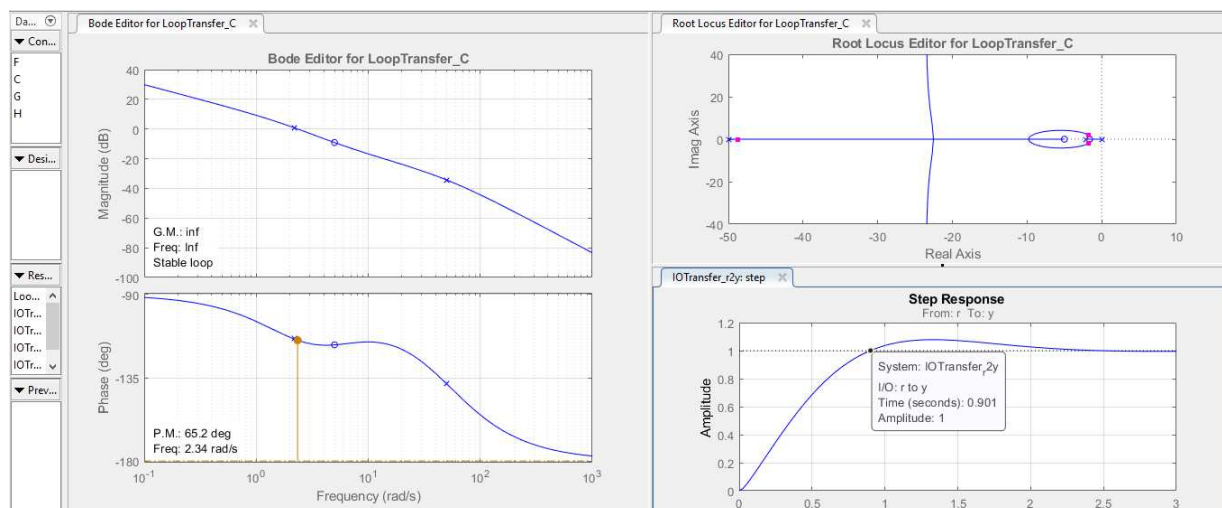


Figura 6.36: Roll:  $sisotool(F(s))$

Si può notare come il tempo di salita rispetti la specifica richiesta ( $t_s \simeq 0.9$ ), così come il margine di fase che infatti è di  $65.2^\circ$ .

Di seguito, si ritrovano i grafici dell'andamento del roll con riferimento costante e variabile, sia nel modello lineare che non lineare. Le oscillazioni sono ancora una volta dell'ordine di  $10^{-3}$  (ciò significa che il drone rimane in posizione di hovering con oscillazioni poco visibili). Inoltre si può notare come nel modello lineare, con riferimento variabile, le oscillazioni non crescono all'aumentare del tempo (come invece accadeva per il PID che raggiungeva quindi un punto di instabilità).

