UNIVERSITÀ
POLITECNICA
DELLE MARCHE

Facoltà di Ingegneria
Corso di Laurea in Ingegneria Informatica e dell'Automazione

Studio e sviluppo di sistemi di controllo lineari per l'assetto di droni

Study and development of linear control systems for drone attitude

Candidato:
Manuel Gizzarone

Relatore:
Prof. Gianluca Ippoliti

Correlatore:
Prof. Giuseppe Orlando

Anno Accademico 2021 - 2022

UNIVERSITÀ
POLITECNICA
DELLE MARCHE

Facoltà di Ingegneria
Corso di Laurea in Ingegneria Informatica e dell'Automazione

Studio e sviluppo di sistemi di controllo lineari per l'assetto di droni

Study and development of linear control systems for drone attitude

Candidato:
Manuel Gizzarone

Relatore:
Prof. Gianluca Ippoliti

Correlatore:
Prof. Giuseppe Orlando

Anno Accademico 2021 - 2022

Università Politecnica delle Marche
Facoltà di Ingegneria
Corso di Laurea in Ingegneria Informatica e dell'Automazione
Via Brecce Bianche – 60131 Ancona (AN), Italy

"Guardate le stelle e non i vostri piedi. Provate a dare un senso a ciò che vedete, e chiedervi perché l'universo esiste.
Siate curiosi."

Stephen Hawking

Sommario

Il presente elaborato propone una trattazione completa di tutte le fasi di progettazione per i sistemi di controllo degli angoli di orientamento *roll, pitch* e *yaw* di un drone a quattro eliche, con l'obbiettivo di ottenere dei miglioramenti nelle prestazioni rispetto ai controllori originali. Il quadricottero di riferimento è il *Parrot Mambo*, scelto per la sua facilità nell'interfacciarsi a MATLAB e *Simulink*, ambienti utilizzati per le simulazioni di volo e l'analisi dei dati. Verranno introdotti e descritti accuratamente tutti i concetti necessari alla comprensione dei argomenti affrontati, con particolare attenzione al modello *Simulink* di riferimento e agli strumenti utilizzati per la realizzazione dei sistemi di controllo lineari, quali "sintesi per tentativi" e "rappresentazione in spazio di stato". Il tutto è stato svolto esclusivamente in simulazione, attraverso il modello *Simulink* ed una serie di Toolbox utili alla renderizzazione di ambienti virtuali di volo.

Dopo la sintesi e l'implementazione dei nuovi sistemi di controllo all'interno del modello *Simulink* utilizzato, per validare la loro qualità si è passati alla fase di *testing*. Sono state eseguite numerose simulazioni di volo in differenti condizioni, prima utilizzando segnali di riferimento costanti e, dopo aver constatato il corretto funzionamento dei controllori nel mantenimento di una posizione di stabilità (*hovering*), si è passati al caso di riferimenti variabili nel tempo, con l'obbiettivo di valutare la risposta del sistema anche in condizioni più "critiche". Tutti i risultati ottenuti con i nuovi regolatori lineari per controllo della dinamica di volo del drone (non lineare), sono stati analizzati e confrontati con quelli degli originali sistemi di controllo, della tipologia *PID* (non lineari).

In collaborazione con il progetto di tesi [1], riguardante lo sviluppo di algoritmi di visione per l'inseguimento di traiettorie per droni, sono stati eseguiti test sui controllori sviluppati anche in determinate circostanze, al fine di valutare più accuratamente la loro qualità e migliorare la risposta degli algoritmi utilizzati per l'inseguimento della traiettoria.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione ai droni

1.1 Aeromobili

Fin dai tempi più antichi, uno tra i più grandi desideri dell'uomo è quello di volare. Grazie all'evoluzione tecnologica e scientifica questo è ormai possibile da diversi anni grazie all'invenzione dei cosiddetti *aeromobili*. Un aeromobile è una macchina costruita dell'uomo capace di spostarsi nell'aria, nata dell'esigenza di trasportare persone e cose in modo più rapido sfruttando appunto l'atmosfera terrestre. Un qualsiasi aeromobile per la sostentazione sfrutta le forze di reazione dell'atmosfera stessa in cui esso è immerso. Tra queste quelle più importanti sono la *spinta di Archimede* e la *portanza*, una tra le componenti della *forza aerodinamica*.

In genere la classificazione degli aeromobili viene effettuata principalmente in base al tipo di sostentazione aerodinamica utilizzata per il volo e, in secondo luogo, in base al tipo di propulsione utilizzata. Si possono quindi classificare nel seguente modo:

Aerostati: mezzi meno densi dell'aria che ottengono una spinta ascenzionale in base al *principio di Archimede*. Un esempio di aerostato è la mongolfiera.

Aerodine: appartengono a tale categoria gli aeromobili *più pesanti* dell'aria (aventi una densità maggiore), la cui sostentazione è ottenuta grazie alla presenza di un *"organo sostentante"* capace di interagire con l'aria circostante in modo da generare una spinta ascensionale necessaria al volo. In base alla tipologia di tale *organo sostentante* si possono distinguere tre categorie:

- aerodine a sostentazione aerodinamica: la sostentazione è dovuta al moto relativo dell'*organo sostentante* rispetto all'aria. Ad esempio, rientrano in tale classe di aeromobili i *velivoli* (comunemente conosciuti come aeroplani o aerei), la cui spinta ascensionale è dovuta alla presenza di superfici alari che garantiscono la generazione della *portanza*, gli *elicotteri* in cui l'organo sostentante è una pala in rotazione (grazie a dei motori) attorno ad un albero fisso, etc. In tale categoria appartengono anche i ***droni*** la cui azione sostentante risulta essere analoga a quella di un comune velivolo o di un elicottero con più pale rotanti (tricotteri, quadricotteri, octacotteri, etc.).

- aerodine a sostentazione per reazione diretta: la sostentazione è garantita dalla presenza di sistemi meccanici (*turboreattori*) capaci di accelerare grandi quantità di aria o gas in modo tale da garantire una spinta ascensionale. Un esempio sono i *missili*.
- aerodine a sostentazione mista: tali macchine utilizzano entrambe le tipologie di sostentazione precedentemente citate per garantire una spinta sufficiente al volo.

1.2 Droni

1.2.1 Definizione e applicazioni

Il *drone* è un particolare tipo di aeromobile caratterizzato dall'assenza di un pilota umano a bordo. Per questo motivo essi appartengono alla famiglia degli *aeromobili a pilotaggio remoto*(A.P.R.) e di conseguenza anche alla più amplia categoria degli U.A.V. (*Unmanned aerial vehicle*) cioè veicoli aerei capaci di eseguire azioni e portare a termine compiti specifici in maniera del tutto autonoma e senza l'intervento umano.

Il volo è controllato da un insieme di sistemi elettronici (sensori, attuatori) i quali possono essere gestiti da remoto da piloti a terra (*aeromobili a pilotaggio remoto*) oppure in maniera del tutto automatica da dei software di controllo chiamati *Flight Control Systems* caricati direttamente all'interno del "computer" di bordo del drone.

I primi veicoli aerei a pilotaggio remoto furono sviluppati nel corso del XX secolo per scopi militari ma il loro sviluppo è continuato anche nel XXI secolo grazie all'avanzamento delle tecnologie di controllo e alla diminuzione dei costi di progettazione e produzione. Oggi infatti essi risultano trovare applicazione in svariati campi non militari come ad esempio per l'ispezione e il monitoraggio , per le consegne di prodotti, per la sorveglianza, etc. Tra i settori in cui essi hanno acquisito un ruolo di fondamentale importanza troviamo l'agricoltura, il commercio e l'industria; è grazie a tecnologie come queste per cui sentiamo molto più spesso parlare, ad esempio, di agricoltura e industria 4.0 (figura 1.1). Questo perché essi permetto di eseguire particolari azioni in maniera molto più precisa e accurata rispetto all'essere umano e soprattutto in totale autonomia.

Figura 1.1: Drone agricoltura 4.0

1.2.2 Principali caratteristiche

La crescente richiesta di apparecchi di questo tipo in svariati settori applicativi, ha spinto le varie aziende produttrici ad ideare e realizzare nuovi veicoli con differenti caratteristiche per soddisfare tutti i requisiti richiesti in base al loro impiego. Infatti, si trovano in commercio droni sempre più piccoli ed economici in modo tale da renderli più versatili e utilizzabili in diversi contesti e condizioni ambientali e soprattutto accessibili ad un maggior numero di persone. Nella tabella 1.1 vengono riportate alcune delle caratteristiche tipiche di droni attualmente in commercio.

Tabella 1.1: Caratteristiche tipiche droni commerciali

Specifica	Valore
Dimensione	< 50 cm
Peso	< 900 g
Raggio	1 - 15 km
Altitudine	< 300 m
Costo	20 - 1700 €

Volendo invece fare una classificazione in base al sistema di volo utilizzato, si possono distinguere due categorie:

Droni ad ala fissa: apparecchi simili a piccoli velivoli il cui controllo in volo è garantito dalle superfici alari e dalla coda. Le principali caratteristiche sono: elevata autonomia e sicurezza di volo, possibilità di trasporto di carichi con dimensioni abbastanza elevate. Tali apparecchi hanno però una dinamica di volo limitata essendo impossibilitati nel rimanere in volo stazionario e nel decollare/atterrare verticalmente.

Droni ad ala rotante: è la categoria più comune in quanto tutti i droni che siamo abituati a vedere in circolazione appartengono ad essa. Essi sono caratterizzati da eliche che ruotando ad alte velocità consentono di generare una spinta necessaria al sollevamento verticale. I droni ad elica rotante, infatti, rientrano nella categoria dei veicoli aerei ad atterraggio e decollo verticale o anche *Vertical Take-Off and Landing* (V.T.O.L.). In questa tipologia di droni è quindi anche possibile mantenere una posizione stazionaria durante il volo, come negli elicotteri, caratteristica che li rende molto utili in applicazioni in cui è richiesta precisione e stabilità nei movimenti durante il volo.

Essi vengono anche chiamati *multicotteri* in quanto possono essere costituiti da più eliche rotanti: si parla infatti di *tricotteri*, *quadricotteri*, *octacotteri*, etc. in base al numero di eliche presenti. I più comuni attualmente in commercio sono i *quadricotteri* mentre per le applicazioni professionali vengono impiegati solitamente droni di maggiori dimensioni e quindi *octacotteri*. Di seguito vengono mostrate alcune tra le tipologie citate.

(a) Drone ad ala fissa

(b) Octacottero

(c) Quadricottero della DJI

Figura 1.2: Esempi tipologie di droni

D'ora in avanti prenderemo in considerazioni solamente la **tipologia ad ala rotante**, in quanto è quella di interesse per lo sviluppo dell'elaborato.

1.2.3 Componenti principali e sensoristica

Per la realizzazione di movimenti durante il volo in totale autonomia, i droni hanno la necessità di essere equipaggiati con determinate componenti, sia hardware che software, e sensori in modo tale da potersi spostare nello spazio autonomamente e soprattutto in sicurezza, cioè evitando situazioni pericolose per il drone stesso e per l'ambiente circostante. Quindi, grazie alla presenza di un gran numero di sensori disposti lungo tutta la superficie del drone, quest'ultimo riesce, ad esempio, a rilevare autonomamente potenziali pericoli durante il volo e tramite un sistema di controllo e di attuatori ad eseguire determinate azioni atte ad evitare tali situazioni e mantenere quindi in sicurezza l'ambiente in cui il drone opera.

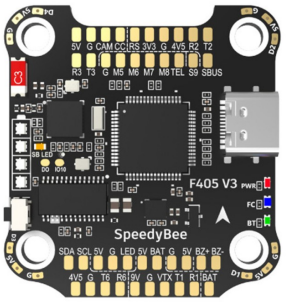
In linea generale le principali parti che costituiscono un qualsiasi drone ad ala rotante sono:

- **Telaio (frame)**: è la struttura portante della varie parti che costituiscono il drone. É di fondamentale importanza per la stabilità del drone e di conseguenza necessita di un'elevata precisione nel bilanciamento della struttura stessa.
- **Attuatori (actuator)**: essi permetto al drone di sollevarsi e rimanere in volo e di eseguire gli spostamenti voluti nello spazio. Generalmente essi sono di natura

elettrica sfruttando le ottime prestazioni fornite dai motori cosiddetti *Brushless*, cioè senza spazzole (componenti caratterizzanti dei motori DC a collettore). Per la regolazione della velocità di rotazione si utilizzano i cosiddetti ESC, *regolatori di velocità*.

- **Eliche (propeller)**: componenti montate direttamente sugli attuatori che ruotando ad alte velocità permettono la generazione di una spinta ascensionale necessaria al volo del drone. Esistono diverse tipologie in base alle modalità costruttive impiegate; troviamo, ad esempio, eliche a passo fisso e a passo variabile.
- **Batteria (battery)**: componente fondamentale al funzionamento di tutti i dispositivi montati sul drone. Essa fornisce l'energia necessaria al funzionamento di ciascuna componente.
- **Sensori (sensors)**: essi sono montati all'interno del telaio e forniscono continuamente informazioni al sistema di controllo in merito a numerose variabili come, ad esempio, posizione ed orientamento nello spazio. Le principali tipologie di sensori necessarie al volo sono: sensore ad ultrasuoni, sensore di pressione, sensore per campo megnatico, girosopio ed IMU (*Unità di misura inerziale*.
- **Scheda di controllo (Flight Controller)**: è il cuore del drone ed è grazie ad essa che riesce a volare seguendo la traiettoria desiderata. É la principale componente hardware in cui al suo interno è caricato il software di controllo, composto da complessi algoritmi che a partire dai valori delle variabili misurate dai sensori permettono di calcolare le giuste spinte che ogni attuatore dovrà generare. Tutto ciò è possibile grazie alla presenza di una CPU (*Central Peocessing Unit*) all'interno della scheda di controllo stessa.
- **Componenti aggiuntive-ausiliarie**: oltre alle componenti sopra citate, spesso è possibile equipaggiare il drone con qualsiasi tipologia di sensore/componente in base a specifiche necessità. Alcuni esempi sono: GPS, videocamere, termocamere, sensori di temperatura e pressione, sensori di gas, porta pacchi per il trasporto di merce, etc.

Poiché gli **studi effettuati nell'elaborato riguardano** il quadricottero *Parrot Mambo*, andiamo a dare una piccola introduzione alla struttura e ai principali principi fisici dei **quadricotteri** in generale.



(a) Flight Controller

(b) Motore *Brushless*

Figura 1.3: Esempi componenti di un drone

1.3 Quadricotteri

1.3.1 Configurazione: *"a più"* e *"a croce"*

In generale, la struttura di un quadricottero può essere di due diverse tipologie: *struttura a più (+)* e *struttura a croce (X)*. La differenza sostanziale è nell'orientamento dei 4 rotori rispetto al sistema di riferimento solidale con il drone, il cosiddetto *Body Fixed Frame of Reference* (B.F.F.R.), dove per convenzione gli assi x, y e z sono orientati rispettivamente verso la parte frontale, laterale destra ed inferiore del drone, con una conseguente diversa distribuzione delle forze che ogni motore dovrà esercitare per permettere gli spostamenti in volo.

Configurazione a più o a "+"

I quattro rotori sono allineati agli assi x ed y del *Body Frame* ovvero si hanno due motori in corrispondenza della parte anteriore e posteriore del drone e gli altri due sono posti ortogonalmente ai precedenti, cioè lungo l'asse y, uno nella parte destra ed uno nella parte sinistra rispetto al corpo.

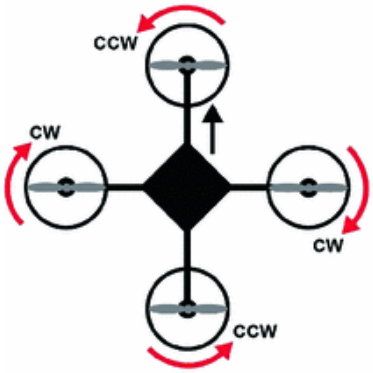


Figura 1.4: Configurazione a più

Configurazione a croce o a "X"

I quattro rotori sono disallineati rispetto al *Body Frame*, nello specifico i motori sono ruotati di un angolo pari a 45° rispetto gli assi x ed y e di conseguenza il corpo del drone si trova in una posizione intermedia rispetto i quattro rotori.

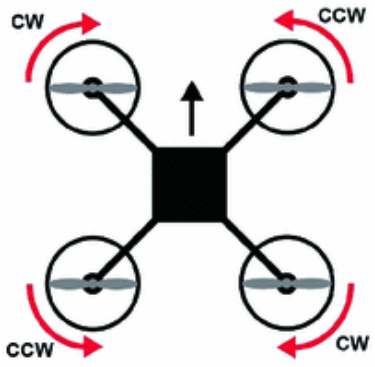


Figura 1.5: Configurazione a croce

Per entrambe le configurazioni è garantita una perfetta simmetria complessiva del drone la quale lo rende una sistema adatto a stabilizzarsi in un punto dello spazio durante il volo, la cosiddetta posizione di *Hovering*.

Le caratteristiche di volo complessive sono analoghe per tutte e due le configurazioni.

Da questo punto dell'elaborato in poi **prenderemo in considerazione solamente la *configurazione a croce*** in quanto è la struttura del drone (*Parrot Mambo*) utilizzato per lo sviluppo della tesi.

1.3.2 Fisica del volo

I quadricotteri, ma più in generale i droni, sono dei sistemi capaci di compiere spostamenti nello spazio e di conseguenza presentano 6 gradi di libertà (*Degrees of Freedom* (D.o.F.): oltre alle 3 direzioni di spostamento nello spazio $\mathbb{R}^3$ bisogna considerare anche gli spostamenti angolari attorno gli assi x, y e z del *Body Frame*, i rispettivi angoli di *roll, pitch* e *yaw* (Figura 1.6). Tali spostamenti angolari per convenzione vengono anche indicati nel seguente modo:

- Angolo di **Rollio** ϕ: spostamenti angolari attorno l'asse x del *Body Frame*.
- Angolo di **Beccheggio** θ: spostamenti angolari attorno l'asse y del *Body Frame*.
- Angolo di **Imbardata** ψ: spostamenti angolari attorno l'asse z del *Body Frame*.

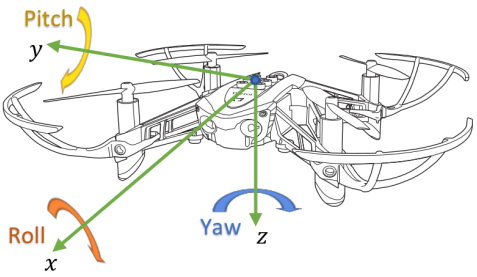


Figura 1.6: Gradi di libertà Parrot Mambo

Si hanno quindi 6 D.o.F. ma per il controllo degli spostamenti si fa affidamento solamente alla potenza fornita ad ognuno dei 4 motori. Per questo motivo questo risulta essere un esempio di **sistema sotto-attuato**, cioè ci sono meno variabili di controllo (4 motori) rispetto al numero di gradi di libertà. Per questo motivo non sarà possibile raggiungere contemporaneamente dei riferimenti dati per tutti e 6 D.o.F. É comunque possibile controllare tutti i movimenti del quadricottero in quanto quattro tra i sei gradi di libertà dipendono in realtà l'uno dall'altro e di conseguenza possono essere considerati come un'unica variabile da controllare (roll - x / pitch - y). In totale, quindi, i D.o.F. "semplificati" sono quattro, come le variabili di controllo (attuatori).

Vediamo ora una breve descrizione dei fenomeni fisici caratteristici della dinamica di un qualsiasi drone durante il volo.

Movimenti nello spazio

Per la realizzazione di spostamenti di qualsiasi genere nello spazio, i quadricotteri sfruttano un importante principio fisico: la **Terza legge della Dinamica di Newton** il quale afferma che *"se un corpo A esercita una forza su un corpo B, allora B esercita su A una forza uguale e contraria"*. Applicato al nostro caso si ha che il momento torcente generato dalla rotazione di un'elica in una dato verso, genera a sua volta un momento torcente identico in modulo ma di verso opposto.

Poiché nei quadricotteri si hanno due tipologie di eliche, *tipologia cw* e *tipologia ccw*, capaci di generare una spinta ascensionale la prima tramite rotazione oraria (*clockwise o cw*) e l'altra tramite rotazione antioraria (*counterclockwise o ccw*), si hanno quindi due coppie di momenti torcenti uguali e opposti.

Nella *configurazione a croce* le eliche che ruotano nello stesso verso e che quindi generano lo stesso momento torcente sono diagonalmente opposte, come mostrato in figura 1.5.

Andando ad agire sull'entità di tali momenti torcenti, cioè modificando (aumentando o diminuendo) le velocità angolari di determinate eliche, è possibile realizzare

qualsiasi spostamento/rotazione nello spazio nonché mantenere una posizione stabile (*Hovering*).

- Posizione di *Hovering*: per il mantenimento di una posizione di stabilità nello spazio è necessario che la somma vettoriale dei momenti torcenti generati sia nulla e che, una volta in volo, la spinta totale esercitata dalle eliche uguagli il peso del drone.

  Questo avviene solamente nel momento in cui tutti e quattro i rotori ruotano a velocità angolare costante (tale da generare una spinta uguale al peso stesso) e uguale tra loro. In questo modo la risultante delle forze e dei momenti agenti sul drone è nulla e quindi si ha una posizione di stabilità (Figura 1.7).

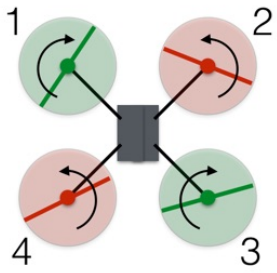


Figura 1.7: Posizione di Hovering

- Spostamento verticale: la fase di decollo è un classico esempio di spostamento verticale fino a raggiungere una determinata quota desiderata. Per permettere tale tipo di traslazione è necessario che tutti e quattro i rotori ruotino alla stessa velocità (altrimenti si verificherebbe l'effetto giroscopico dovuto al momento torcente risultante) tale per cui la spinta totale generata dalle eliche sia maggiore del peso del drone. Aumentando o diminuendo in modo uniforme la velocità di tutti i motori si ottengono rispettivamente traslazioni verso l'alto e verso il basso.

- Spostamento longitudinale: sono tutti gli spostamenti che avvengono lungo la direzione x del *Body Frame*, ovvero movimenti frontali. Per ottenere traslazioni longitudinali, ad esempio in avanti, è necessario andare a aumentare la velocità delle eliche posteriori e diminuire quella delle eliche anteriori (in modo uniforme per non avere torsioni). La spinta risultante risulta essere quindi maggiore nella parte posteriore e di conseguenza si avrà un'inclinazione in avanti (rotazione attorno l'asse y o pitch) la quale permette al drone di avanzare longitudinalmente. Viceversa per spostamenti nel verso opposto alla direzione delle x positive.

  Nella figura 1.8 le eliche in verde (posteriori) sono quelle che ruotano a velocità angolare maggiore rispetto quelle in rosso (anteriori). Lo spostamento sarà quindi frontale con la conseguente formazione di un angolo di *pitch* necessario per ottenere spostamenti longitudinali.

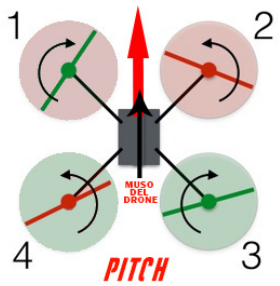


Figura 1.8: Traslazione longitudinale

- Spostamento trasversale: sono tutti gli spostamenti che avvengono lungo la direzione y del *Body Frame*, ovvero movimenti laterali. La logica è analoga al caso di traslazioni longitudinali: ad esempio, per spostamenti verso destra (direzione delle y positive) si procede andando ad aumentare in modo uniforme la velocità delle eliche poste nella parte sinistra del drone e diminuendo quelle nella parte destra. Il risultato sarà quindi un'inclinazione (angolo di roll) verso il lato destro del drone con la conseguente traslazione verso destra. Viceversa per spostamenti verso sinistra.

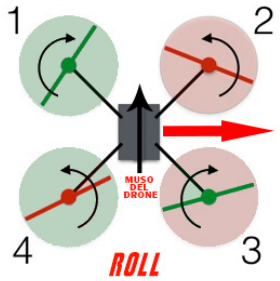


Figura 1.9: Traslazione trasversale

Nell'esempio in figura 1.9 le eliche in verde (laterali sinistre) sono quelle che generano una spinta maggiore rispetto quelle in rosso (laterali destre). Lo spostamento sarà quindi verso destra con la conseguente formazione di un angolo di *roll* necessario per ottenere spostamenti trasversali.

- Rotazione attorno l'asse verticale passante per il C.d.M.: per ottenere rotazioni attorno l'asse z del *Body Frame* si deve per prima cosa garantire il mantenimento dello stato di *Hovering* durante la manovra di *imbardata*. Per questo motivo è necessario che la spinta risultante sia pari alla forza peso del drone. Sotto questa ipotesi, per la realizzazione di manovre di *imbardata* si va ad agire sulla velocità di rotazione delle eliche diagonalmente opposte (quelle con stesso verso di rotazione). In particolare, ricordando che la rotazione delle eliche genera a sua volta un momento torcente uguale in modulo ma opposto a quello delle

eliche stesse, è possibile quindi realizzare, ad esempio, rotazioni orarie andando ad aumentare (uniformemente) la velocità angolare delle eliche di *tipologia ccw* e diminuendo (uniformemente) quella delle eliche di *tipologia cw*. Viceversa per rotazioni antiorarie.

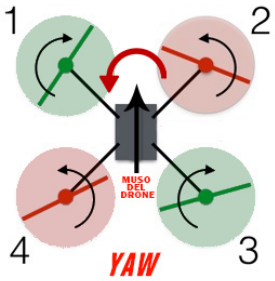


Figura 1.10: Manovra di imbardata

Nell'esempio riportato in figura 1.10 le eliche in verde (tipologia cw) sono quelle con velocità angolare maggiore rispetto quelle in rosso (tipologia ccw). Il momento torcente risultante sarà quindi antiorario con la conseguente rotazione del drone in tale verso (angolo di *yaw*).

Drag Force

Oltre alla spinta ascensionale (*Thrust*) e al momento torcente generato dalla rotazione delle eliche, durante il volo un drone è soggetto anche ad altre forze tra cui quella dovuta all'attrito viscoso con l'aria (fluido in cui è immerso), la cosiddetta *Drag Force* o "forza di trascinamento". Essa è di natura reattiva in quanto è presente solamente durante il moto e non in condizioni stazionarie.

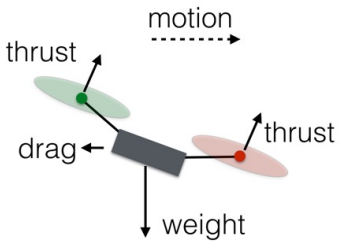


Figura 1.11: Drag Force

Osservando la figura 1.11 si possono ricavare la direzione ed il verso di tale forza:

- Direzione: coincidento con quella del moto traslazionale (*motion*)
- Verso: opposto a quello del moto di traslazione

Per quanto riguarda il modulo della *Drag Force* esso è dato dalla seguente relazione:

$$\vec{F_d} = -\frac{1}{2}\rho C_d A \vec{v}^2 \tag{1.1}$$

in cui:

ρ : densità del fluido in cui è immerso il drone (aria)

C_d : coefficiente di drag

A : area della sezione trasversale del drone (rispetto alla direzione di traslazione)

$\vec{v}$: vettore velocità lineare del drone

Il segno meno nell'equazione 1.1 sta ad indicare che il verso della forza di attrito è sempre opposto alla direzione del moto del drone di velocità $\vec{v}$.

La *Drag Force* ha degli effetti anche sul momento torcente del drone andando ad agire, in particolare, solamente sulla componente relativa a rotazioni attorno l'asse di imbardata (*yaw*). Il momento torcente totale ha quindi il seguente valore:

$$\vec{M_d} = \begin{bmatrix} 0 \\ 0 \\ d(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2) \end{bmatrix} \tag{1.2}$$

Come si nota dall'equazione 1.2, le componenti rotazionali relative agli assi di rollio e di beccheggio, come anticipato, sono nulle mentre in merito al momento torcente sull'asse di imbardata esso ha un valore proporzionale alla somma algebrica dei quadrati delle velocità angolari delle singole eliche ω_i. La costante di proporzionalità d è detta *costante di drag* e varia in base al drone preso in considerazione.

Blade Flapping

Durante la rotazione, le eliche sono soggette a determinate forze causate dalla differenza di pressione generata e da disturbi esterni come, ad esempio, il vento. Negli aeromobili ad ala rotante, come i droni e quindi anche i quadricotteri, le eliche sono progettate in modo tale da essere allo stesso tempo flessibili, per ridurre al minimo le vibrazioni generate dall'attraversamento dell'aria durante il volo, ma robuste e capaci di resistere a forti sollecitazioni senza compromettere la propria struttura.

La capacità della lama dell'elica di flettersi sotto l'effetto delle forze agenti su di essa, prende il nome di *Blade Flapping*.

Nei droni, più in generale negli aeromobili ad ala rotante, il *flap* delle eliche avviene sia verso l'alto che verso il basso. In particolare, si ha la formazione di un angolo di *flap* verso l'alto (rispetto al mozzo del motore) quando l'elica si muove nella stessa direzione del moto traslazionale del drone (*advancing blade*). Invece, si ha la formazione di un angolo di *flap* verso il basso (rispetto al mozzo del motore) nel

momento in cui l'elica è diretta nel verso opposto rispetto al verso di avanzamento del drone (*retreating blade*). Di conseguenza, il *blade flapping* è un fenomeno periodico e per questo motivo anche l'angolo di flapping varia continuamente durante la rotazione dell'elica, passando da valori positivi a valori negativi. Il periodo dipende dalla velocità angolare di rotazione.

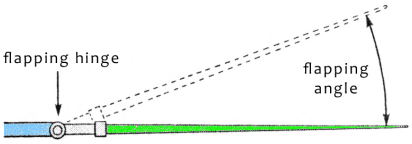


Figura 1.12: Fenomeno di "blade flapping"

In figura 1.12 è mostrato un esempio di angolo di *flap* positivo, cioè l'*advancing flap*. Il fulcro (mozzo del motore) rispetto al quale si misura tale flessione è chiamato *flapping hinge*, riportato nella parte sinistra della figura 1.12.

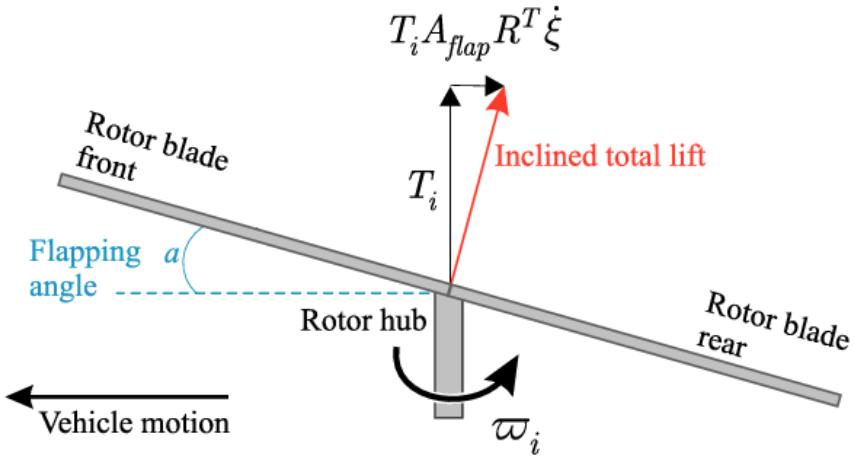


Figura 1.13: Angoli di flapping"

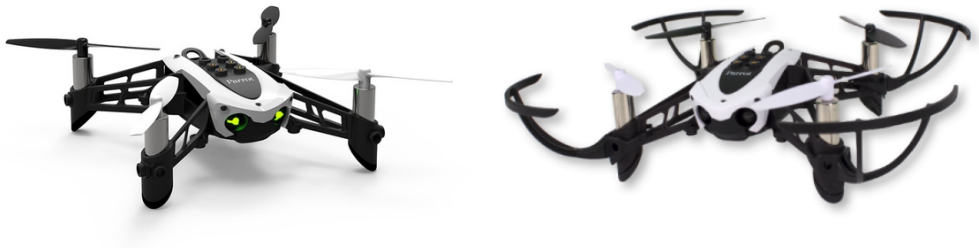
Come si nota dalla figura 1.13, il fenomeno del *balde flapping* influenza la direzione della spinta generata dall'elica. Infatti, la direzione del vettore di *thrust* T (sempre perpendicolare al piano su cui giace l'elica) risulta essere deviata verso la parte posteriore del drone di un angolo α rispetto l'asse passante per il mozzo del rotore.

Analogamente si ha la formazione di un angolo di *flapping* β anche verso la parte laterale del drone.

Per quanto affermato precedentemente, risulta quindi necessario tenere in considerazione tale fenomeno quando si vanno a costruire dei modelli che descrivano la dinamica di un quadricottero.

1.4 Parrot Mambo

Il drone di riferimento per lo studio eseguito nella presente tesi è il *Parrot Mambo minidrone* (figura 1.14) progettato e realizzato dall'omonima casa produttrice *Parrot Drone SAS*.



(a) Senza carene protettrici (b) Con carene protettrici

Figura 1.14: Parrot Mambo minidrone

Il *Parrot Mambo* appartiene alla categoria dei *minidroni* ed è stato scelto proprio per le sue dimensioni ridotte. Inoltre, è interfacciabile facilmente con l'ambiente *MATLAB-Simulink* offrendo quindi la possibilità di progettare controllori personalizzati e caricarli via *bluetooth* direttamente all'interno del *flight control system* del drone.

Specifiche tecniche

Tabella 1.2: Specifiche tecniche Parrot Mambo

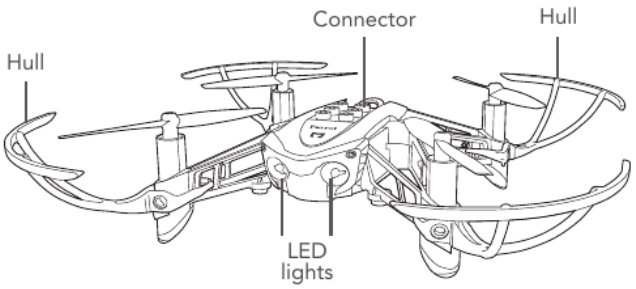
Specifica	Valore
Dimensioni	18 x 18 x 5 cm (con carene)
Peso	63 g
Batteria	660 mAh LiPo
Motori	4 Coreless brushless (2 *cw* e 2 *ccw*)
Velocità max.	30 km/h
Quota max.	4 m
Portata bleutooth max.	100 m
Costo	50€
Sensoristica	IMU, sensore ad ultrasuoni, sensore di pressione
Camera verticale	Risoluzione 720p a 60 FPS

Il pacco batterie da 660 mAh permette un'autonomia di volo di circa 8-10 minuti, a fronte di un tempo di ricarica di 30 minuti con caricatore da 2.1 Ampere.

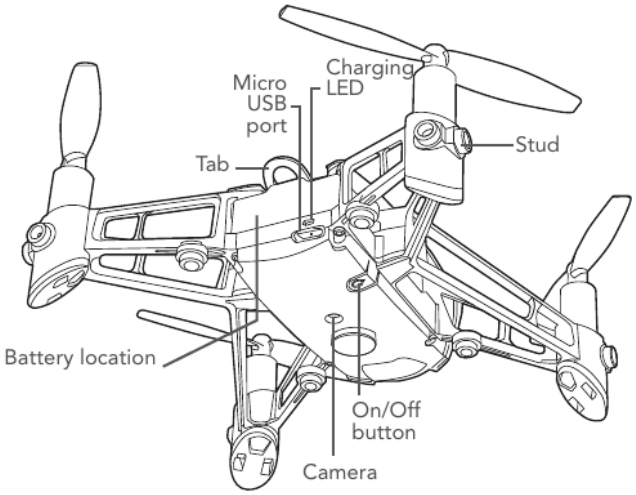
In merito alla sensoristica montata, sensore di pressione barometrica e sensore ad ultrasuoni hanno il compito di monitorare l'altitudine del drone mentre l'IMU (a 3

assi) consente di misurare le variabili inerziali, come velocità e posizione durante il volo.

Per monitorare lo stato del drone durante il volo, sono presenti due indicatori luminosi nella parte frontale, come si può osservare in figura 1.14 e indicato nella figura 1.15a.



(a) Vista anteriore-superiore



(b) Vista posteriore-inferiore

Figura 1.15: Componenti Parrot Mambo

Di seguito indicato il significato di tali indicatori luminosi:

Tabella 1.3: Indicatori luminosi Parrot Mambo

LED	Significato
Verde fisso	Drone pronto
Rosso fisso	Problema rilevato
Arancione fisso	Drone in avviamento
Rosso lampeggiante	Batteria scarica

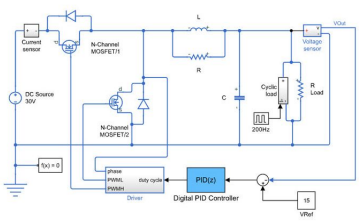
Capitolo 2

Modello *Simulink* del minidrone *Parrot Mambo*

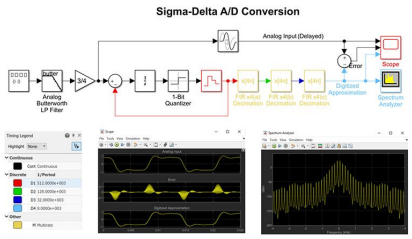
Nel seguente capitolo si andrà ad analizzare nel dettaglio il modello in ambiente *Simulink* del minidrone *Parrot Mambo*, sfruttando il progetto **asbQuadcopter** messo a disposizione dalla *MathWorks*, azienda produttrice di *MATLAB e Simulink*. Verrà fatta una descrizione generale del progetto per poi passare ad un'analisi più dettagliata dei singoli blocchi (sistemi) che lo costituiscono ai fini di comprendere al meglio le sottoparti (sottosistemi) di nostro interesse per il successivo sviluppo di sistemi di controllo lineari che sostituiranno quelli già presenti nel modello (*PID*).

2.1 Breve introduzione all'ambiente Simulink

Simulink è un ambiente di programmazione grafica a blocchi basato su MATLAB utilizzato per la modellazione, la simulazione e l'analisi di sistemi dinamici multi-dominio. La sua interfaccia grafica principale è composta da strumenti per la creazione di diagrammi a blocchi, completamente personalizzabile ed espandibile grazie alla presenza di numerose librerie di blocchi aggiuntive. Offre una stretta integrazione con il resto dell'ambiente *MATLAB* e per questo esso è largamente utilizzato nell'ambito scientifico-ingegneristico, ad esempio nel controllo automatico di sistemi complessi, nello studio ed elaborazione di segnali (figura 2.1b), nella simulazione multi-dominio e nella progettazione basata su modelli.



(a) Elettronica di potenza



(b) Elaborazione di segnali

Figura 2.1: Esempi utilizzo Simulink

La "potenza" del mondo *MATLAB e Simulink* sta nella capacità di generare codice in linguaggio *C, C++, CUDA, PLC, Verilog e VHDL* direttamente dal modello a blocchi disegnato e per questo è possibile programmare i propri sistemi embedded senza scrivere codice, diminuendo quindi la percentuale di errori dovuti alla programmazione.

2.2 Parrot Mambo & Simulink: il progetto asbQuadcopter

L'azienda americana *The MathWorks Inc.* produttrice di *MATLAB* e *Simulink*, come già anticipato, mette a disposizione di tutti gli utenti il progetto **asbQuadcopter** contenente la modellazione completa di un quadricottero, più nello specifico del minidrone *Parrot Mambo*, descritto nella sezione 1.4.

Tramite tale progetto è possibile andare ad eseguire delle simulazioni di volo direttamente all'interno del proprio *PC* e senza l'utilizzo diretto del drone. In questo modo è possibile andare a testare il modello, eventualmente modificato, prima in simulazione e poi nella realtà, caricando i vari controllori direttamente all'interno del drone tramite il pacchetto aggiuntivo *Simulink Support Package for Parrot Minidrones*.

Prima di procedere all'avvio del progetto, per il suo corretto funzionamento è necessario essere in possesso dei seguenti *Toolbox*:

1. *Control System Toolbox*
2. *Signal Processing Toolbox*
3. *Aerospace Blockset*
4. *Aerospace Toolbox*
5. *Simulink 3D Animation*
6. *Simulink Coder*

Per l'installazione di *Toolbox* aggiuntivi è sufficiente cercarli all'interno dell'*Add-On Explorer* che si aprirà seguendo il percorso *Add-Ons - Get Add-Ons* direttamente dall'interfaccia grafica *MATLAB*.

Per avviare la creazione di un nuovo progetto all'interno del nostro *PC* è sufficiente digitare all'interno della Command Window di MATLAB il comando *asbQuadcopterStart*. Una volta generato, è possibile trovare il progetto seguendo il seguente percorso: *C:/Users/nomeUtente/MATLAB/Projects/examples*.

All'apertura del progetto verranno automaticamente inizializzate nel *Workspace* tutte le variabili d'ambiente necessarie al funzionamento del modello e verranno visualizzate due finestre, una contenente il progetto *Simulink* complessivo (figura 2.2) e l'altra la simulazione in 3D del volo del drone (figura 2.3).