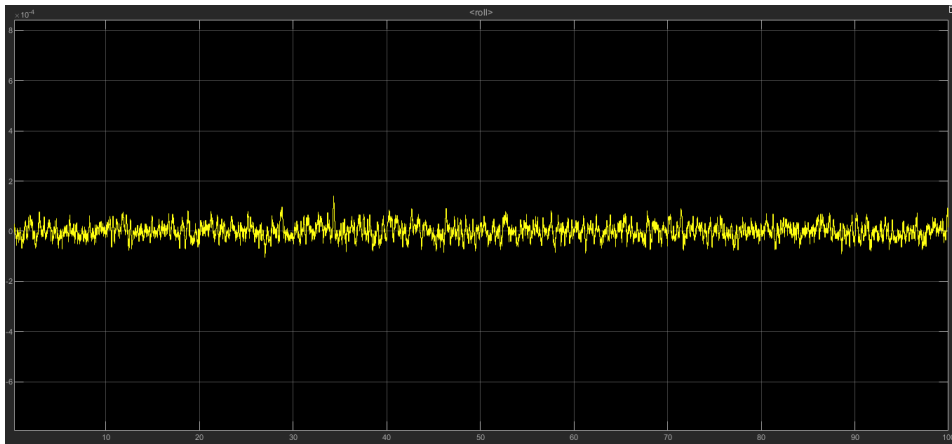


Figura 6.37: Roll: modello lineare, riferimento costante

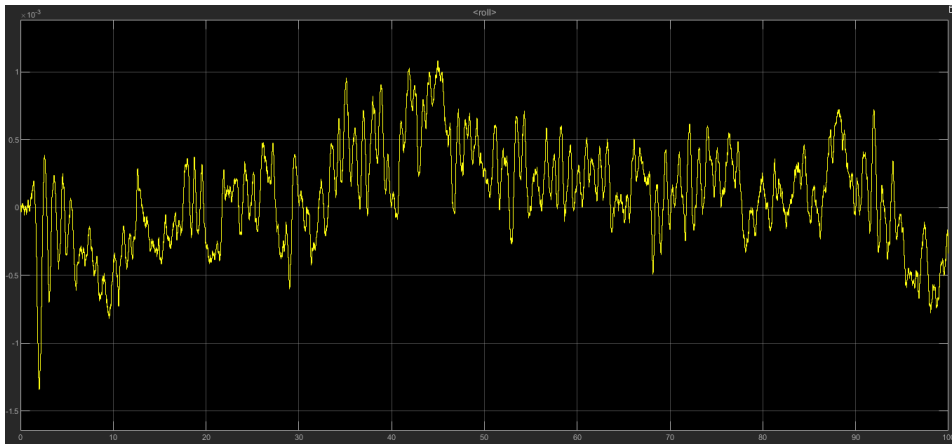


Figura 6.38: Roll: modello lineare, riferimento variabile

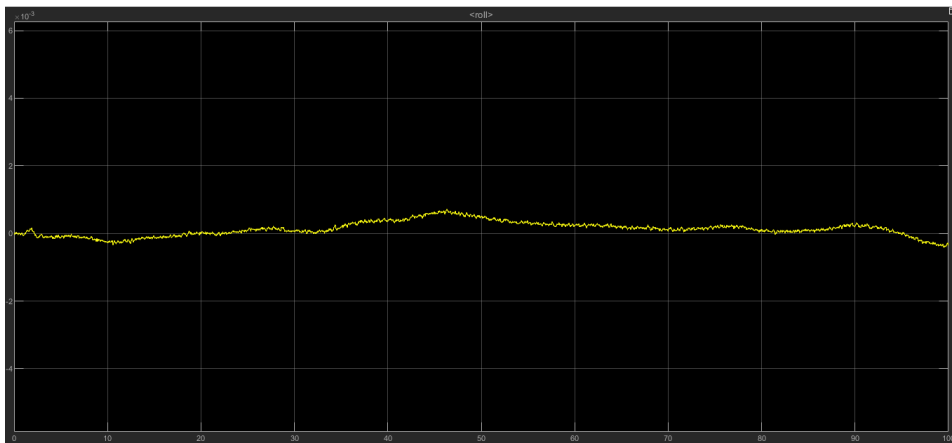


Figura 6.39: Roll: modello non lineare, riferimento costante

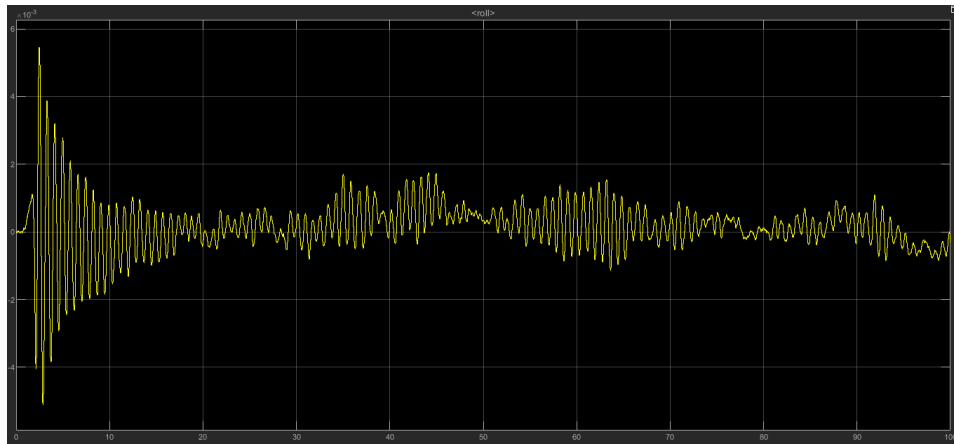


Figura 6.40: Roll: modello non lineare, riferimento variabile