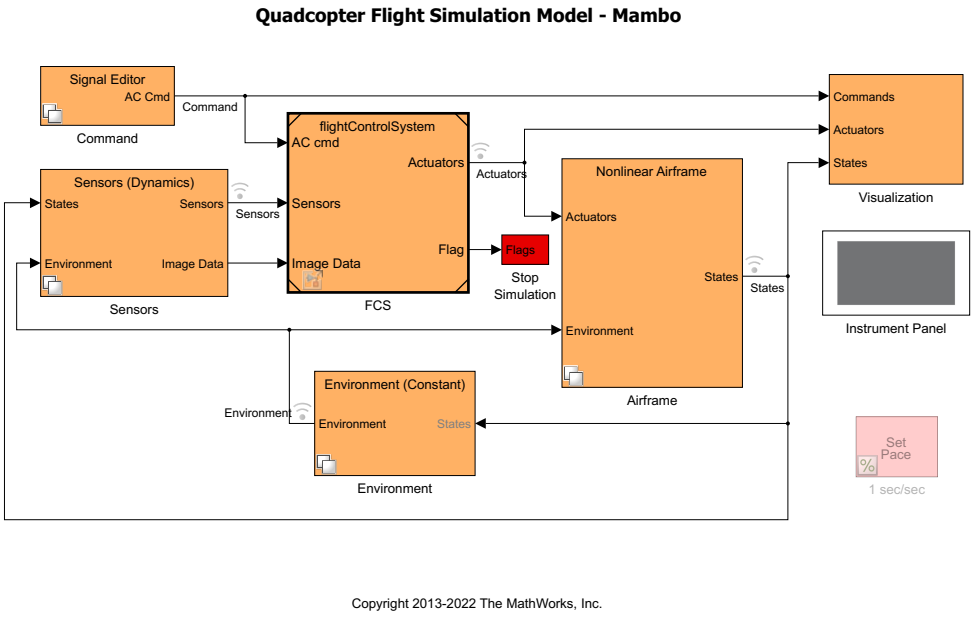


Figura 2.2: Quadcopter Flight Simulation Model

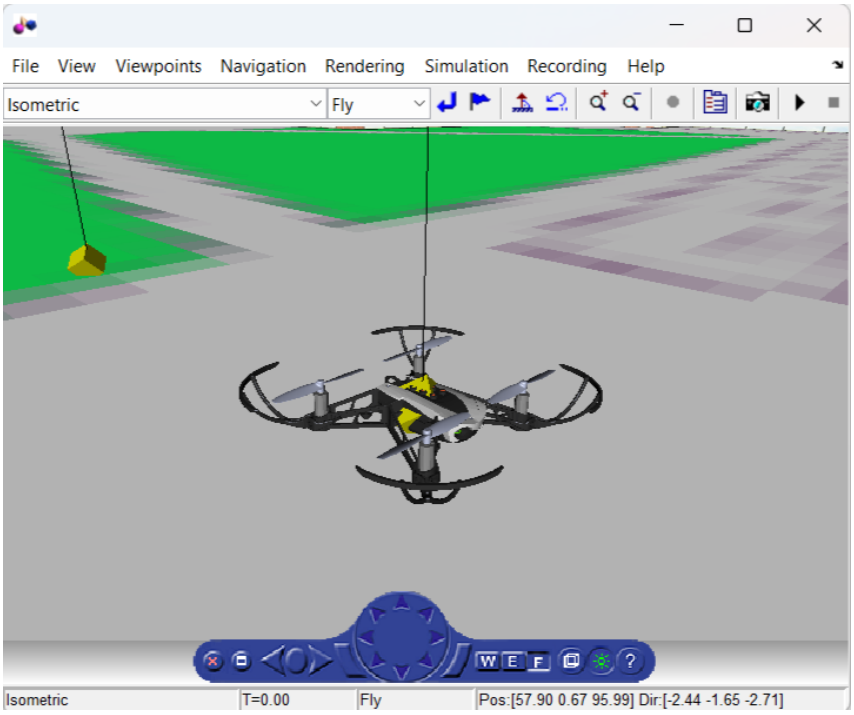


Figura 2.3: Simulazione 3D del volo

Osservando lo schema in figura 2.2 si può notare la classica struttura del *controllo in retroazione*, in cui il processo $P(s)$ da controllare è rappresentato dal sottosistema *Airframe* mentre il controllore $G(s)$ si trova all'interno del blocco *Flight Control System (FCS)*, il quale progetta la legge di controllo sulla base delle informazioni ricevute in merito allo stato del processo, ricavato tramite le misure rilevate dei sensori del drone (blocco *Sensors*) e all'ambiente, rappresentato dal blocco *Environment*. I riferimenti $u(s)$ vengono inviati al controllore tramite il blocco *Command*.

2.2.1 Componenti del modello complessivo

Il modello di simulazione del volo complessivo risulta essere composto da una serie di sistemi ciascuno con compiti specifici:

Command : blocco responsabile della generazione dei segnali di riferimento desiderati.

Sensors : blocco che ha il compito di gestire tutta la parte sensoristica del drone. In particolare, una volta misurate tutte le variabile di stato e d'ambiente sfruttando appunto i vari sensori, comunicherà i valori letti al sistema di controllo del drone.

Flight Control System (FCS) : blocco contenente il "cervello" del modello ovvero il sistema di controllo che, a partire dai riferimenti in ingresso e dai valori delle variabili di stato e d'ambiente, progetta la legge di controllo adeguata, ovvero piloterà la potenza fornita ai singoli motori in modo da rispettare i riferimenti desiderati.

Environment : blocco che ha il compito di riprodurre l'ambiente in cui è immerso il drone e di conseguenza conterrà i valori delle variabili d'ambiente.

Stop Simulation : blocco di controllo che, a partire dai dati forniti dai sensori, permette di interrompere immediatamente il volo nel caso in cui si verifichino potenziali situazioni di pericolo per il drone (angoli di inclinazione troppo elevati, ostacoli, sitazioni non riconosciute, etc.).

Airframe : blocco contenente il processo da controllare. Al suo interno è presente il modello fisico-matematico che descrive la dinamica di un quadricottero durante il volo.

Visualization : blocco ausiliario che consente la visualizzazione, istante per istante, dei segnali di riferimento in ingresso al sistema, delle variabili di stato del processo e d'ambiente e della potenza fornita agli attuatori.

Instrument Panel : blocco contenente tutti i dati di volo del quadricottero, quali velocità, altitudine, inclinazione, etc.

Set Pace : blocco che consente di modificare la frequenza di aggiornamento dei dati, in modo da rendere più fluida la visualizzazione della simulazione di volo.

Essendo l'elaborato volto alla realizzazione di controllori lineari per alcuni tra i gradi di libertà del quadricottero, i blocchi di fondamentale importanza per il proseguo della tesi sono ***Command, FCS, Airframe*** *e* ***Visualization***.

Per completezza di trattazione verranno comunque descritti tutti i principali sotto-sistemi del modello, soffermando l'attenzione sui quattro sopracitati.

Di seguito il dettaglio delle varie sotto-parti del *Flight Simulation Model*.

2.3 Blocco *Airframe*

Entrando nel blocco *Airframe* esso risulta essere suddiviso in due blocchi, *Linear Airframe* e *Nonlinear Airframe*, come osservabile in figura 2.4. Le due sotto-parti rappresentano rispettivamente il modello linearizzato e quello non lineare della dinamica di volo del drone. Il secondo risulta essere quello che più si avvicina alla realtà e che quindi riproduce meglio la fisica durante il volo. Tramite la variabile globale *VSS_VEHICLE*, caricata nel Workspace all'apertura del progetto, è possibile andare a scegliere quale tra i due modelli utilizzare per le simulazioni, in particolare se:

- *VSS_VEHICLE = 1* ⇒ Modello non lineare
- *VSS_VEHICLE = 0* ⇒ Modello lineare

Di default la variabile è settata sul modello non lineare e quindi sul valore 1.

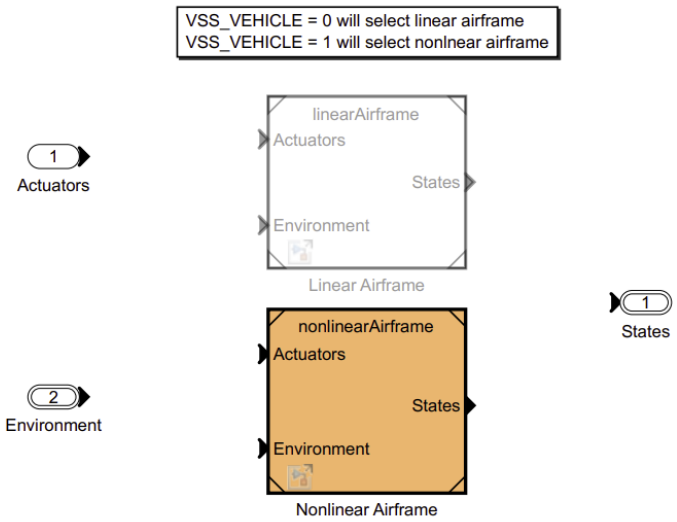


Figura 2.4: Blocco Airframe

Per entrambe le configurazioni e quindi per l'intero blocco *Airframe*, si noti la presenza di due ingressi e di un'uscita: il primo ingresso *Actuators* rappresenta la potenza (*Thrust*) sviluppata dagli attuatori, calcolata dal sistema di controllo, il secondo ingresso *Environment* contiene le misure di tutte le variabili d'ambiente. Tramite tali ingressi il risultato del blocco sarà nient altro che lo stato del sistema, rappresentato dalla variabile d'uscita *States*.

2.3.1 *Nonlinear Airframe*

Il modello del processo, essendo di natura non lineare, sarà contenuto all'interno del blocco *Nonlinear Airframe*, mostrato in figura 2.5.

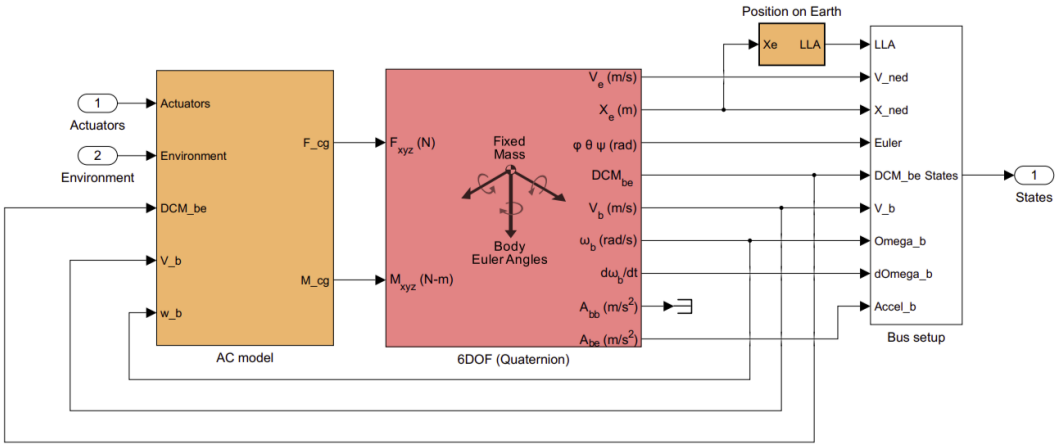


Figura 2.5: Blocco Nonlinear Airframe

Il blocco in figura 2.5 risulta a sua volta essere composto da ulteriori blocchi, *AC model, 6DOF, Position on Earth* e *Bus setup*.

Blocco AC model

Il blocco *AC model*, mostrato in figura 2.6, contiene la modellazione matematica della fisica di volo di un quadricottero e quindi ha il compito di calcolare le forze ed i momenti torcenti applicati al drone rispetto al *Body Frame*, cioè il sistema di riferimento solidale con esso. Si avranno quindi 3 componenti per la forza e 3 componenti per i momenti torcenti, una per ogni asse.

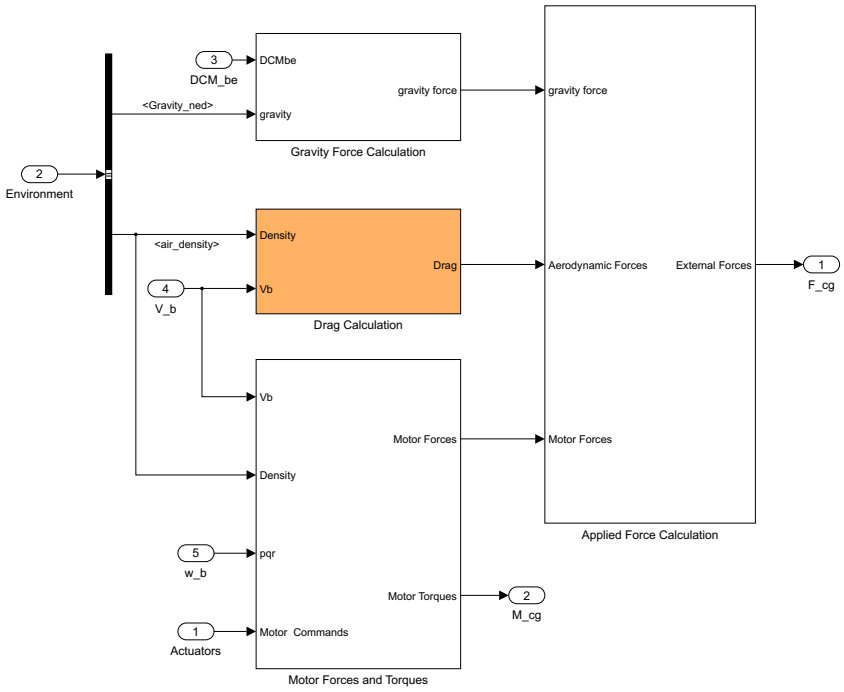


Figura 2.6: Blocco AC model

Osservando la struttura interna del blocco, esso risulta essere suddiviso in ulteriori sotto-sistemi, ognuno con specifici compiti. Vediamo brevemente le loro caratteristiche principali.

Gravity Force Calculation : blocco responsabile del calcolo delle componenti della forza di gravità applicata al drone e quindi rispetto al *Body Frame*. Per operare la conversione da *Inertial Frame* a *Body Frame* viene utilizzata la *Direct Cosine Matrix* DCM_{be}. Il secondo ingresso *gravity* rappresenta la constante di gravità g, mentre il guadagno K è pari al valore della massa del drone. Il risultato del prodotto tra *gravity* e il guadagno K, non è nient altro che la forza di gravità rispetto l'*Inertial Frame*. Tramite il prodotto matriciale con la DCM_{be} si ottiene la *gravity force* desiderata, cioè rispetto il sistema di riferimento solidale con il drone.

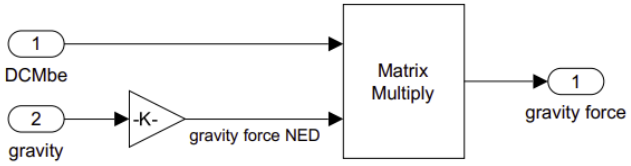
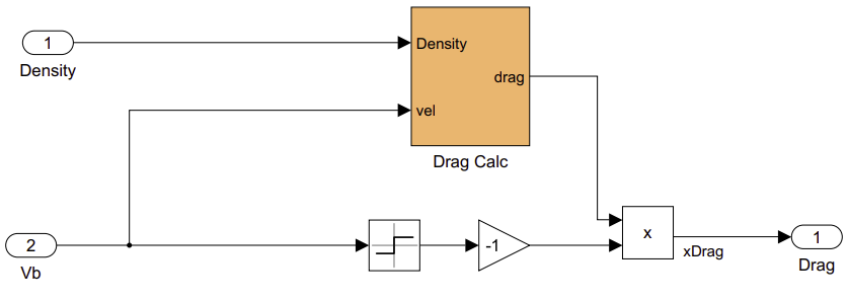
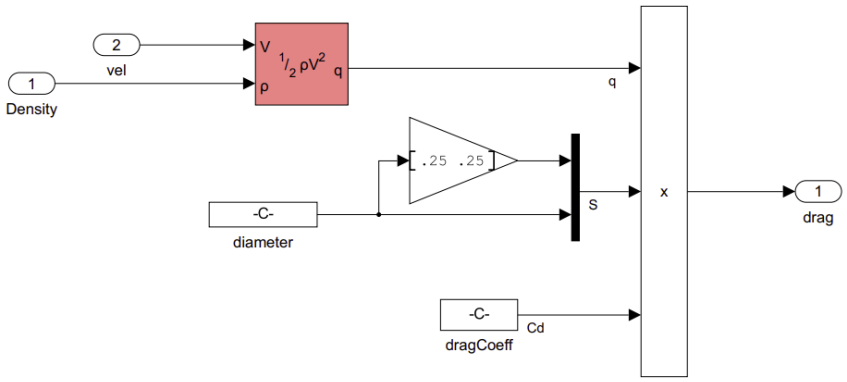


Figura 2.7: Blocco Gravity Force Calculation

Drag Calculation : blocco responsabile del calcolo delle tre componenti della forza di attrito viscoso (*Drag Force*) dovuta allo spostamento del drone nell'aria. Per meggiori dettagli in merito, si veda l'equazione 1.1 in cui vengono spiegate nel dettaglio tutte le proprietà.



(a) Blocco Drag Calculation



(b) Blocco Drag Calc

Figura 2.8: Blocchi calcolo Drag Force

Nella parte inferiore della figura 2.8a si noti come viene preso il segno della velocità V_b, calcolata rispetto al *Body Frame*, e moltiplicato per un guadagno pari a -1. Questo perché, come argomentato nella sezione 1.3.2, la *Drag Force* ha sempre direzione concorde con la velocità del moto ma verso opposto, in quanto si oppone allo spostamento.

Nella parte superiore del blocco in figura 2.8a notiamo la presenza di un blocco ausiliario *Drag Calc* (figura 2.8b) in cui avviene l'effettivo calcolo del modulo della forza di attrito, rispecchiando quanto riportato nell'equazione 1.1. L'unica precisazione da fare riguarda il calcolo delle superficie di aria A impattata durante il volo (in figura indicata con la lettera S): il quadricottero viene considerato come un parallelepipedo in cui le superfici superiore e inferiore (supposte uguali) siano quattro volte maggiori rispetto le superfici anteriore e posteriore (supposte uguali) e la superficie laterale. Di conseguenza la superficie risulta avere 3 componenti, una per le superfici anteriori e posteriori, una per la superficie laterale ed una per le superfici superiori ed inferiori. La combinazione

delle 3 componenti in un unico vettore, indicato con S, avviene tramite l'utilizzo di un blocco *Mux*.

Motor Forces and Torques : blocco responsabile del calcolo delle forze traslazionali (*Thrust*) e dei momenti torcenti, rispetto al *Body Frame*, generati dalla rotazione di ciascun motore. La sua struttura è mostrata nella figura 2.9.

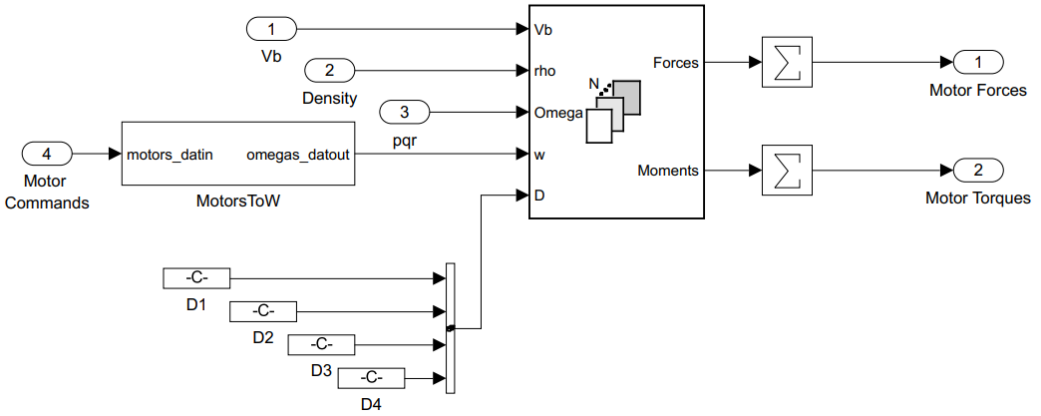


Figura 2.9: Blocco Motor Forces and Torques

Notiamo la presenza di ulteriori due blocchi ausiliari:

- blocco *MotorsToW* (Figura 2.10): ha il compito di convertire i comandi inviati ai motori (*Motor Commands*) in segnali interpretabili dagli stessi, ad esempio valori di tensione e corrente da applicare agli attuatori per ottenere le velocità di rotazione ω desiderate.

- blocco *For Each Subsystem* (Figura 2.11): blocco in cui avviene l'effettivo calcolo delle forze e dei momenti torcenti generati dai rotori del quadricottero. Per ripetere il calcolo per tutti e quattro i motori, tale blocco presenta un ciclo *for each* che di conseguenza eseguirà sempre 4 iterazioni, dove ad ogni iterazione verrà eseguito un algoritmo che a partire da un serie di ingressi come V_b, ω_i, ρ, etc., consente il calcolo delle forze e dei momenti del motore considerato. L'algoritmo è il medesimo per tutti e quattro i motori, in quanto essi sono considerabili identici.

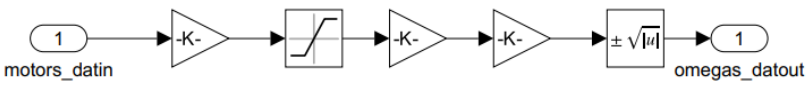


Figura 2.10: Blocco MotorsToW

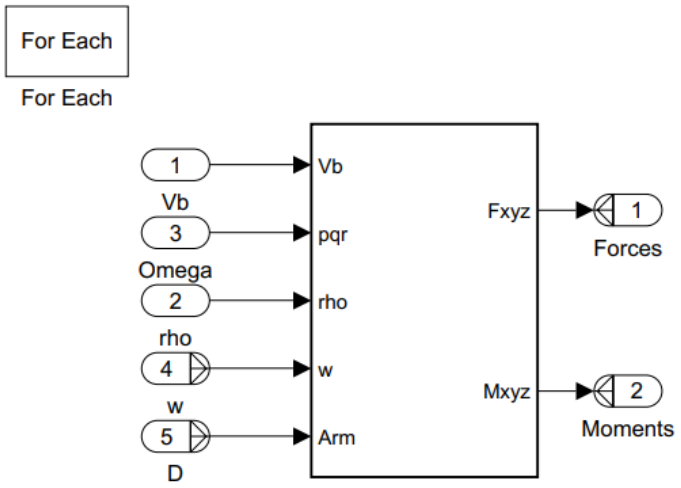


Figura 2.11: Blocco For Each Subsystem

L'implementazione dell'algoritmo si trova all'interno del blocco *Rotor Dinamics*, presente in figura 2.11. La sua logica non viene mostrata in quanto molto complessa e di poca rilevanza ai fini del presente elaborato.

Applied Force Calculation : blocco che consente il calcolo totale delle forze lineari che contribuiscono alla dinamica di volo. In particolare, a partire dai risultati dei blocchi precedentemente riportati ovvero *Gravity Force Calculation, Drag Calculation* e *Motor Forces and Torques*, calcola le componenti della forza lineare risultate rispetto al *Body Frame*. La sua struttura è riportata nella figura 2.12.

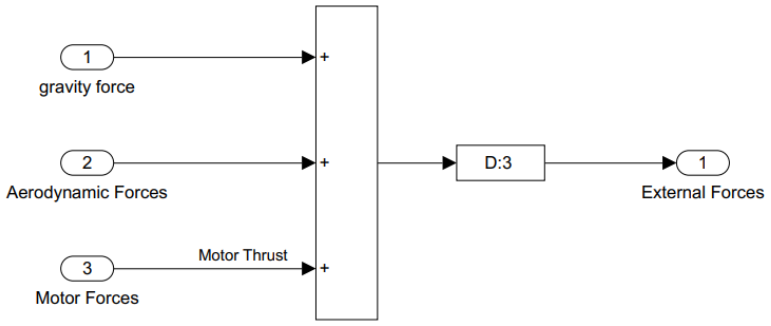


Figura 2.12: Blocco Applied Force Calculation

Blocco 6DOF

Il blocco *Six Degrees of Freedom* (6DOF), presente in figura 2.5, consente di calcolare lo stato complessivo del drone, ovvero tutte le posizioni, rotazioni (angoli

di Eulero), velocità e accelerazioni che costituiscono la dinamica di volo, scaturite dalla risultante delle forze e dai momenti applicati al sistema. É il blocco predefinito del Toolbox *Aerospace Blockset*.

L'attributo *Quaternion* si riferisce all'utilizzo di un'estensione dei numeri complessi, i *quaternioni*, rappresentati da un'espressione del tipo $a + bi + cj + dk$ in cui sono presenti due componenti aggiuntive per evitare problemi legati alla presenza di singolarità nella rappresentazione degli angoli di Eulero.

Blocco Position on Earth

Il blocco *Position on Earth*, in figura 2.13, stima la latitudine, la longitudine e l'altitudine geodetiche a partire dalla posizione X_e e dall'altezza dalla superficie h_{ref} espressi rispetto al sistema di riferimento piatto, il *Flat Earth Frame*. Il sistema di coordinate *Flat Earth Frame* presuppone che l'asse z abbia orientamento positivo verso il basso. Tale sistema di riferimento non tiene conto della curvatura terrestre, a differenza del sistema LLA (*Latitude Longitude Altitude*). L'effettiva conversione da *Flat Earth Frame* a *LLA* avviene all'interno del blocco *Flat Earth to LLA*, visualizzabile in figura 2.13 (in rosso).

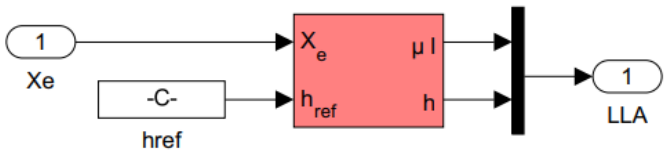


Figura 2.13: Blocco Position on Earth

Blocco Bus setup

Il blocco *Bus Setup*, in figura 2.14, consente di raggruppare tutte le variabili che caratterizzano lo stato del sistema, precedentemente argomentate, in un unico *vettore di stato* (utilizzando un blocco *Mux*), indicato in figura con *States*. Questo viene fatto in modo tale da poter comunicare, tramite un sistema di *bus*, tutte le informazioni sullo stato del sistema durante il volo al sistema di controllo.

2.3.2 *Linear Airframe*

Oltre al modello reale non lineare di un quadricottero, all'interno del progetto *asbQuadcopter* è disponibile anche una versione lineare del modello all'interno del

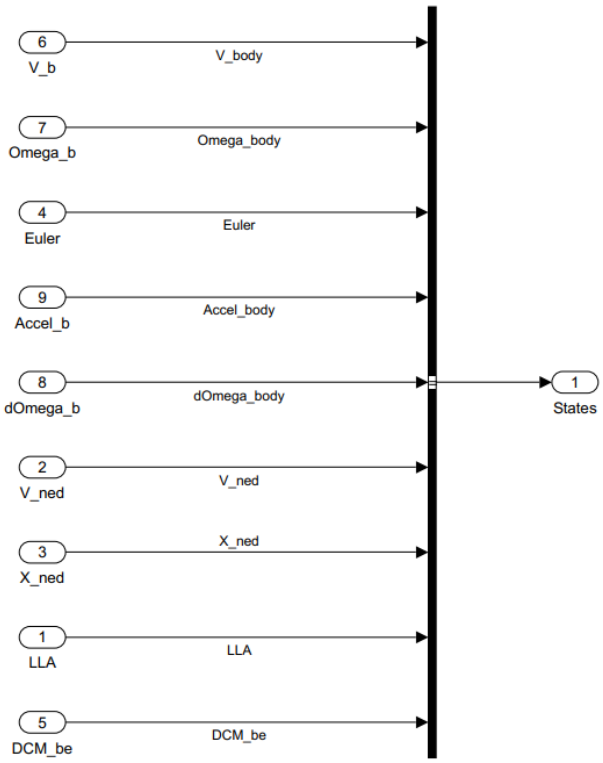


Figura 2.14: Blocco Bus Setup

blocco *Linear Airframe* (Figura 2.15). Essendo una versione linearizzata, la rappresentazione avviene tramite un modello in spazio di stato descritto dal seguente sistema di equazioni lineari:

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx + Du \end{cases} \tag{2.1}$$

dove:

- x: vettore delle variabili di stato $(n \times 1)$
- u: vettore degli ingressi $(m \times 1)$
- y: vettore delle uscite $(q \times 1)$
- A: matrice della dinamica $(n \times n)$
- B: matrice degli ingressi $(n \times m)$
- C: matrice delle uscite $(q \times n)$
- D: matrice di legame diretto ingresso-uscita $(q \times m)$

Nella maggior parte dei casi la matrice D è nulla ($D = 0 \Rightarrow$ sistema strettamente proprio), in quanto l'uscita non dipende in maniera diretta dall'ingresso. Il sistema 2.1 si riduce quindi alla forma:

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx \end{cases} \tag{2.2}$$

L'utilizzo di modelli lineari in spazio di stato semplificano di molto la trattazione, a discapito dell'accuratezza del modello rispetto a quello non linearizzato che più si avvicina alla realtà.

Per l'utilizzo del modello in spazio di stato bisogna per prima cosa individuare le dimensioni dei vettori di stato, ingresso e uscita (x, u, y). Successivamente è necessario ricavare dalla procedura di linearizzazione le 4 matrici A, B, C, D del sistema 2.1 che verranno poi utilizzate nella sintesi dei regolatori mediante tecniche di controllo lineare.

Le fasi necessarie alla costruzione del modello lineare verranno successivamente analizzate nel capitolo 3, in cui verranno ricavate le matrici del sistema in spazio di stato relative al quadricottero in esame, il *Parrot Mambo*. Vedremo che il sistema trovato sarà del tipo 2.2.

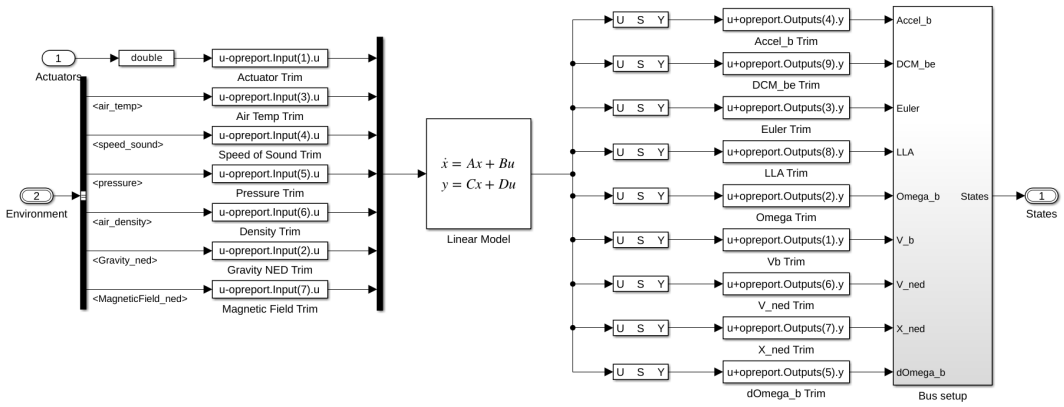


Figura 2.15: Blocco Linear Airframe

2.4 Blocco *Flight Control System* (FCS)

Il *Flight Control System* è uno tra i blocchi di maggiore importanza all'interno del modello complessivo in figura 2.2, che sarà oggetto di modifiche durante lo sviluppo dell'elaborato. Al suo interno è contenuta tutta la logica utilizzata per il controllo del drone, sia in termini di stabilità durante il volo che per eventuali inseguimenti di traiettorie.

In figura 2.16 è riportata la macro-struttura interna all'apertura del blocco *FCS*. In particolare, esso risulta essere composto da due sotto-sistemi principali:

- *Image Processing System*: contiene la logica di processamento delle immagini provenienti dalla telecamera verticale del *Parrot Mambo* (si veda tabella 1.2 per le specifiche tecniche). Esso consente di convertire i fotogrammi raccolti

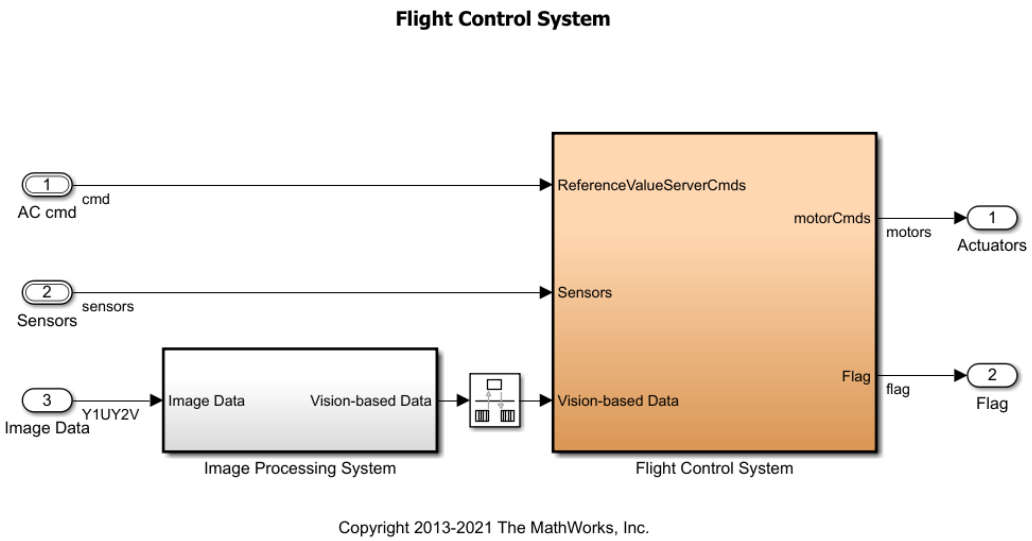


Figura 2.16: Blocco FCS

(in ingresso) in informazioni utili (in uscita) per il controllo della quota e della traiettoria del quadricottero.

Ai fini del presente elaborato tale blocco non è di fondamentale importanza e di conseguenza non verrà analizzato nel dettaglio.

- *Flight Control System*: contiene tutta la logica di controllo, ovvero tutti i sistemi atti al controllo della traiettoria e della stabilità del drone. In particolare, dati in ingresso i segnali di riferimento *AC cmd* (provenienti dal blocco *Command*), le informazioni sui sensori dal blocco *Sensors* e le informazioni in uscita dal blocco *Image Processing System*, consente di determinare il *Thrust* che ogni rotore dovrà generare, in figura indicato con *Actuators*. Si noti la presenza di un'ulteriore uscita *Flag* la quale consente di evitare situazioni indesiderate, come ad esempio collisioni con l'ambiente circostante, disattivando istantaneamente i motori.

  Tale blocco è di fondamentale importanza per lo sviluppo della tesi, poiché al suo interno sono presenti i controllori che successivamente si andranno a sostituire con regolatori lineari.

Analizzando la struttura interna del secondo blocco esaminato (Figura 2.17), essa risulta a sua volta essere divisa in ulteriori sotto-blocchi di seguiti elencati.

sensordata_group : blocco che prende in ingresso i dati provenienti da tutti i sensori montanti a bordo del drone, li organizza e raggruppa all'interno di una struttura dati *sensordata_datin* e tramite il *bus* di sistema essi vengono passati al blocco *Estimator* per la stima dello stato attuale. La struttura è mostrata nella figura 2.18.

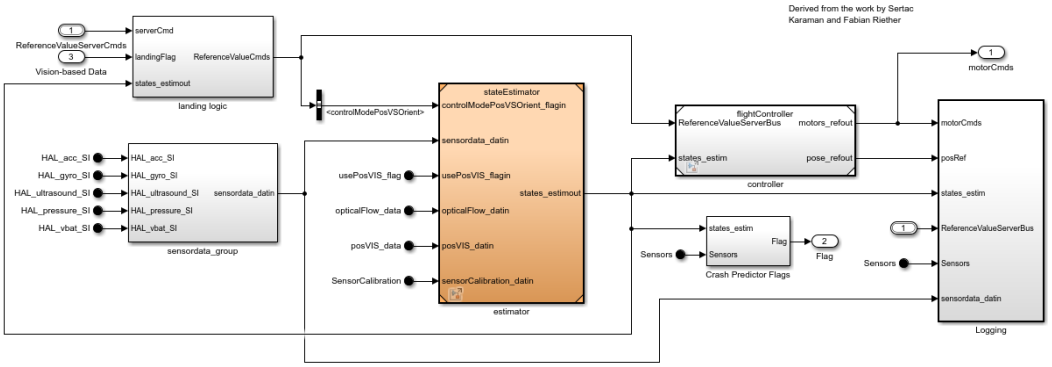


Figura 2.17: Blocco Flight Control System

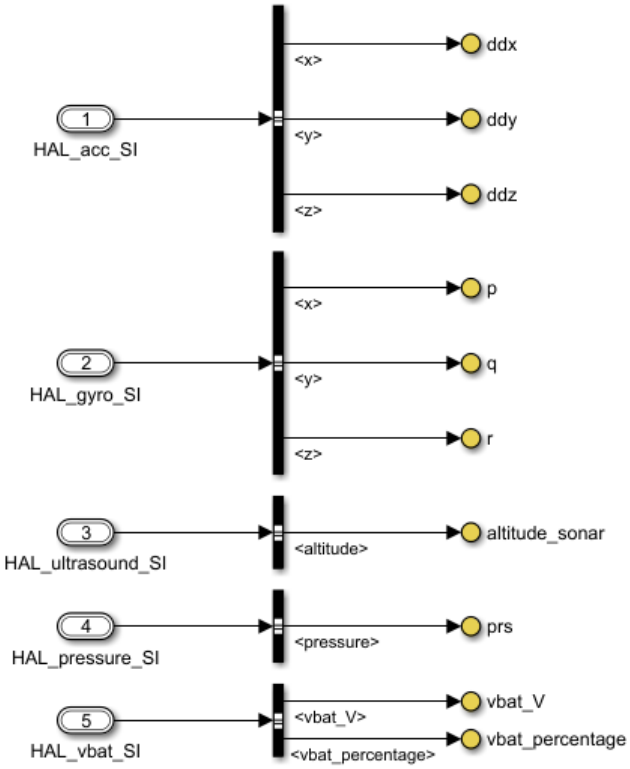


Figura 2.18: Blocco sensordata_group

landing logic : blocco che gestisce la logica durante la fase di atterraggio del drone, a partire da informazioni in merito allo stato attuale e dai dati sulle immagini processate dalla telecamera verticale. Consente di ottenere in uscita comandi interpretabili dal blocco *Controller* per la regolazione del *Thrust* esercitato dai motori. La struttura è mostrata nella figura 2.19.

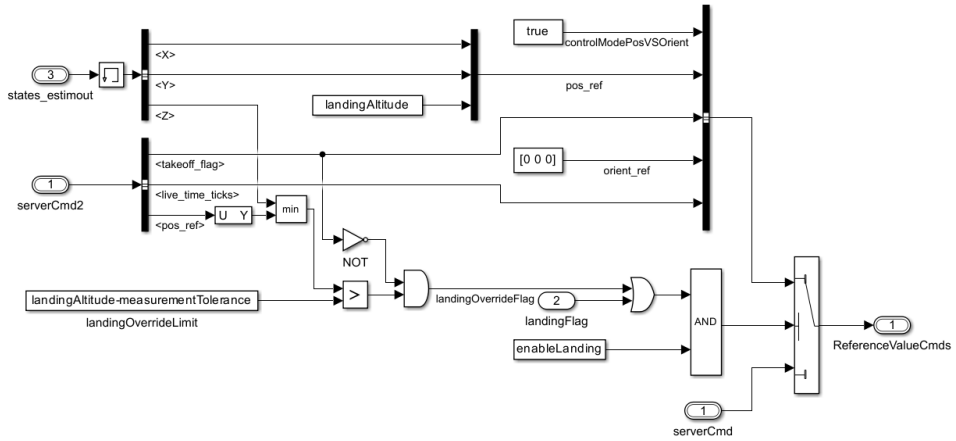


Figura 2.19: Blocco landing_logic

estimator : blocco responsabile della stima dello stato complessivo del drone durante il volo, a partire dalle informazioni provenienti dai sensori. Le stime in uscita riguardano posizione, velocità lineare ed angolare e rotazioni del drone rispetto il *Body Frame*. Per il calcolo degli stati stimati viene utilizzato un sistema di filtri composto da *filtri di Kalman* e un filtro complementare posto dopo la fase di pre-processamento dati dei sensori (blocco *SensorPreprocessing*). L'uscita verrà poi utilizzata anche dai blocchi *Controller* e *Crash Predictor Flags* rispettivamente per il calcolo dell'*errore d'inseguimento* e per la prevenzione di situazioni di pericolo. La struttura è mostrata nella figura 2.20.

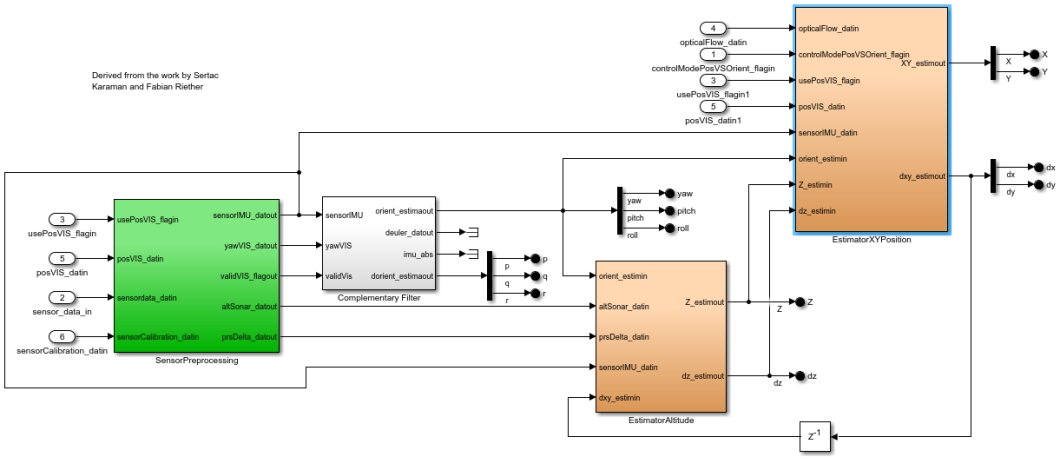


Figura 2.20: Blocco estimator - State Estimator

Osservando la figura 2.20 si nota appunto la presenza di un filtro complementare nella parte sinistra, mentre i *filtri di Kalman* si trovano all'interno dei blocchi *EstimatorAltitude* e *EstimetorXYPosition*.

controller : blocco che produce i comandi da inviare agli attuatori (in figura 2.17 indicati con *motorCmds*), il cosiddetto *sforzo di controllo*. Il calcolo viene eseguito a partire dall'*errore di inseguimento* dato dalla differenza tra i riferimenti impartiti, provenienti dal blocco *Command* e la stima dello stato attuale, proveniente dal blocco *Estimator*. Tale errore verrà poi dato come ingresso ai controllori veri e propri (PID) per i 6 gradi di libertà. La struttura è mostrata nella figura 2.21.