## 6.8.2 luogo delle radici

Dopo aver scritto la funzione di trasferimento del processo  $P(s)$  su MATLAB, attraverso il comando  $sisotool(P(s))$  è possibile visualizzare il luogo delle radici di  $P(s)$  e progettare in tempo reale un controllore che permetta al drone di mantenere la posizione di hovering. Anche in questo caso si tratta di un sistema a fase minima.

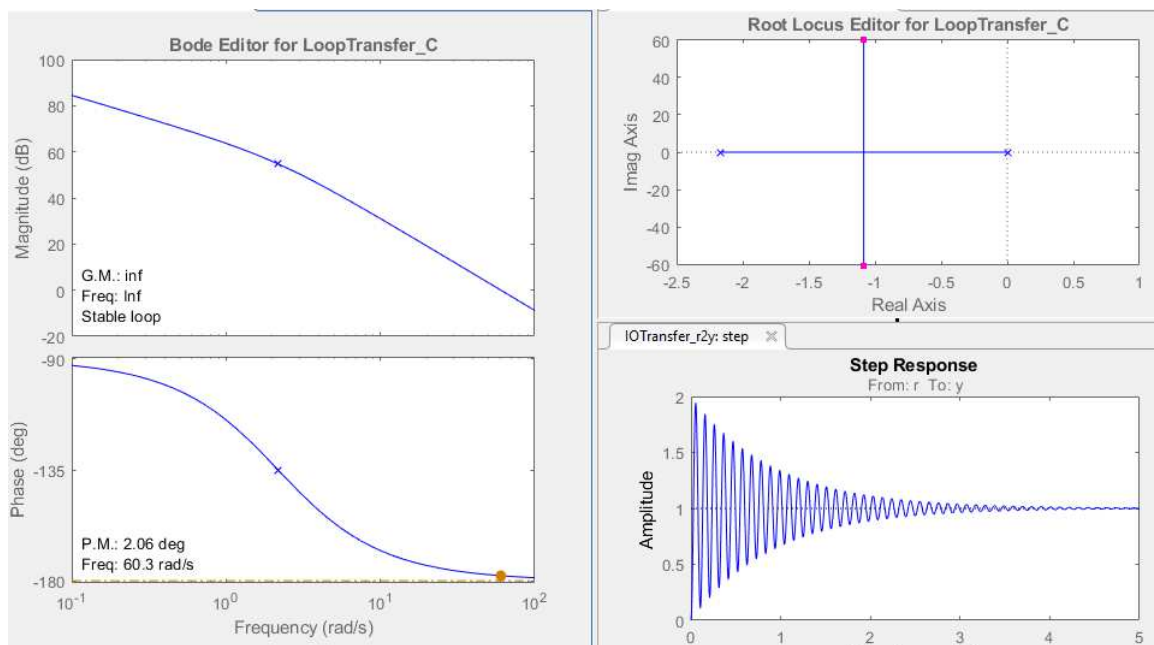


Figura 6.41: Roll:  $sisotool(P(s))$

Il controllore progettato per il roll ha la seguente forma:

$$G(s) = 0.016779 \frac{(s + 5.125)}{(s + 46.61)} \quad (6.27)$$

Quindi:  $F(s) = G(s) \cdot P(s)$

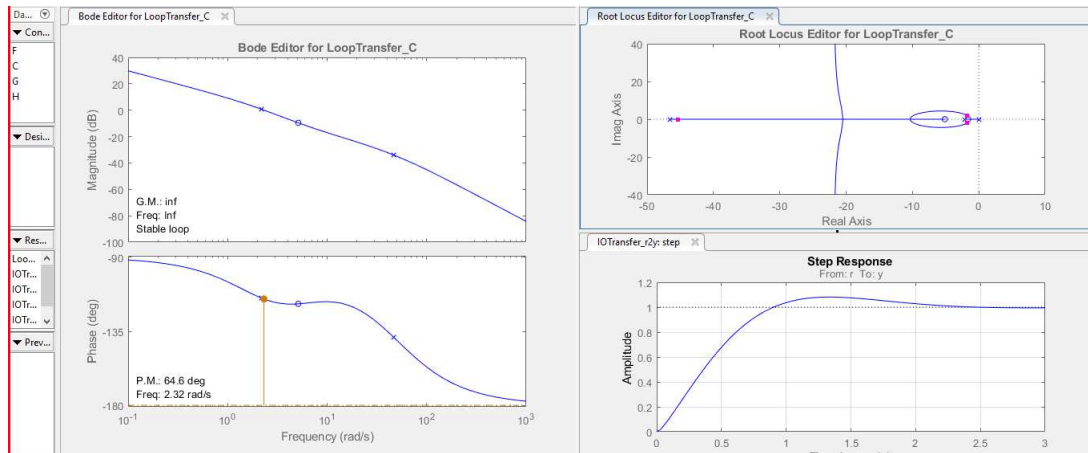


Figura 6.42: Roll: sisotool(F(s))

Dopo aver digitato sulla Command Window di MATLAB il comando  $c2d(G(s), 0.005, 'zoh')$  si otterrà il controllore a tempo discreto:

$$G(z) = 0.016779 \frac{(z - 0.9771)}{(z - 0.7921)} \quad (6.28)$$

Di seguito vengono riportati i grafici relativi all'andamento del roll, controllato con la tecnica del luogo delle radici.