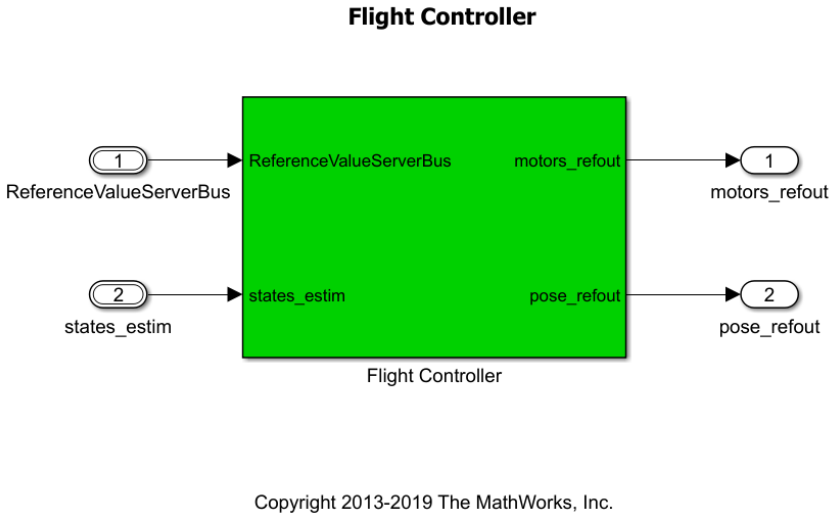


Figura 2.21: Blocco controller

La struttura del blocco *Flight Controller*, in verde nella figura 2.21, verrà successivamente analizzata nel dettaglio in quanto è la parte del modello che più ci interessa per lo sviluppo dei controllori lineari.

Crash Predictor Flags : blocco che, come anticipato, contiene un algoritmo che consente di predire situazioni indesiderate in base alle informazioni sullo stato stimato del sistema. In base al valore che assume l'uscita *flag*, calcolata dal blocco stesso, è possibile continuare il volo o interrompere immediatamente la spinta dei motori. La struttura è mostrata nella figura 2.22.

Logging : blocco che consente di raccogliere tutti i dati in merito allo stato, ai sensori, ai riferimenti e ai comandi dati ai motori durante il volo. La raccolta avviene tramite i blocchi *To Workspace*, che permettono la creazione direttamente nel file system del *Parrot Mambo* di MAT DATA file con all'interno tutti le informazioni e i dati sopracitati, relativi al volo eseguito. In questo modo è così possibile eseguire degli studi direttamente sui voli reali. La struttura è mostrata nella figura 2.23.

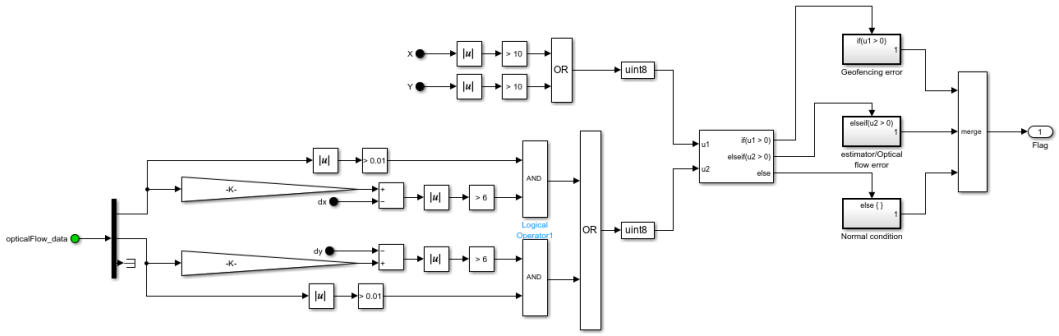


Figura 2.22: Blocco Crash Predictor Flags

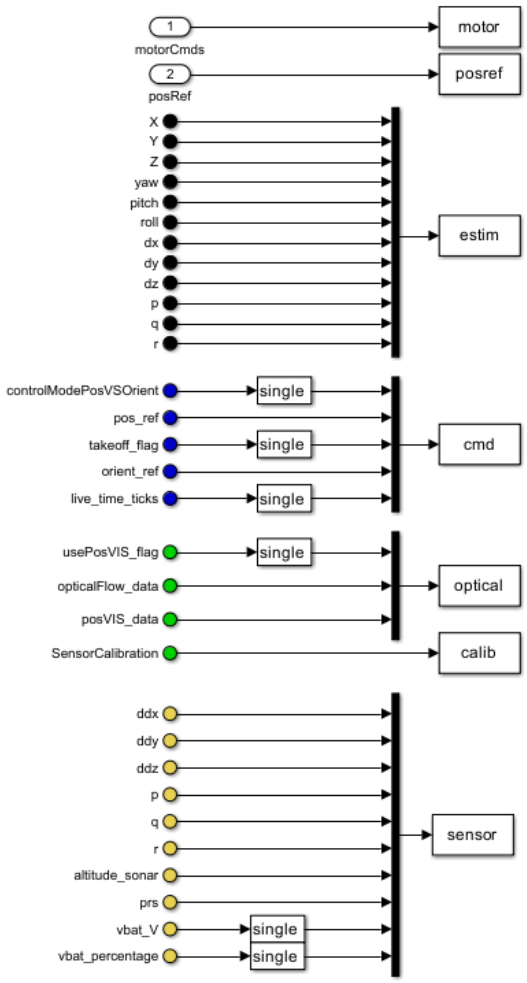


Figura 2.23: Blocco Logging

Tra i blocchi sopra descritti, prendiamo ora in considerazione il blocco *controller*, in figura 2.21. Esso è composto a sua volta del blocco *Flight Controller* in cui sono effettivamente contenuti i sotto-sistemi che implementano i controllori PID dei 6 gradi di libertà, oltre che a sotto-blocchi ausiliari necessari per la conversione dello sforzo di controllo in segnali interpretabili dai motori.

2.4.1 Blocco *Flight Controller*

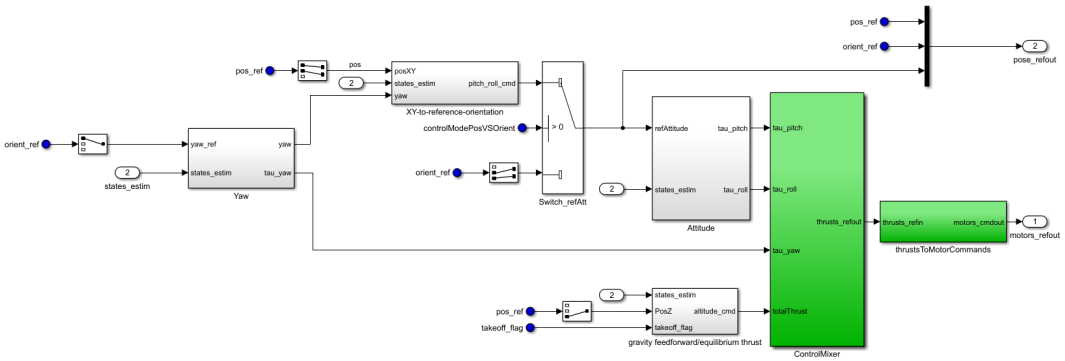


Figura 2.24: Blocco Flight Controller

I blocchi in verde della figura 2.24 sono quelli utili alla conversione dei segnali, cioè il blocco *ControlMixer* e *thrustsToMotorCommands*.

- Blocco *ControlMixer*: blocco che, tramite l'utilizzo di un blocco *Mux*, consente di raggruppare in un unico vettore tutte le azioni di controllo prodotte dai controllori dei gradi di libertà. Tale vettore verrà poi moltiplicato per la matrice *Controller.Q2Ts*, caricata nel *Workspace* all'avvio del progetto, che consente di pesare con i giusti coefficienti tutte le azioni di controllo tenendo conto anche del segno in base al verso dei vari momenti torcenti generati. L'uscita sarà quindi il vettore *thrusts_refout*, che andrà poi in ingresso al secondo blocco ausiliario per l'effettiva conversione in segnali per i motori.

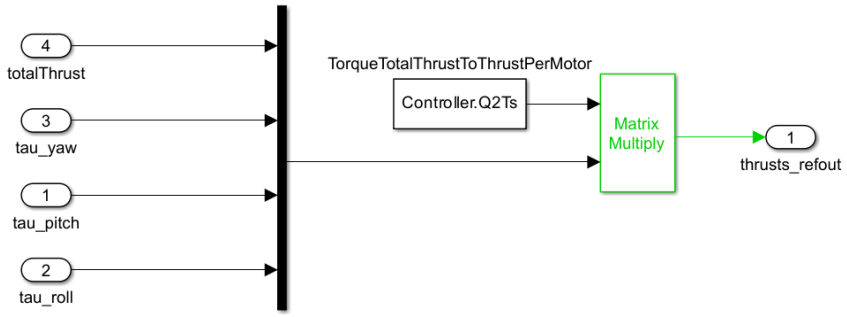


Figura 2.25: Blocco ControlMixer

- Blocco *thrustsToMotorCommands*: blocco che dato in ingresso il *Thrust* da generare per ogni motore, converte tale informazione in un segnale interpretabile

dei motori (*motors_cmdout*), che si tramuta in velocità di rotazione delle eliche. Per fare ciò utilizza il guadagno *Vehicle.Motor.thrustToMotorCommand* (presente nel *Workspace* MATLAB del progetto), seguito da un saturatore per evitare di inviare segnali fuori del range ammissibile per i motori e un secondo guadano *MotorDirections* utile per pesare i segnali con i relativi segni in base al verso di rotazione dei motori.

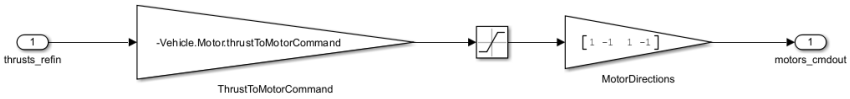


Figura 2.26: Blocco thrustsToMotorCommands

I restanti blocchi della figura 2.24 sono quelli di maggior interesse, in quanto al loro interno è presente l'implementazione degli algoritmi di controllo del drone. Essi sono *Yaw, XY-to-reference-orientation, Attitude* e *gravity feedforward/equilibrium thrust*.

Di seguito riportiamo il dettaglio di questi blocchi controllori.

2.4.2 Blocco *Yaw*

La logica del controllo di ogni grado di libertà si basa sull'utilizzo di controllori *PID* (*Proportional Integral Derivative*), particolari sistemi di regolazione di natura non lineare. Nel caso del controllo di rotazioni attorno l'asse di imbardata, appunto l'angolo di *yaw*, la struttura del controllore è mostrata nella figura 2.27.

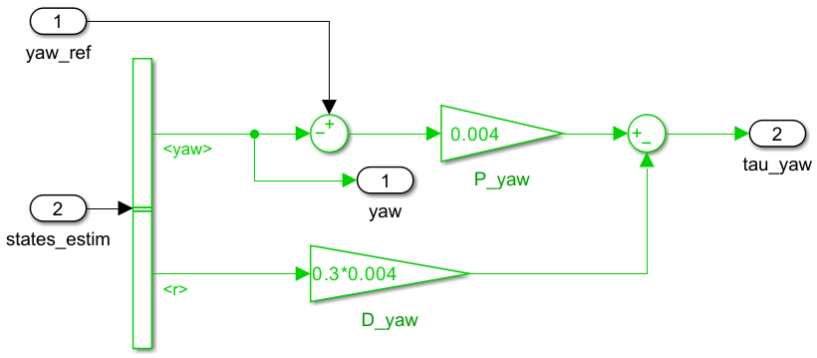


Figura 2.27: Blocco Yaw

É facile notare come in realtà la struttura del controllore è del tipo *PD* (*Proportional Derivative*), essendo presenti una parte proporzionale, una derivativa e nessuna integrativa. In generale, gli ingressi al blocco sono due, il riferimento per l'angolo *yaw* (*yaw_ref*) e la stima dello stato del sistema (*states_estim*), da cui tramite un blocco *Bus Selector* vengono selezionate solamente le componenti dello stato relative

all'angolo di *yaw* e alla velocità angolare r attorno l'asse di imbardata. Una volta calcolato l'*errore di inseguimento* dato dalla differenza tra *yaw_ref* e l'angolo *yaw* stimato attuale, esso andrà in ingresso alla parte proporzionale del controllore, mentre la velocità angolare dello *yaw* r andrà, da sola, in ingresso alla parte derivativa. La differenza tra le uscite delle due parti del controllore (uscita parte proporzionale − uscita parte derivativa) da luogo al segnale di controllo *tau_yaw*.

2.4.3 Blocco *XY-to-reference-orientation*

Il blocco seguente ha lo scopo di controllare la dinamica del drone per spostamenti lungo le direzioni x ed y del *Body Frame*. In particolare, per il controllo di tali coordinate, il controllore implementato è ancora una volta della tipologia *PD* essendo presenti solamente una parte proporzionale ed una derivativa, come riportato in figura 2.28.

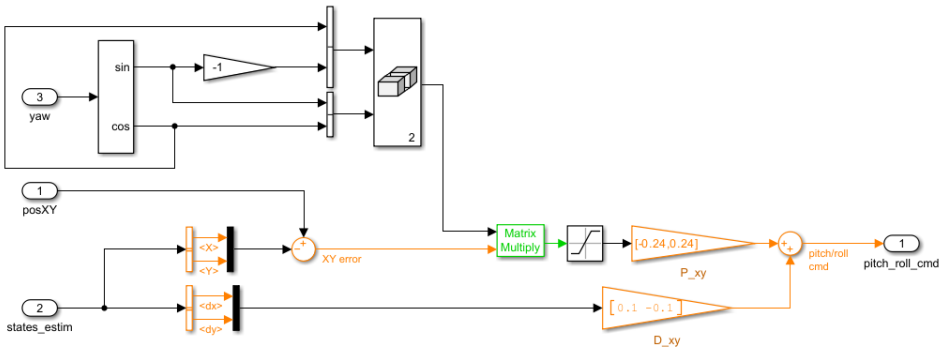


Figura 2.28: Blocco XY-to-reference-orientation

Per raggiungere una determinata posizione di riferimento X, Y nello spazio tramite traslazioni trasversali e longitudinali, cioè nel piano xy, è necessario anche considerare lo stato attuale dell'angolo di *yaw* in modo tale da poter determinare le giusta azione di controllo per i relativi motori in base all'orientamento rispetto l'asse di imbardata e alla posizione di riferimento nello spazio da raggiungere. Per questo motivo in figura 2.28 sono presenti, nella parte superiore sinistra, dei blocchi aggiuntivi proprio per considerare quanto appena descritto e tenerne poi conto nella fase di progettazione dell'azione di controllo da parte del regolatore proporzionale. Infatti, l'ingresso della parte P sarà una combinazione tra l'*errore di inseguimento* delle coordinate X, Y e lo stato attuale dello *yaw*, ottenuto tramite i blocchi *Matrix Concatenate* e *Matrix Multiply*.

La parte derivativa D agisce invece sulle componenti delle velocità lineari dx e dy nel piano xy.

La somma delle uscite delle parti P e D sarà il segnale di comando *pitch_roll_cmd* il quale andrà poi in ingresso al blocco *Attitude* (nel caso in cui sia verificata la condizione presente nel blocco *Switch* in figura 2.24) per il controllo dell'assetto durante lo spostamento.

NOTA: L'uscita *pitch_roll_cmd* del presente blocco, come si nota dalla figura 2.24, può essere indirizzata in due diversi modi in base alla condizione presente nel blocco *Switch*. L'indirizzamento è regolato dal valore della variabile booleana *controlModelPosVSOrient*, in particolare se:

- $controlModelPosVSOrient > 0$, ovvero $controlModelPosVSOrient = true$, l'uscita *pitch_roll_cmd* verrà indirizzata, tramite il *bus di sistema*, verso il blocco *Attitude* per il controllo dell'assetto mediante gli angoli di *pitch* e *roll*.
- $controlModelPosVSOrient \leq 0$, ovvero $controlModelPosVSOrient = false$, l'uscita non verrà inviata sul *bus di sistema* e di conseguenza nessun blocco la riceverà. I riferimenti per *pitch* e *roll* (da cui dipendono anche le traslazioni nel piano xy) saranno quelli impartiti dal blocco *Command* in modo totalmente indipendente dalla posizione di riferimento (X, Y).

Questo tipo di condizione è dovuta alla natura stessa del nostro sistema, in quanto esso risulta essere di tipo *sotto-attuato*, ovvero il controllo dei movimenti in xy avviene tramite gli angoli di *pitch* e *roll*. Si veda la sottosezione 1.3.2 per maggiori dettagli sui movimenti di un quadricottero durante il volo.

2.4.4 Blocco *Attitude*

Il blocco *Attitude* implementa il controllore per il mantenimento di un assetto stabile durante il volo del drone (*condizione di Hovering*). Questo è possibile andando ad agire sui valori assunti dagli angoli di *pitch* e *roll*, in modo tale da renderli il più possibile prossimi al valore nullo. La struttura del controllore utilizzato è della tipologia PID, dove rispetto i casi precedenti abbiamo anche la presenza di una parte integrativa. In figura2.29 è riportata la sua struttura.

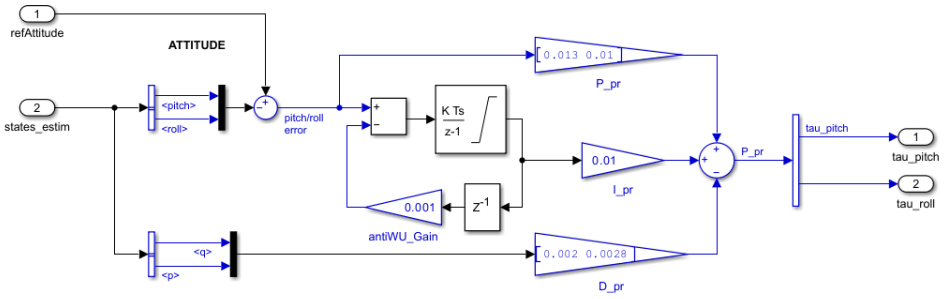


Figura 2.29: Blocco Attitude

Il ragionamento seguito è molto simile ai casi precedenti (si veda 2.4.2 e 2.4.3), con una piccola nota da aggiungere in merito alla parte integrativa I. Dalla figura 2.29 si può notare la presenza di un sistema di *feedback* sull'ingresso della parte I. Questo è necessario per evitare problemi legati al fenomeno dell'*integral wind-up*, dovuto alla saturazione degli attuatori in condizioni critiche per il drone, ad esempio a causa

di raffiche di vento verticali o di pesi troppo elevati da sollevare/trasportare. Il controllo del quadricottero in volo con attuatori saturi diventerebbe molto complesso e spesso impossibile, di conseguenza per ovviare tale problematica viene utilizzato un guadagno *antiWU_Gain* in *feedback* negativo sull'uscita del blocco *Discrete-Time Integrator*, che viene quindi sottratto all'*errore di inseguimento 'pitch/roll error'*.

2.4.5 Blocco *gravity feedforward/equilibrium thrust*

All'interno del blocco *gravity feedforward/equilibrium thrust* è contenuta la logica di controllo per l'ultimo grado di libertà non ancora analizzato, cioè la coordinata z o altitudine. Il regolatore è della tipologia PD, come nel caso dello *yaw* e del controllo dei movimenti nel piano xy. L a struttura segue quindi gli stessi schemi precedentemente analizzati nelle sottosezioni 2.4.2 e 2.4.3, come osservabile nella figura 2.30.

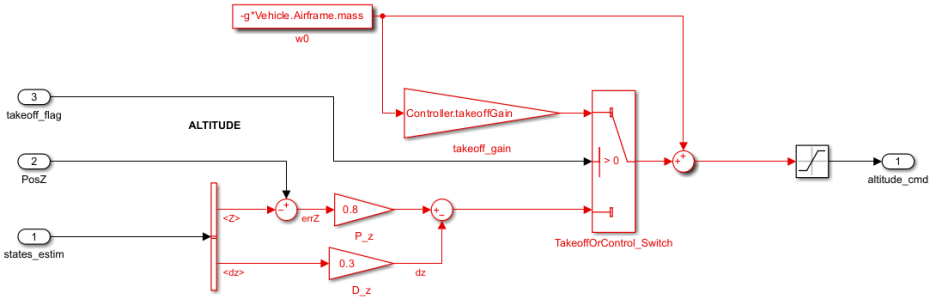


Figura 2.30: Blocco gravityfeedforward/equilibrium thrust

L'unico accorgimento da fare risiede nel blocco *Switch* presente nella parte destra della figura 2.30 in cui viene eseguito un controllo sulla variabile booleana di ingresso *takeoff_flag*. In base al valore assunto da tale condizione l'azione di controllo viene gestita nel seguente modo:

- $takeoff\_flag > 0$, ovvero $takeoff\_flag = true$, l'azione di controllo viene gestita da un guadagno *takeoff_gain*, presente nel Workspace del progetto, moltiplicato per la forza peso del drone. Questo vale solamente per il primo secondo di volo (in cui $takeoff\_flag = true$) al fine ottenere, tramite il guadagno utilizzato, un decollo rapido.
- $takeoff\_flag \leq 0$, ovvero $takeoff\_flag = false$, l'azione di controllo viene gestita dal controllore PD. Questo avviene successivamente alla fase di decollo del drone (primo secondo di volo), in cui di conseguenza la variabile di controllo $takeoff\_flag$ viene settata a $false$.

In entrambi i casi viene comunque prodotto in uscita uno sforzo di controllo *altitude_cmd*, in cui viene comunque considerato anche il contributo della forza peso del drone *g*Veihicle.Airframe.mass*, con lo scopo di ottenere una spinta ascensionale in ogni caso maggiore o, al massimo, uguale.