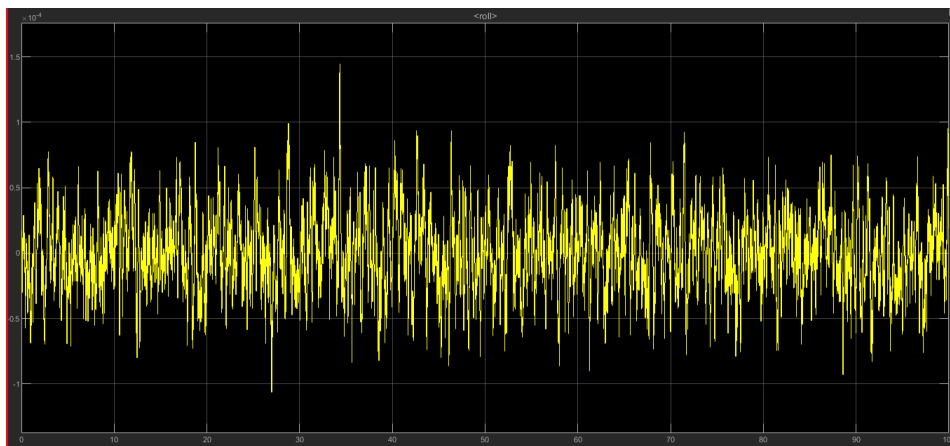


Figura 6.43: Roll: modello lineare riferimento costante

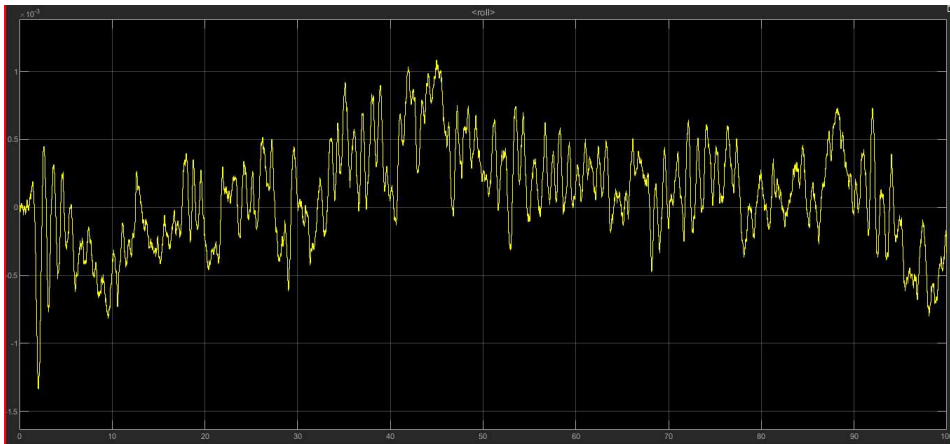


Figura 6.44: Roll: modello lineare riferimento variabile

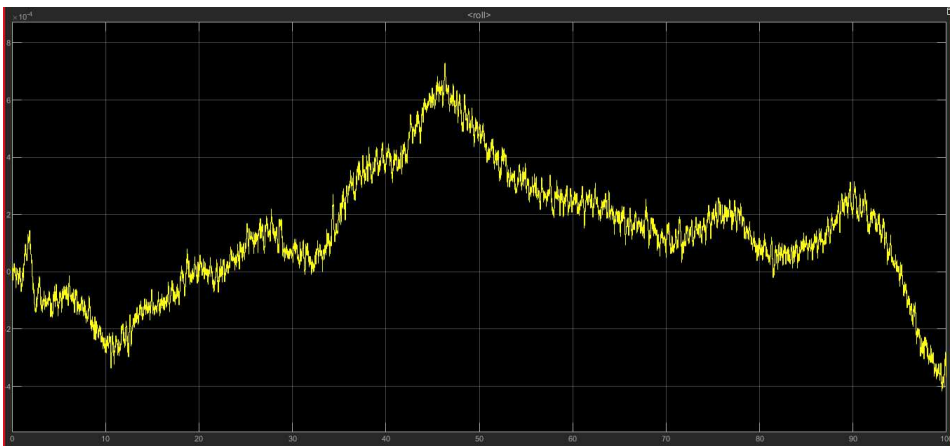


Figura 6.45: Roll: modello non lineare riferimento costante

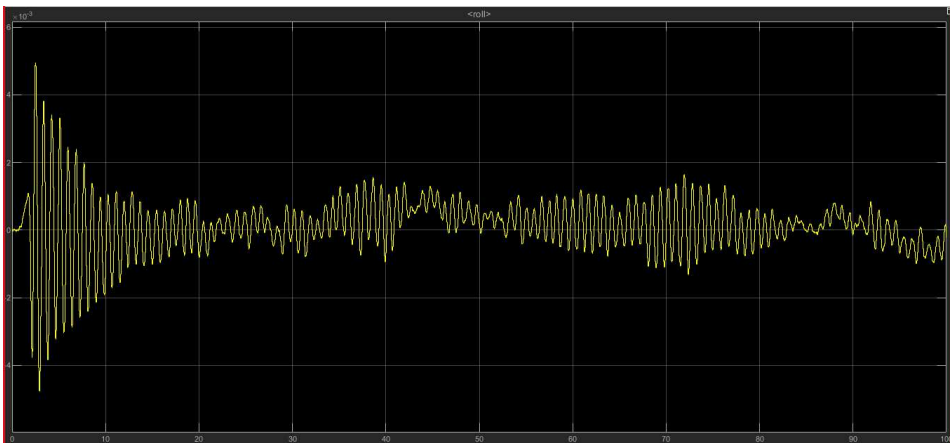


Figura 6.46: Roll: modello non lineare riferimento variabile



## 6.9 Prestazioni roll

Per valutare le prestazioni dei controllori appena creati si utilizzano gli indici visti in precedenza. Anche in questo caso, entrambi i controllori progettati migliorano i 3 indici e il tempo di simulazione.