2.5 Blocco *Command*

Il blocco *Command* è di fondamentale importanza in quanto è grazie ad esso che risulta possibile generare ed inviare segnali al sistema di controllo (FCS) per permettere al quadricottero di seguire la traiettoria di riferimento desiderata durante il volo. Dal modello complessivo (figura 2.2) entrando nel blocco *Command*, la struttura interna risulta suddivisa in ulteriori quattro sotto-parti, come mostrato in figura 2.31.

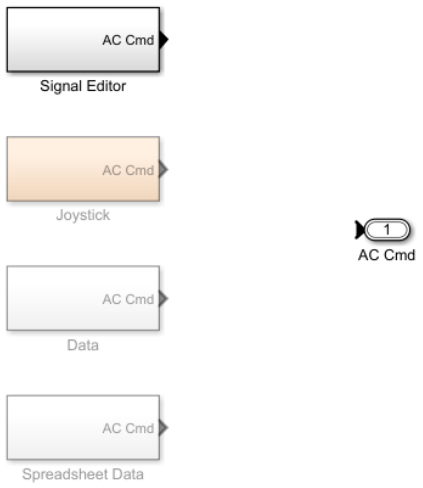


Figura 2.31: Blocco Command

I quattro blocchi della figura 2.31 rappresentano dei metodi alternativi per la generazione dei segnali di riferimento. Il settaggio del generatore da utilizzare avviene tramite la variabile d'ambiente *VSS_COMMAND*, presente all'interno del Workspace del progetto. Vediamo ora in che modo poter scegliere tra la diverse tipologie di generazione del segnale.

- *VSS_COMMAND = 0* ⇒ **Signal Editor**: la generazione dei segnali avviene attraverso il blocco predefinito *Signal Editor*.
- *VSS_COMMAND = 1* ⇒ **Joystik**: la generazione dei segnali avviene tramite l'utilizzo di un Joystik, non escludendo la possibilità anche in questo caso di utilizzare un blocco *Signal Editor*.
- *VSS_COMMAND = 2* ⇒ **MATLAB Data**: la generazione dei segnali avviene tramite dati, in merito ai riferimenti desiderati, memorizzati all'interno di MATLAB Data file.
- *VSS_COMMAND = 3* ⇒ **Spreedsheet Data**: la generazione dei segnali avviene, come nel caso precedente, tramite dati memorizzati questa volta all'interno di fogli di calcolo, come ad esempio *Excel* (file .xlsx).

Il valore di default è settato a 0, quindi sul generatore *Signal Editor*.

Per il presente elaborato il sistema di generazione utilizzato è tramite il blocco predefinito *Signal Editor*, di cui andiamo ora ad analizzare le caratteristiche interne più nel dettaglio (la trattazione degli altri sistemi di generazione segnali viene tralasciata in quanto non necessaria).

2.5.1 Blocco *Signal Editor*

La struttura interna è mostrata all'interno della figura 2.32 in cui si può notare la presenza del blocco MATLAB predefinito *Signal Editor*, indicato con l'etichetta *Position/Attitude Reference*, dove avviene l'effettiva generazione dei riferimenti.

Nel caso di un quadricottero, come già noto dalla sezione 1.3.2, i gradi di libertà (*D.o.F.*) sono in totale sei, suddivisi in tre per la posizione (X, Y, Z) nello spazio (*Position Reference*) e tre per gli angoli di inclinazione attorno gli assi del *Body Frame*, i rispettivi angoli di *yaw, pitch* e *roll* (*Attitude Reference*). Di conseguenza i segnali di riferimento da generare saranno appunto sei, pari cioè al numero di *D.o.F.*

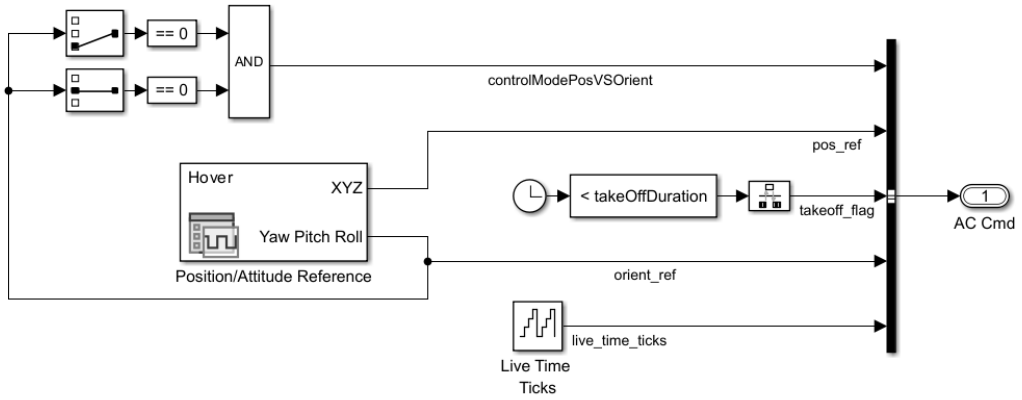


Figura 2.32: Blocco Signal Editor

L'uscita del blocco *AC Cmd* consente di immettere nel *bus di sistema* le varie informazioni necessarie alla successiva elaborazione da parte del *Flight Control System* (FCS). Essa viene composta a partire dal blocco *Bus Creator* il quale combina un insieme di cinque elementi di input in un *bus*, dove tali ingressi (input) sono rappresentati dalle seguenti variabili:

1. *controlModelPosVSOrient*: variabile booleana responsabile di monitorare l'assetto (*Hovering*) durante gli spostamenti in volo. Essa risulta essere settata a *true* solamente nel caso in cui lo stato attuale degli angoli di *pitch* e *roll* risulti essere nullo per entrambi, ovvero la condizione di *Hovering* (vedi nota sotto-sezione 2.4.3).

2. *pos_ref*: variabile contenente i riferimenti per i tre *D.o.F.* X, Y, Z.

3. *orient_ref*: variabile contenente i riferimenti per i tre *D.o.F. roll, pitch* e *yaw*.

4. *takeoff_flag*: variabile booleana responsabile di monitorare la fase di decollo del drone. In particolare, essa risulta essere settata a *true* fin quando il clock di sistema risulta essere inferiore della variabile d'ambiente *takeOffDuration*, di default caricata nel Workspace con in valore di 1 secondo. Il suo scopo è descritto nella sotto-sezione 2.4.5.

5. *live_time_ticks*: segnale generato dal blocco *Counter Free-Running* che emette una serie crescenti di valori ad una data frequenza data dal tempo di campionamento T_s e fino ad un limite massimo dato dal numero di *bit* a disposizione per la rappresentazione. Nel caso specifico abbiamo $T_s = 0.005s$, con rappresentazione a $32bit$. Una volta raggiunto il limite superiore il contatore riparte da 0.

NOTA: Per semplicità di utilizzo in fase di progettazione dei regolatori lineari, il blocco *Position/Attitude Reference*, della tipologia *Signal Editor*, verrà sostituito con un blocco *Signal Builder* predefinito all'interno dell'ambiente *Simulink* (Figura 2.33).

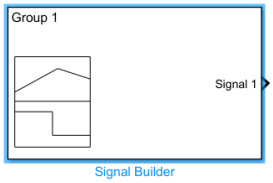


Figura 2.33: Blocco illustrativo Signal Builder

2.6 Blocco *Visualization*

Il blocco *Visualization* consente di eseguire analisi sulla maggior parte dei dati in merito allo stato del sistema, ai riferimenti imposti e ai comandi inviati ai motori durante tutta la durata del volo, ricevendo informazioni rispettivamente dai blocchi *Airframe, Command* e *FCS*. Navigando al suo interno esso risulta essere a sua volta suddiviso in due macro-blocchi *Extract Flight Instruments* e *Visualization*. Tralasciando il contenuto del primo, soffermiamo l'attenzione sul secondo blocco che risulta essere quello di nostro interesse, la cui struttura interna è mostrata nella figura 2.34.

Oltre alle tre variabili di ingresso *States, Commands* e *Actuators*, il blocco risulta essere suddiviso in 4 sotto-parti in base alla modalità di visualizzazione desiderata. La scelta avviene tramite il settaggio della variabile d'ambiente *VSS_VISUALIZATION*, caricata nel Workspace all'apertura del progetto. Vediamo i criteri di scelta.

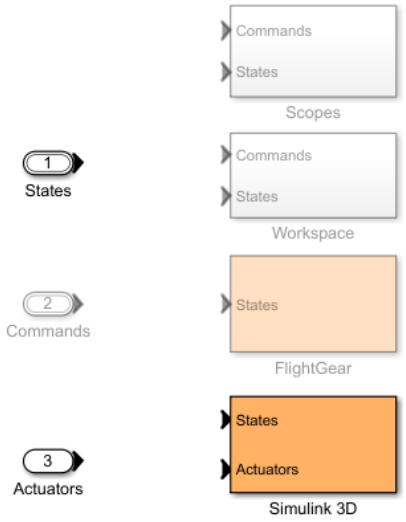


Figura 2.34: Blocco Visualization

- *VSS_VISUALIZATION = 0* ⇒ **Scopes**: i dati in ingresso vengono visualizzati tramite degli oscilloscopi virtuali, ottenuti mediante l'utilizzo di blocchi *Scope*.
- *VSS_VISUALIZATION = 1* ⇒ **Workspace**: i dati in ingresso vengono memorizzati all'interno di strutture dati (*.mat*) salvate direttamente all'interno del Workspace del progetto. Il salvataggio avviene tramite l'utilizzo di blocchi *To Workspace*.
- *VSS_VISUALIZATION = 2* ⇒ **FlightGear**: la visualizzazione dei dati avviene attraverso un modello di simulazione 3D interno al Toolbox *Aerospace Blockset*.
- *VSS_VISUALIZATION = 3* ⇒ **Simulink 3D**: la visualizzazione dei dati avviene attraverso una simulazione 3D del volo, ottenuta tramite l'elaborazione dei dati da parte del blocco *VR Sink* della libreria interna al Toolbox *Simulink 3D Animation*. Un esempio di simulazione di volo è mostrato nella figura 2.3.

Il valore di default è settato al valore 3, cioè visualizzazione mediante *Simulink 3D*.

NOTA: Per la visualizzazione e l'analisi dei dati ottenuti in fase di progettazione dei controllori verrà utilizzato il blocco *Scopes* (figura 2.35), di conseguenza la variabile d'ambiente *VSS_VISUALIZATION* verrà modificata e impostata sul valore 0.

Inoltre, alla struttura del blocco visualizzata in figura 2.35 verranno applicate delle modifiche per consentire di poter visualizzare solamente i dati di nostro interesse, ovvero i riferimenti (tramite la variabile d'ingresso *Commands*) e la relativa stima sui valori dei 6 *D.o.F.*, estraibile dalla variabile *States*.

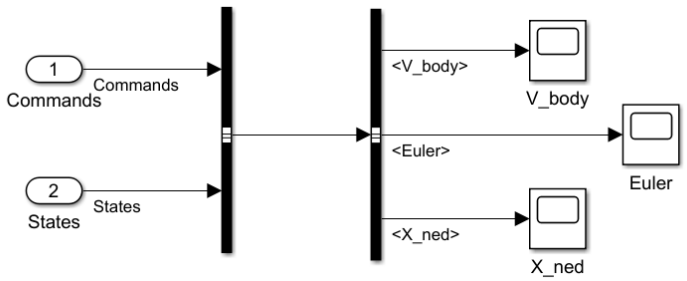
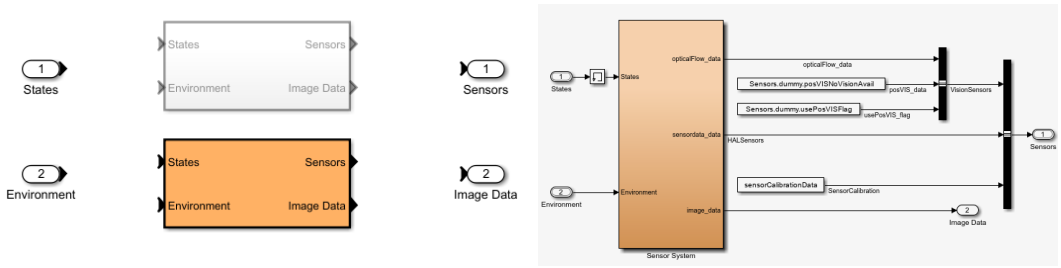


Figura 2.35: Blocco Scopes

Per i restanti blocchi verranno solamente riportate immagini relative alla struttura interna (senza entrare nel dettaglio dei vari sotto-sistemi presenti) con brevi descrizioni, non essendo di fondamentale importanza all'interno del presente elaborato.

2.7 Blocco *Sensors*

Il blocco *Sensors* (Figura 2.36) è responsabile della gestione e processamento dei dati provenienti da tutti i sensori montati a bordo del drone, comprese le informazioni in merito alla camera verticale.



(a) Struttura generale (b) Struttura sotto-sistema interno

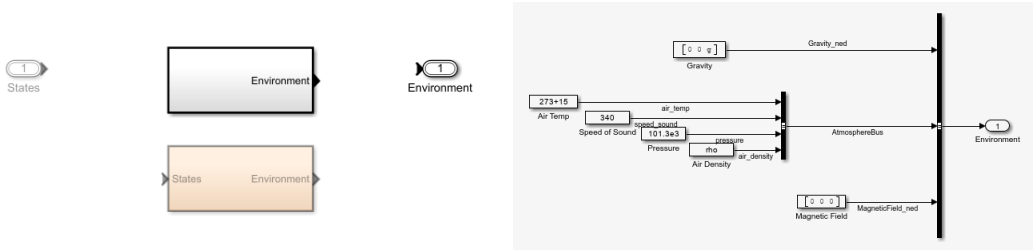
Figura 2.36: Blocco Sensors

Dalla figura 2.36a notiamo la presenza di due macro-blocchi tramite cui è possibile scegliere la natura dei sensori (*feddthrought sensors, dynamic sensors*) e la presenza o meno di "rumore" (disturbi) nelle misurazioni. La scelta avviene tramite il settaggio della variabile d'ambiente *VSS_SENSORS*, di default impostata sul valore 1, ovvero *dynamic sensors* con "rumore".

Nella figura 2.36b è mostrata la struttura più interna del blocco precedente, in cui è presente un blocco *Sensors System* responsabile della logica di lettura e comunicazione dati, messi in circolazione sul *bus di sistema* attraverso le variabili *Sensors* e *Image Data*.

2.8 Blocco *Environment*

Il blocco *Environment* (Figura 2.37) è fondamentale per il corretto funzionamento del modello di simulazione in quanto consente di simulare l'ambiente di volo di un quadricottero. Tramite il parametro d'ambiente *VSS_ENVIRONMENT* è possibile decidere se utilizzare delle variabili d'ambiente indipendenti o meno della posizione del drone. Di default la variabile è settata sul valore 0 corrispondente a variabili d'ambiente costanti cioè indipendenti dalla posizione nello spazio.



(a) Struttura generale (b) Struttura sotto-sistema interno

Figura 2.37: Blocco Environment

Dalla figura 2.37b notiamo appunto il settaggio delle variabili in merito alla forza di gravità, parametri atmosferici e campo magnetico terrestre. A differenza dei primi due, quest'ultima variabile d'ambiente viene trascurata nel modello utilizzato, osservando infatti il settaggio a 0 per tutte e tre le sue componenti.

2.9 Blocco *Instrument Panel*

Il blocco *Instrument Panel* permette la visualizzazione dei principali dati di volo, come ad esempio altitudine, velocità di rotazione dei motori, orientamento nello spazio, velocità verticale e traslazionale. Esso è costituito da una serie di blocchi predefiniti del Toolbox *Aerospace Blockset*, che consentono la rappresentazione dei dati sopracitati. La sua interfaccia grafica, in figura 2.38, ha una struttura molto simile ad una plancia aerea.

Figura 2.38: Blocco Instrument Panel

2.10 Blocco *Stop Simulation*

Il blocco *Stop Simulation* (Figura 2.39) funge da sistema di controllo per evitare potenziali situazioni dannose per il drone stesso e per l'ambiente circostante. La sua attivazione dipende dalle decisioni prese dal *Flight Control System* (FCS) sulla base delle informazioni provenienti dai sensori. Il blocco *Flag has been triggered* sta ad indicare che, una volta verificata la condizione $flag = true$ con la conseguente interruzione dei motori, la variabile viene poi settata nuovamente al suo valore $false$ di default, tramite il blocco *Logical Operator - NOT*.

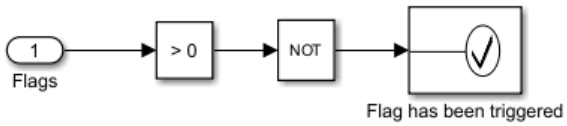


Figura 2.39: Blocco Stop Simulation

Capitolo 3

Rappresentazione in *spazio di stato* della dinamica di un quadricottero

Nel seguente capitolo si andranno ad analizzare le fasi di linearizzazione del modello matematico di un quadricottero, che per sua natura è di tipo non lineare. Quest'ultimo è quello che più aderisce alla realtà ma la sua struttura risulta essere molto complessa è di conseguenza non può essere preso come modello di riferimento per lo sviluppo del seguente elaborato. Per la successiva progettazione di regolatori lineari, argomento del capitolo 5, è quindi necessario lavorare su un modello di partenza appunto lineare, di cui vengono di seguito discusse le varie caratteristiche.

3.1 Processo di linearizzazione del modello non lineare

Partiamo dal considerare la dinamica di un quadricottero come un sistema $MIMO$ non lineare e stazionario, cioè indipendente dal tempo. Le equazioni del sistema 3.1 descrivono in linea generale il *"movimento" nello stato* $\dot{x}(t)$ e l'uscita $y(t)$.

$$\begin{cases} \dot{x}(t) = f(x(t), u(t)) \\ y(t) = g(x(t), u(t)) \end{cases} \tag{3.1}$$

La dinamica $\dot{x}(t)$ e l'uscita $y(t)$ dipendono, in generale, dallo stato del sistema $x(t)$ e dall'ingresso $u(t)$ tramite generiche funzioni non lineari f e g. Come si può notare dal sistema 3.1 la dipendenza dal tempo t non avviene in maniera diretta all'interno delle funzioni, ma essendo u e x dipendenti a loro volta dal tempo, allora anche $\dot{x}$ e y dipenderanno in maniera indiretta dal tempo t.

Per linearizzare tali funzioni non lineari si utilizzano gli sviluppi in serie di *Taylor* e di conseguenza è necessario andare a definire un *punto di lavoro* x_0 che rappresenti per il sistema un *punto di equilibrio*, riconducibile ad esempio allo stato di *hovering* durante il volo.

Un *punto di lavoro* per essere uno stato di equilibrio deve soddisfare le condizioni di equilibrio nominale rispetto l'ingresso costante $\bar{u}$ a partire dall'istante t_0 in poi, ovvero devono valere le seguenti proprietà:

$$u(t) = \bar{u} \quad \forall t \geq t_0 \tag{3.2}$$

$$x(t) = \bar{x} \quad \forall t \geq t_0 \tag{3.3}$$

$$\dot{x}(t) = f(x(t), u(t)) = f(\bar{x}, \bar{u}) = 0 \quad \forall t \geq t_0 \tag{3.4}$$

$$y(t) = g(x(t), u(t)) = g(\bar{x}, \bar{u}) = \bar{y} \quad \forall t \geq t_0 \tag{3.5}$$

Poiché lo sviluppo di Taylor di una funzione in un punto, se esiste, permette di esprimere la funzione nell'intorno del punto come un polinomio lineare con infiniti termini (approssimazione *locale*), di conseguenza andiamo a considerare delle piccole oscillazioni intorno al punto di lavoro $(\bar{x}, \bar{u})$ precedentemente definito.

$$x(t) = \bar{x} \pm \delta x(t) \tag{3.6}$$

$$u(t) = \bar{u} \pm \delta u(t) \tag{3.7}$$

$$y(t) = \bar{y} \pm \delta y(t) \tag{3.8}$$

In questo modo è possibile riscrivere il sistema 3.1 considerando solamente un intorno locale del punto di lavoro all'interno del quale andremo a linearizzare le funzioni f e g mediante lo sviluppo in serie di Taylor.

$$\begin{cases} \dot{x}(t) = \dot{\bar{x}} + \delta\dot{x}(t) = f(x(t), u(t)) = f(\bar{x} + \delta x(t), \bar{u} + \delta u(t)) \\ y(t) = \bar{y} + \delta y(t) = g(x(t), u(t)) = g(\bar{x} + \delta x(t), \bar{u} + \delta u(t)) \end{cases} \tag{3.9}$$

Dall'equazione 3.3 osserviamo come lo stato $x(t)$ risulta essere pari al valore costante $\bar{x}$ per ogni $t \geq t_0$ e di conseguenza la sua dinamica $\dot{\bar{x}}$ presente nella prima equazione del sistema 3.9 risulta essere nulla, potendo quindi semplificare quest'ultima andando ad eliminare tale termine.

Fatte tali considerazioni preliminari, sono ora soddisfatte tutte le ipotesi necessarie all'applicazione degli sviluppi in serie di Taylor e di conseguenza, supponendo le funzioni f e g sufficientemente regolari, possiamo sviluppare le equazioni del sistema 3.9 rispetto le variabili x ed u utilizzando l'approssimazione al primo ordine della serie e trascurando i termini di grado superiore.

$$\begin{cases} \delta\dot{x}(t) = f(\bar{x}, \bar{u}) + \left.\dfrac{\partial f(x,u)}{\partial x}\right|_{x=\bar{x},u=\bar{u}} \delta x(t) + \left.\dfrac{\partial f(x,u)}{\partial u}\right|_{x=\bar{x},u=\bar{u}} \delta u(t) \\ \bar{y} + \delta y(t) = g(\bar{x}, \bar{u}) + \left.\dfrac{\partial g(x,u)}{\partial x}\right|_{x=\bar{x},u=\bar{u}} \delta x(t) + \left.\dfrac{\partial g(x,u)}{\partial u}\right|_{x=\bar{x},u=\bar{u}} \delta u(t) \end{cases} \tag{3.10}$$

Sfruttando le ipotesi espresse nelle equazioni 3.4 e 3.5, cioè $\bar{x}$ è uno stato di

equilibrio per l'ingresso $u(t) = \bar{u}$, è possibile semplificare alcuni termini all'interno del sistema 3.10 ($f(\bar{x}, \bar{u}) = 0$, $g(\bar{x}, \bar{u}) = \bar{y}$). Il risultato ottenuto è il seguente:

$$\begin{cases} \delta\dot{x}(t) = \left.\dfrac{\partial f(x,u)}{\partial x}\right|_{x=\bar{x},u=\bar{u}} \delta x(t) + \left.\dfrac{\partial f(x,u)}{\partial u}\right|_{x=\bar{x},u=\bar{u}} \delta u(t) \\ \delta y(t) = \left.\dfrac{\partial g(x,u)}{\partial x}\right|_{x=\bar{x},u=\bar{u}} \delta x(t) + \left.\dfrac{\partial g(x,u)}{\partial u}\right|_{x=\bar{x},u=\bar{u}} \delta u(t) \end{cases} \tag{3.11}$$

Il sistema ottenuto risulta essere una versione linearizzata del sistema di partenza 3.1 di natura non lineare. Questo risultato è però applicabile solamente in *forma locale* in quanto l'approssimazione mediante l'utilizzo delle serie di Taylor ha validità solamente nell'intorno di raggio δ del punto di lavoro considerato, nel nostro caso lo stato $x(t) = \bar{x}$ a cui il sistema è giunto grazie all'ingresso costante $u(t) = \bar{u}$ il quale produce un'uscita anch'essa costante $y(t) = \bar{y}$.

Ricordando quanto già discusso nella sezione 2.3.2 ed in particolare soffermando l'attenzione sul sistema di equazioni 2.1, possiamo esprimere il sistema 3.11 come un sistema in *spazio di stato* di seguito riportato:

$$\begin{cases} \delta\dot{x}(t) = A\delta x(t) + B\delta u(t) \\ \delta y(t) = C\delta x(t) + D\delta u(t) \end{cases} \tag{3.12}$$

I coefficienti della combinazione lineare tra lo stato $\delta x(t)$ e l'ingresso $\delta u(t)$, sono appunto le matrici A, B, C, D di un generico sistema $MIMO$ espresso mediante rappresentazione in *spazio di stato*. Nel caso specifico, i valori di tali matrici coincidono con quelli presenti all'interno della *matrice Jacobiana* $J_h(x, u)$ associata alle funzioni f e g del sistema 3.1 (considerando le derivate calcolate nel punto di lavoro scelto).

$$J_h(x,u) = \begin{pmatrix} \dfrac{\partial f(x,u)}{\partial x} & \dfrac{\partial f(x,u)}{\partial u} \\ \dfrac{\partial g(x,u)}{\partial x} & \dfrac{\partial g(x,u)}{\partial u} \end{pmatrix} = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \in \mathbb{R}^{2\times 2} \tag{3.13}$$

Combinando la 3.12 e la 3.13 e considerando le derivate nella matrice *Jacobiana* calcolate nel punto di lavoro ($x = \bar{x}, u = \bar{u}$) si ottiene il sistema linearizzato 3.11.

3.2 Sistema in *spazio di stato* del *Parrot Mambo minidrone*

Come già affrontato nella sotto-sezione 2.3.2, all'interno del progetto *asbQuadcopter* di *Simulink*, oltre al modello reale della dinamica di un quadricottero è presente anche una versione linearizzata all'interno del blocco *Airframe - Linear Airframe*, utilizzabile settando la variabile d'ambiente *VSS_VEHICLE* sul valore 0. La struttura del blocco è visualizzabile nella figura 2.15, in cui si nota la classica "forma" di un sistema lineare espresso mediante rappresentazione in *spazio di stato*, ovvero nella forma 2.1. Per maggiori dettagli si faccia riferimento alla sotto-sezione 2.3.2, precedentemente citata.

Per la progettazione di regolatori lineari è quindi necessario utilizzare un modello lineare in *spazio di stato* (in generale $MIMO$) e di conseguenza bisogna essere a conoscenza delle relative variabili di stato, ingresso e uscita e dei valori delle matrici del sistema A, B, C, D. Per la dinamica di un quadricottero, tale modello seppur lineare risulta comunque avere una struttura abbastanza complessa, essendo presenti sia numerose variabili di ingresso che di uscita (sistema $MIMO$).

Nel caso del minidrone *Parrot Mambo* tutte queste informazioni sono già state calcolate e sono presenti all'interno del progetto *asbQuadcopter*. In particolare, dalla schermata principale del progetto è possibile accedere a tali informazioni navigando all'interno della cartella *linearAirframe* seguendo il seguente percorso: *"linearAirframe/linearizedAirframe.mat"*. Così facendo verrà caricata all'interno del Workspace del progetto la struttura dati *'linsys'* in cui sono presenti i valori delle variabili e delle matrici standard del sistema. Nelle figure 3.1, 3.2, 3.3, 3.4 è mostrata la struttura delle matrici A, B, C, D mentre nelle figure 3.5, 3.6, 3.7 sono riportate le variabili di stato, ingresso e uscita.

Le dimensioni dei vettori e delle matrici sono indicate nella parte superiore delle singole figure.

linsys_A | linsys_B | linsys_C | linsys_D

12x12 double

	1	2	3	4	5	6	7	8	9	10	11	12
1	0	3.6611e-17	0	0	0	0	0	0	0	1	3.0292e-23	-5.2296e-12
2	-3.6611e-17	0	0	0	0	0	0	0	0	0	1	5.2295e-12
3	-6.8347e-13	-2.0323e-28	0	0	0	0	0	0	0	0	-5.2295e-12	1.0000
4	0	-9.8100	0	-3.4402e-09	-3.4402e-09	6.8532e-13	0	0	0	-5.4616e-11	0.1380	-3.2848e-13
5	9.8100	-2.8478e-22	0	-3.6595e-09	-3.6595e-09	-6.8520e-13	0	0	0	-0.1380	5.8098e-11	-3.5159e-13
6	5.4456e-11	5.4456e-11	0	-6.8347e-13	6.8347e-13	0	0	0	0	3.2920e-13	3.5087e-13	0
7	1.7216e-24	1.3478e-19	3.2920e-13	1.0000	-1.4969e-18	-5.2296e-12	0	0	0	0	0	0
8	-1.3478e-19	2.0176e-37	3.5087e-13	1.4969e-18	1	5.2295e-12	0	0	0	0	0	0
9	-3.2920e-13	-3.5087e-13	0	5.2296e-12	-5.2295e-12	1.0000	0	0	0	0	0	0
10	0	0	0	-5.7590e-08	-5.7590e-08	-2.7287e-14	0	0	0	-2.1715	9.1429e-10	4.3301e-13
11	0	0	0	4.0369e-08	4.0369e-08	-2.1644e-14	0	0	0	6.4089e-10	-1.6192	-5.1363e-13
12	0	0	0	-1.4434e-19	-1.2831e-19	3.6185e-19	0	0	0	1.2157e-13	1.1242e-13	-6.8807e-09

Figura 3.1: Matrice della dinamica A

linsys_A | linsys_B | linsys_C | linsys_D

12x14 double

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

Figura 3.2: Matrice degli ingressi B

linsys_A linsys_B linsys_C linsys_D

33x12 double

	1	2	3	4	5	6	7	8	9	10	11	12
1	0	-9.8100	0	-3.4402e-09	-3.4402e-09	1.8445e-15	0	0	0	-5.4616e-11	0.1380	7.2109e-16
2	9.8100	-2.8478e-22	0	-3.6595e-09	-3.6595e-09	-1.7339e-15	0	0	0	-0.1380	5.8098e-11	-7.2109e-16
3	5.4456e-11	5.4456e-11	0	0	0	0	0	0	0	0	0	0
4	0	5.5511e-12	0	0	0	0	0	0	0	0	0	0
5	-5.2296e-12	-5.2295e-12	-1.0000	0	0	0	0	0	0	0	0	0
6	1.4969e-18	1.0000	-5.2295e-12	0	0	0	0	0	0	0	0	0
7	0	8.3096e-30	1.0000	0	0	0	0	0	0	0	0	0
8	5.5511e-12	0	0	0	0	0	0	0	0	0	0	0
9	-1.0000	1.4969e-18	-5.2296e-12	0	0	0	0	0	0	0	0	0
10	0	-1.0000	0	0	0	0	0	0	0	0	0	0
11	1.0000	-2.9029e-23	0	0	0	0	0	0	0	0	0	0
12	5.5511e-12	5.5511e-12	0	0	0	0	0	0	0	0	0	0
13	1	0	0	0	0	0	0	0	0	0	0	0
14	0	1	0	0	0	0	0	0	0	0	0	0
15	0	0	1	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	9.0026e-06	0	0	0	0	0
17	0	0	0	0	0	0	0	1.2127e-05	0	0	0	0
18	0	0	0	0	0	0	0	0	-1.0000	0	0	0
19	0	0	0	0	0	0	0	0	0	1	0	0
20	0	0	0	0	0	0	0	0	0	0	1	0
21	0	0	0	0	0	0	0	0	0	0	0	1
22	0	0	0	1.0000	0	0	0	0	0	0	0	0
23	0	0	0	0	1	0	0	0	0	0	0	0
24	0	0	0	0	0	1.0000	0	0	0	0	0	0
25	1.7216e-24	1.3478e-19	3.2920e-13	1.0000	-1.4969e-18	-5.2296e-12	0	0	0	0	0	0
26	-1.3478e-19	2.0176e-37	3.5087e-13	1.4969e-18	1	5.2295e-12	0	0	0	0	0	0
27	-3.2920e-13	-3.5087e-13	0	5.2296e-12	-5.2295e-12	1.0000	0	0	0	0	0	0
28	0	0	0	0	0	0	1	0	0	0	0	0
29	0	0	0	0	0	0	0	1	0	0	0	0
30	0	0	0	0	0	0	0	0	1	0	0	0
31	0	0	0	-5.7590e-08	-5.7590e-08	-2.7287e-14	0	0	0	-2.1715	9.1429e-10	4.3301e-13
32	0	0	0	4.0369e-08	4.0369e-08	-2.1644e-14	0	0	0	6.4089e-10	-1.6192	-5.1363e-13
33	0	0	0	-1.4434e-19	-1.2831e-19	3.6185e-19	0	0	0	1.2157e-13	1.1242e-13	-6.8807e-09

Figura 3.3: Matrice delle uscite C

linsys_A linsys_B linsys_C linsys_D

33x14 double

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

Figura 3.4: Matrice di legame ingresso-uscita D

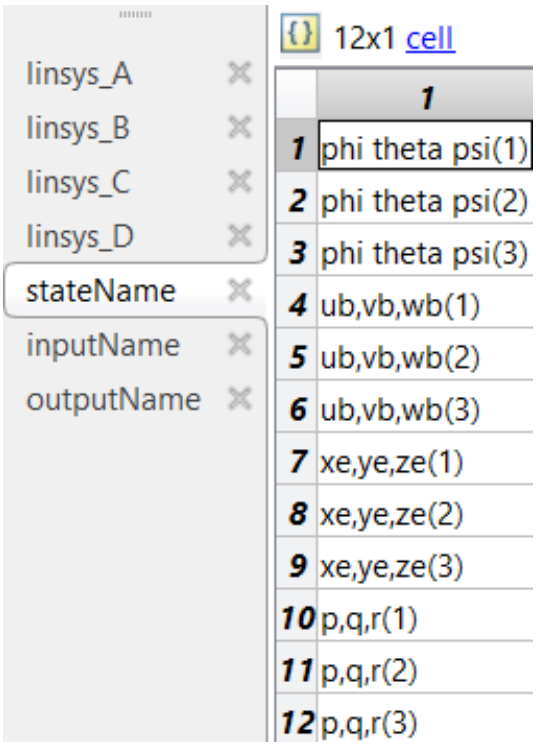


Figura 3.5: Vettore delle variabili di stato

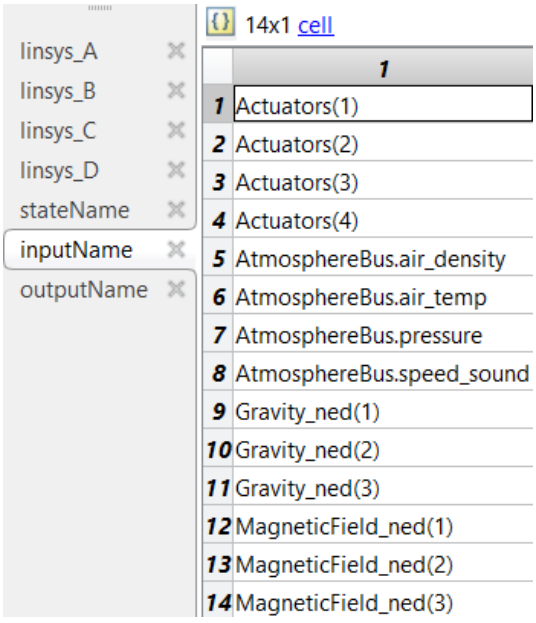


Figura 3.6: Vettore delle variabili di ingresso

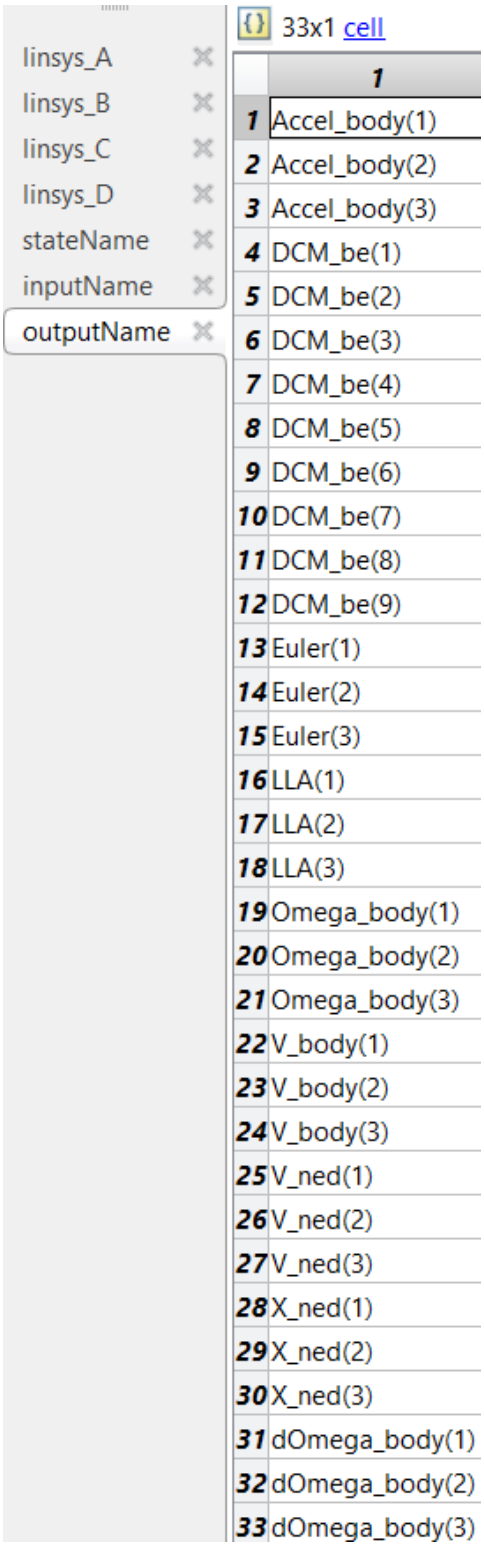


Figura 3.7: Vettore delle variabili di uscita

Analizzando la struttura delle variabili e delle matrici del sistema in *s.s.* possiamo trarre le seguenti considerazioni:

- le **variabili di stato** x_i del nostro sistema sono **12** e di conseguenza la matrice della dinamica A avrà dimensione 12×12
- gli **ingressi** u_i al sistema sono **14** avendo quindi una matrice degli ingressi B di dimensione 12×14
- le **uscite** y_i del sistema sono **33** che implica, quindi, una matrice delle uscite C di dimensione 33×12

Nel capitolo 5 in cui si affronterà la fase di sviluppo dei controllori tramite tecniche di controllo lineare, si utilizzerà il modello lineare in *spazio di stato* fin ora descritto come modello di riferimento. Inoltre, sfruttando le proprietà di tale rappresentazione è possibile considerare separatamente i legami tra le diverse variabili in gioco, in modo tale da poter scegliere dal modello completo solamente la sotto-parte relativa alle variabili interessate dallo studio. Per rendere più chiara l'idea, se ad esempio si volesse progettare un regolatore per il controllo della coordinata X, si sceglierà dal sistema totale solamente la parte dipendente da tale variabile e successivamente si costruirà, tramite le matrici standard "ridotte", la relativa funzione di trasferimento anch'essa "ridotta", cioè riferita solamente alle variabili oggetto dell'analisi.

Capitolo 4

Tecniche di controllo a confronto

Nel seguente capitolo si andranno ad enunciare le principali caratteristiche e differenze tra alcune tipologie di sistemi di controllo a retroazione negativa o *controreazione*, sia lineari che non. Nello specifico si presenterà una breve analisi dei controllori *PID* (non lineari) che come visto nella sezione 2.4 sono i sistemi di controllo utilizzati all'interno del progetto *asbQuadcopter* originale per la regolazione dei 6 *D.o.F.* (per maggiori dettagli si faccia riferimento alle sotto-sezioni 2.4.2, 2.4.3, 2.4.4, 2.4.5). Successivamente si andranno ad analizzare due tecniche di controllo di natura lineare, ovvero la sintesi per tentativi o *sintesi in frequenza* e la *sintesi con luogo delle radici*, che verranno utilizzate nel capitolo 5 per la progettazione dei regolatori lineari in sostituzione ai *PID* originali.

4.1 Sistemi di controllo *"PID"*

I controllori *PID* (*Proportional-Integrative-Derivative*) sono dei particolari sistemi in retroazione negativa ampiamente impiegati nel settore automatico-industriale, infatti essi sono i sistemi di controllo in *controreazione* più comuni negli apparecchi robotici presenti all'interno delle industrie automatizzate. In tali applicazioni viene di solito utilizzata una versione semplificata, ovvero un controllore *PI* in cui risulta essere assente l'azione derivativa. A causa della presenza di parti integrative I e derivative D, è facile intuire che la loro natura è non lineare.

Lo schema di controllo, in linea di principio, risulta essere il seguente:

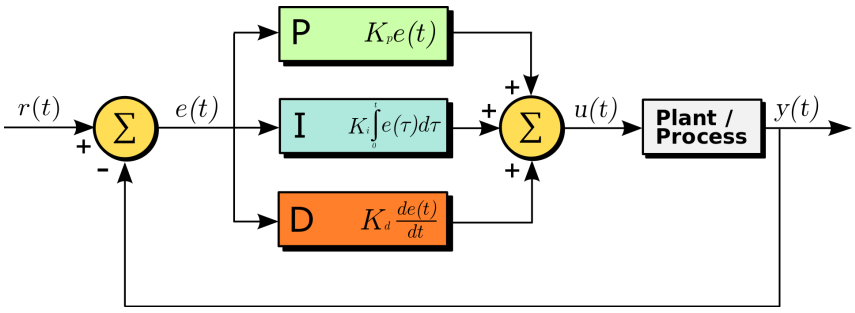


Figura 4.1: Schema di controllo per controllori PID

Osservando lo schema in figura 4.1 è possibile fare alcune considerazioni in merito al funzionamento di tali sistemi di controllo. In particolare, notiamo per prima cosa il classico schema dei regolatori con retroazione negativa, in cui abbiamo un *feedback* dell'uscita dal processo $y(t)$ la quale, attraverso dei sensori, viene misurata e sottratta all'input di riferimento $r(t)$. In questo modo si ottiene l'*errore di inseguimento* $e(t) = r(t) - y(t)$ che risulta essere la variabile di fondamentale importanza in tutti i sistemi di controllo a *controreazione*. Quello che si vuole ottenere durante il funzionamento a regime, è sempre un errore $e(t)$ quanto più possibile basso, ovvero $e(t) \to 0$ per $t \to +\infty$.

Questi sistemi di controllo sono in grado di "reagire" ad un eventuale errore $e(t)$ sia positivo che negativo in modo tale da renderlo quanto più possibile tendente al valore nullo così da avere un uscita $y(t)$ prossima all'input di riferimento $r(t)$. Inoltre, la reazione all'errore può essere regolata e ciò rende questa tipologia di controllori un sistema molto versatile. La regolazione avviene mediante un ingresso di controllo al processo $u(t)$ dato dalla somma dei contributi forniti dalle tre parti proporzionale P, integrativa I e derivativa D, come mostrato nello schema di controllo 4.1.