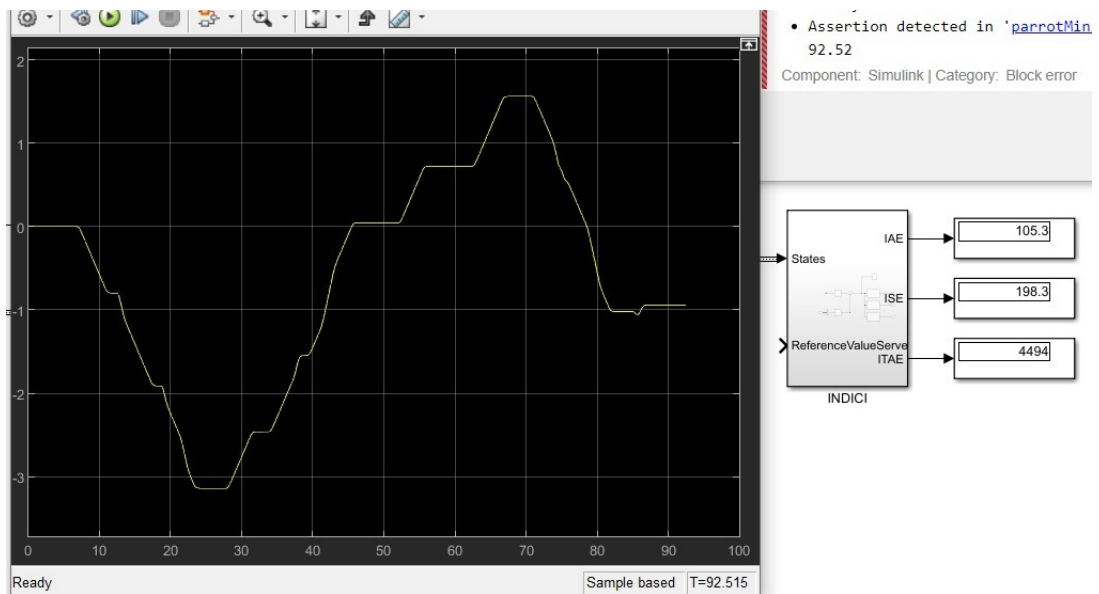


Figura 6.47: Prestazioni roll con controllore PID

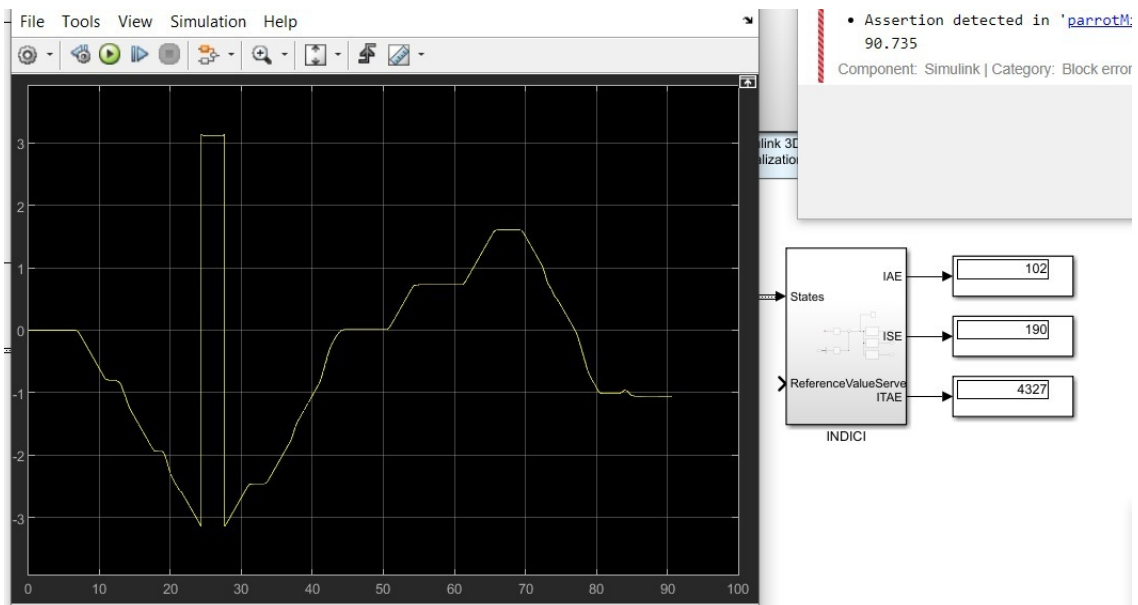


Figura 6.48: Prestazioni roll con il luogo delle radici

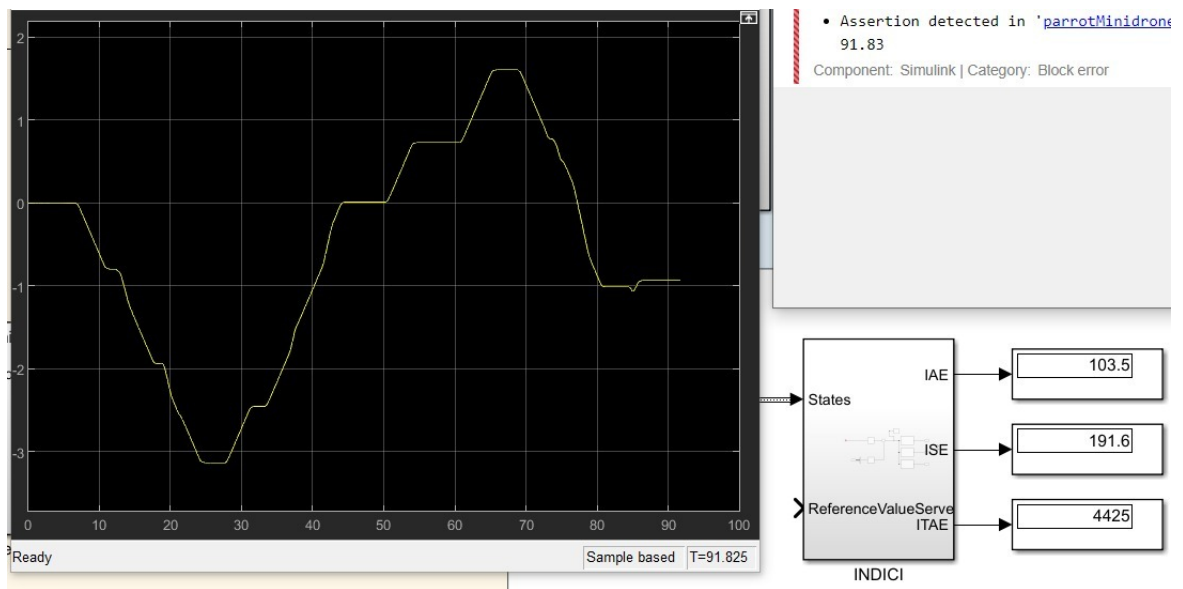


Figura 6.49: Prestazioni roll con la sintesi in  $\omega$

# Capitolo 7

## Conclusioni

Il seguente elaborato aveva come scopo quello di analizzare il comportamento dei controllori di default dei due gradi di libertà del pitch e del roll, e di progettare ulteriori controllori che ne migliorassero le prestazioni, sia a livello di errore di regime permanente, che di tempo di simulazione, mantenendo comunque basse le oscillazioni intorno al valore di riferimento. Si è giunti dunque ai risultati sperati utilizzando le due tecniche di controllo lineare della sintesi in  $\omega$  e del luogo delle radici. Si può notare infatti come tutti i controllori riducano gli indici di prestazione e il tempo di simulazione mantenendo oscillazioni di ampiezza intorno a  $10^{-3}/10^{-4}$ . Infine viene anche eliminata l'instabilità presente nel modello lineare con riferimento variabile del roll.

Uno sviluppo futuro potrebbe essere quello di migliorare ancora tali controllori, abbassandone ulteriormente gli indici di prestazione e il tempo di simulazione, ad esempio utilizzando tecniche di controllo più complesse.

# Bibliografia

- [1] Gambini Daniele. “Studio e sviluppo in Simulink di tecniche di controllo per minidroni”. Università Politecnica delle Marche, 2019.
- [2] Amato Francesco. “Controllo di un Quadcopter”. Univesità Federico II, Napoli, 2012.
- [3] Wei Zhong Fum. “Implementation of Simulink controller design on iris quadrotor”. Naval Postgraduate School, 2015.
- [4] Mezzanotti Luca. “Studio e sviluppo di controllori per l’assetto di mini droni”. Università Politecnica delle Marche, 2019.
- [5] Rosettani Paolo. *Parrot Mambo and Simulink*. Università Politecnica delle Marche, 2020.
- [6] P.Pounds. “Modelling and Control of a Large Quadrotor Robot”. Yale university, 2010.
- [7] Sovon Rokibujjaman. “A thesis on Quadcopter”. 2017.
- [8] Luukkonen Teppo. “Modelling and control of quadcopter”. School of Science, 2011.
- [9] Jirinec Tomas. “Stabilization and control of unmanned quadcopter”. CZECH TECHNICAL UNIVERSITY IN PRAGUE, 2011.