I controllori *PID*, anche se non lineari, sono relativamente semplici da comprendere, installare e tarare, al confronto con più complessi algoritmi di controllo basati sulla teoria del controllo ottimo e del controllo robusto. La taratura o stima dei parametri, avviene di solito attraverso semplici regole empiriche, come i metodi di *"Ziegler-Nichols"*, che risultano avere buone prestazioni nella maggior parte dei casi (in termini di regolazione a 0 dell'*errore di inseguimento*).

Entrando più nello specifico, vediamo in che modo viene determinato lo *sforzo di controllo* $u(t)$. Analizzando lo schema 4.1, si nota come esso risulta essere dato dalla somma algebrica dei contributi forniti dalle tre azioni P, I, D considerate separatamente, cioè:

$$u(t) = u_P(t) + u_I(t) + u_D(t) \tag{4.1}$$

Di seguito viene riportata una breve descrizione dell'azione di controllo fornita da ciascuna delle tre parti del controllore *PID*.

4.1.1 Controllore Proporzionale

$$u_P(t) = K_P e(t) \tag{4.2}$$

L'azione di controllo di tipo proporzionale P è ottenuta moltiplicando il segnale di input $e(t)$ per un'opportuna costante K_P detta *guadagno proporzionale*, in accordo con l'equazione 4.2.

Nel settore dei "controlli automatici", un regolatore dell'errore di questa tipologia risulta essere uno tra i più semplici, che in alcune applicazioni riesce comunque a rendere stabili sistemi di natura instabile. Per la natura matematica non risulta in ogni caso possibile ottenere la regolazione dell'errore $e(t)$ a zero ma al massimo esso si può rendere costante, sotto opportune ipotesi di controllo.

La costante di guadagno proporzionale K_P indica la "prontezza" del sistema a ciclo chiuso nel reagire ai segnali in ingresso: più essa è maggiore e più sarà rapida la reazione del sistema agli ingressi, ottenendo quindi un sistema molto reattivo. Esiste comunque un limite al valore di tale costante in quanto se troppo elevata, si produrrebbe in uscita un *sforzo di controllo* troppo "forte" che potrebbe danneggiare il sistema o portarlo a malfunzionamenti.

4.1.2 Controllore Integrativo

$$u_I(t) = K_I \int_0^t e(\tau) d\tau \tag{4.3}$$

L'azione di controllo di tipo integrativo o integrale I è proporzionale all'integrale nel tempo del segnale di ingresso $e(\tau)$ con $\tau \in [0; t]$ (Eq. 4.3), dove la costante di proporzionalità è detta *guadagno integrale* K_I.

Un controllore integrativo appartiene alla classe dei "controllori con memoria" ovvero esso tiene conto dell'errore commesso anche in istanti di tempo precedenti, al fine di produrre una graduale azione di controllo che renda l'errore nullo (o costante) a regime permanente. In questo caso, a differenza della parte proporzionale, è quindi possibile produrre un'azione di controllo in uscita (all'istante $t = \tau$) anche nel caso in cui l'errore $e(\tau)$ sia nullo. Questa proprietà fornisce al controllore *PID* la capacità di portare il processo $P(s)$ esattamente al punto di riferimento richiesto, dove la sola azione proporzionale risulterebbe nulla (poiché il punto di riferimento è $r(t)$ e di conseguenza in corrispondenza di esso l'errore $e(t)$ si annulla, annullando anche l'azione del controllore di tipo proporzionale, vedi eq. 4.2).

Il peso dell'azione di tipo I, all'interno del sistema di controllo complessivo, viene dato dalla costate di *guadagno integrale* K_I.

Esiste però un grave problema nel controllori *PID* legato alla presenza di parti integrative. Queste ultime, seppur utili nel migliorare la fedeltà di risposta di un sistema in presenza di disturbi costanti in catena diretta (introducendo un polo nell'origine), risultano essere degli elementi *"matastabili"* dove di conseguenza la condizione di equilibrio stabile non è garantita a livello globale. Per questo motivo i controllori *PID* soffrono del cosiddetto fenomeno del *"integral wind-up"*, già introdotto e discusso nel sottosezione 2.4.4, nel caso del controllore *PID* per l'assetto del drone in cui tale fenomeno viene "smorzato" tramite la presenza di un *feedback negativo* dell'uscita del controllore stesso.

4.1.3 Controllore Derivativo

$$u_D(t) = K_D \frac{de(t)}{dt} \tag{4.4}$$

L'azione di controllo di tipo derivativa D viene introdotta ai fini di migliorare le prestazioni complessive del controllore *PID*, tramite un segnale di controllo proporzionale dalla derivata rispetto al tempo t dell'ingresso $e(t)$. La costante di proporzionalità è detta *guadagno derivativo*, indicato nell'equazione 4.4 con K_D.

L'idea è quella di predire il comportamento futuro del sistema in modo tale da poter compensare rapidamente le variazioni del segnale in ingresso $e(t)$. Se ad esempio l'errore $e(t)$ sta aumentando con una determinata rapidità, l'azione di tipo D cerca di controllare questa variazione in funzione della velocità stessa con cui il segnale sta cambiando nel tempo. Questo consente di evitare che l'errore diventi troppo significativo nel tempo a causa dell'azione proporzionale oppure persista per un certo intervallo temporale non trascurabile, causato dall'azione integrale.

La "predizione del futuro" è possibile giustificarla analizzando il controllore I nel dominio di Laplace. Essa è dovuta all'anticipo di fase di 90° causata dall'introduzione di uno zero nell'origine, con un conseguente aumento del margine di fase e miglioramento dei parametri transitoriali del sistema, quali tempo di salita e sovraelongazione.

Spesso però nelle applicazioni reali l'azione di tipo I viene trascurata in quanto rende i sistemi troppo sensibili alle variazioni di segnale e di conseguenza è sconsigliato l'utilizzo in applicazioni in cui l'attuatore fisico non risulti adeguato a sopportare tali rapidi cambiamenti.

Scegliendo il giusto peso all'interno del sistema di controllo *PID* complessivo, tramite la costante K_D, è comunque possibile implementare controllori con parti derivative per aumentare le prestazioni generali. Questo a patto che il sistema di controllo sia fisicamente realizzabile, in quanto sappiamo che un controllore interamente derivativo è impossibile da realizzare poiché la relativa funzione di trasferimento conterebbe più zeri che poli (funzione impropria), il che va contro le ipotesi di controllo. Di conseguenza l'azione derivativa va sempre affiancata ad almeno un'azione integrativa (che introduce un polo in 0), oppure analogamente introducendo un qualsiasi polo in alta frequenza.

In conclusione, combinando le equazioni 4.1, 4.2, 4.3, 4.4, si ottiene lo *sforzo di controllo* $u(t)$ totale:

$$u(t) = K_P e(t) + K_I \int_0^t e(\tau)d\tau + K_D \frac{de(t)}{dt} \tag{4.5}$$

Il risultato ottenuto nella 4.5 è in accordo con quanto riportato nello schema di controllo generale in figura 4.1.

4.2 Controllo lineare

Nella presente sezione verranno riportate brevemente le principali caratteristiche inerenti al controllo lineare con riferimento specifico alle due tecniche di controllo basate sulla *sintesi in frequenza* e *sintesi con luogo delle radici*.

L'introduzione di tecniche di controllo lineare è di fondamentale importanza per il presente elaborato in quanto su tali concetti teorici si baseranno i controllori reali sviluppati e implementati nei capitoli 5 e 6.

Alla base del controllo lineare c'è sempre un modello matematico (lineare) di partenza di solito rappresentato mediante un sistema in *spazio di stato*, ovvero della tipologia:

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx + Du \end{cases} \tag{4.6}$$

Per maggiori dettagli in merito si faccia riferimento alla sezione 2.3.2, con particolare riferimento ai sistemi di equazioni 2.1 e 2.2.

In generale, lo schema di controllo utilizzato per la progettazione di sistemi di controllo lineari è il seguente:

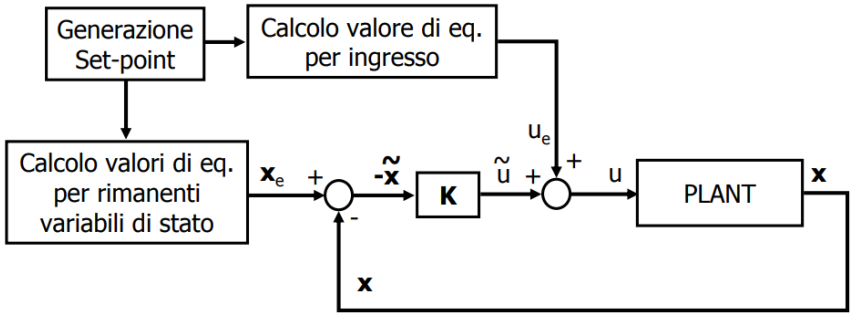


Figura 4.2: Schema di controllo lineare con retroazione negativa dello stato

Dalla figura 4.2 si nota la classica struttura di controllo in *controreazione* o retroazione negativa dello stato x. Inoltre, considerando come caso più generico quello nel quale l'obiettivo di controllo sia fissare alcune variabili di stato in un punto di equilibrio x_e (*set-point*), da queste ottenere i valori di equilibrio per le altre variabili e per l'ingresso u_e, e stabilizzare le variazioni rispetto a questo punto di equilibrio con una retroazione negativa dello stato x, allora la legge di controllo sarà del tipo:

$$\tilde{u} = -K\tilde{x} \tag{4.7}$$

in cui $\tilde{x} = x_e - x$ e la legge di controllo totale $u = \tilde{u} + u_e$.

Dall'equazione 4.7 è facile osservare la linearità della legge di controllo progettata e utilizzata per la stabilizzazione del processo *PLANT*. In generale, l'azione di controllo di un qualsiasi controllore lineare sarà sempre statica e di tipo proporzionale, la cui

costante di proporzionalità K è detta *costante di guadagno del controllore*. Nel caso più semplice in cui lo stato x del sistema si suppone misurabile (*stato accessibile*), il controllore agisce direttamente sul valore di quest'ultimo, mentre nei casi in cui esso non sia misurabile (*stato non accessibile*), allora il controllore dovrà agire su una sua stima calcolata da un ulteriore sistema dinamico detto *osservatore asintotico dello stato* il quale permette di stimare lo stato del processo a partire dai valori delle variabili di *input* e *output*.

Prima di procedere con l'analisi delle due tecniche di sintesi sopracitate, andiamo a dare una definizione più rigorosa di "sintesi", enunciando le principali caratteristiche.

Per *sintesi di un sistema di controllo* si intende la scelta della funzione di trasferimento del controllore in modo che la risposta del sistema in catena chiusa abbia le caratteristiche desiderate. Tali caratteristiche,in una terminologia più 'tecnica', vengono dette *specifiche del controllore* e si dividono in due tipologie:

Specifiche univoche: possono essere soddisfatte in maniera univoca. In altri termini, esiste solamente un modo attraverso il quale esse possano essere rispettate. Rientrano in questa tipologia le specifiche sul regime permanente per ingressi polinomiali, sull'astatismo ai disturbi costanti e sul vincolo nel guadagno del sistema.

Specifiche lasche: possono essere soddisfatte in maniera non univoca, ovvero è possibile seguire approcci differenti che portano comunque allo stesso risultato. Rientrano in questa tipologia le specifiche sul regime permanente per ingressi sinusoidali (non polinomiali) e sulla risposta transitoria del sistema.

4.2.1 Sintesi in frequenza

La prima tecnica di controllo analizzata è quella basata sulla *sintesi in frequenza*, appartenente alla categoria di sintesi per tentativi dove la progettazione del controllore avviene per gradi, in modo tale da soddisfare gradualmente tutte le specifiche richieste. In generale si avranno quindi più controllori, uno per ogni tentativo, la cui unione darà luogo al controllore finale. Nella prima fase di sintesi si soddisfano le specifiche univoche mediante un *"controllore di primo tentativo"*, nella seconda quelle lasche tramite l'implementazione di una *"funzione compensatrice"* $R(s)$.

Per il passaggio da *dominio del tempo* a *dominio della frequenza* è necessario per prima cosa passare al *dominio di Laplace* e successivamente per mezzo della sostituzione $s = j\omega$ si passa al *dominio della frequenza*, in cui lo studio è incentrato sulla 'risposta armonica' del sistema. La sintesi in ω utilizza come strumenti di analisi i *diagrammi di Bode* e di *Nyquist*, fondamentali per lo studio della stabilità e per il calcolo del modulo e della fase di un sistema rispetto all'ingresso di riferimento.

Lo schema concettuale di controllo, a cui tale tecnica fa riferimento, è il seguente:

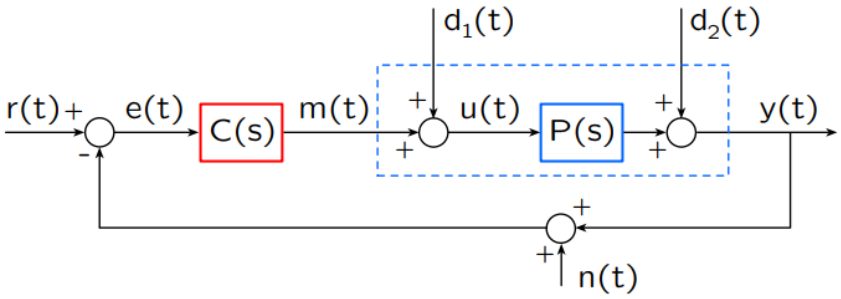


Figura 4.3: Schema di controllo a retroazione unitaria in presenza di disturbi

dove:

- $r(t)$ ingresso di riferimento
- $y(t)$ uscita del processo o 'variabile controllata'
- $m(t)$ ingresso di controllo prodotto dal controllore $C(s)$
- $d_1(t), d_2(t)$ disturbi in catena diretta, agenti rispettivamente in ingresso e in uscita del processo $P(s)$
- $n(t)$ disturbo agente sul ramo di retroazione
- $e(t)$ errore di inseguimento

Nella figura 4.3 il processo è rappresentato dal blocco $P(s)$ mentre il controllore dal blocco $C(s)$ (in genere esso viene indicato con $G(s)$). Per semplicità si considera il sistema in anello chiuso a retroazione unitaria, di conseguenza $K_D = 1$. Inoltre, per convenzione, il prodotto in catena aperta $C(s)P(s)$ viene indicato con $F(s)$, mentre il prodotto in anello chiuso con $W(s)$ (in entrambi casi $s = j\omega$). La relazione che lega la funzione di trasferimento in catena chiusa con quella in catena aperta (o diretta), sotto l'ipotesi $K_D = 1$, è la seguente:

$$W(s) = \frac{F(s)}{1 + F(s)} \tag{4.8}$$

L'equazione 4.8 deriva dalla diretta applicazione della *regola di Mason* per il calcolo di una funzione di trasferimento in presenza di diagrammi di flusso, come nel nostro caso.

É necessario precisare che gli strumenti prima citati, quali *diagrammi di Bode/Nyquist*, permettono di verificare le specifiche in catena diretta, mentre per la verifica in catena chiusa si utilizza la cosiddetta *Carta di Nichols*.

Prima di procedere con le fasi di progettazione adoperate nella *sintesi in frequenza*, andiamo ad elencare alcuni parametri fondamentali utili per la verifica dei requisiti richiesti. In particolare, è possibile eseguire una distinzione in:

- **Parametri in catena chiusa**
  - M_r: modulo alla risonanza, espresso in dB. Rappresenta la differenza tra il massimo valore assunto dal modulo della f.d.t. in catena chiusa ed il suo valore in corrispondenza di $\omega = 0$.

$$M_r = |W(j\omega)|_{dB}^{max} - |W(j0)|_{dB}$$

  - B_3: banda passante, ovvero la pulsazione ω in corrispondenza della quale il valore $|W(j\omega)|_{dB}$ diminuisce di $3dB$ rispetto a $|W(j0)|_{dB}$

- **Parametri in catena aperta**
  - ω_t: pulsazione di attraversamento, ovvero la pulsazione in corrispondenza della quale il modulo della funzione di trasferimento in catena diretta risulta essere pari a $0dB$.

$$\omega_t \quad : \quad |F(j\omega_t)|_{dB} = 0dB$$

  - m_φ: margine di fase, ovvero il valore della fase di $F(j\omega)$ in corrispondenza di $\omega = \omega_t$, aumentato di 180°.

$$m_\varphi = \angle F(j\omega_t) + 180^\circ$$

Tali parametri sono espressi nel dominio della frequenza ω e sono strettamente collegati anche ad alcuni **parametri nel dominio del tempo** espressi quindi in catena chiusa. I più importanti sono:

- t_s: tempo di salita, ovvero il tempo impiegato dell'uscita del sistema $y(t)$ per raggiungere per la prima volta il valore di riferimento $r(t)$.
- $\hat{s}$: sovraelongazione, che rappresenta di quanto l'uscita del sistema $y(t)$ supera il valore di riferimento $r(t)$ durante la fase transitoria.
- t_a: tempo di assestamento, ovvero il tempo necessario al sistema ad assestarsi al valore di regime $r(t)$. Tale intervallo di tempo rappresenta il transitorio del sistema.

Esistono delle **relazioni algebriche** empiriche che mettono in relazione i parametri nel dominio del tempo con quelli nel dominio della frequenza, sia a ciclo chiuso che a ciclo aperto. Esse sono riportate nella 4.9, 4.10, 4.11.

$$\frac{1+\hat{s}}{M_r} \approx 0.85 \tag{4.9}$$

$$[B_3]_{\frac{rad}{s}} \cdot [t_s]_s \approx 3rad \tag{4.10}$$

$$[\omega_t]_{\frac{rad}{s}} \approx 3 \div 5[B_3]_{Hz} \tag{4.11}$$

Utilizzando tali semplici relazioni è quindi possibile passare da requisiti espressi in catena aperta a requisiti espressi in catena chiusa, nonché da parametri nel dominio del tempo a parametri nel dominio della frequenza.

$$t_s, \hat{s} \rightarrow B_3, M_r \rightarrow \omega_t, m_\varphi$$

Andiamo ora ad elencare e descrivere brevemente le fasi progettuali adoperate in questa tipologia di sintesi.

PROCEDURA DI SINTESI

Per prima cosa si vanno ad analizzare le specifiche richiesti e prima di procedere con lo sviluppo del controllore $G(s)$ si verifica se l'attuale funzione di trasferimento del processo $P(s)$ non soddisfi già tali requisiti. Nella maggior parte dei casi questo non accade e di conseguenza si procede con la progettazione di un sistema di controllo.

- **Controllore di primo tentativo $\hat{G}(s)$**

  Si procede con l'implementazione di un primo controllore, detto *controllore di primo tentativo*, atto a soddisfare le specifiche univoche, quali errore a regime e astatismo rispetto a disturbi. In questo modo si ottiene un sistema di controllo della seguente forma:

  $$\hat{G}(s) = \frac{K_G}{s^k} \tag{4.12}$$

  La struttura del controllore $\hat{G}(s)$ deriva direttamente dalle specifiche richieste, nello specifico il numero di poli nell'origine s^k deriva dalla *teoria dei sistemi di tipo k* riguardo all'errore commesso e_k rispetto ingressi canonici di grado $n = k$, mentre la costante di guadagno K_G è legata ad e_k secondo la relazione 4.13.

  $$e_k = \frac{K_D^2}{K_F} = \frac{K_D^2}{K_G K_P} \tag{4.13}$$

  Nell'ipotesi di retroazione unitaria si ha $K_D = 1$ e di conseguenza la 4.13 diventa:

  $$e_k = \frac{1}{K_F} = \frac{1}{K_G K_P} \tag{4.14}$$

  Per quanto riguarda la *tipologia di sistema* desiderata rispetto ingressi canonici di grado k, vale la seguente relazione:

  "Un sistema è di tipo k $\iff$ *W(s) ha k zeri in* $s = 0$ $\iff$ *F(s) ha k poli in* $s = 0$ *"*

Per *sistema di tipo k* si intende un sistema il cui errore commesso e_k risulta essere costante rispetto ingressi di grado $n = k$, pari a zero per ingressi di grado $n < k$ ed infinito per ingressi di grado $n > k$.

Invece, per garantire astatismo rispetto disturbi costanti è necessario che sia presente un polo in $s = 0$ 'a monte' del disturbo. Riferendosi allo schema 4.3 ed in particolare ai disturbi agenti in catena diretta $d_1(t) = cost$ e $d_2(t) = cost$, per garantire astatismo rispetto ad essi è quindi necessario che sia presente almeno un polo in $s = 0$ rispettivamente nella f.d.t. del controllore $G(s)$ e nella f.d.t. del processo $P(s)$. Di conseguenza, se per natura fisica il processo da controllare non ha poli in $s = 0$, sarà impossibile ottenere astatismo rispetto ad un eventuale disturbo d_2 agente sull'uscita di $P(s)$. In tal caso si agirà sul guadagno K_G al fine di rendere accettabile l'influenza di tali disturbi sul sistema complessivo.

Prima di passare alla fase successiva, si vanno a verificare le prestazioni dopo l'aggiunta del controllore $\hat{G}(s)$ per controllare se eventualmente le specifiche siano già state soddisfatte. Se così non fosse, si passa alla progettazione della *funzione compensatrice*.

- **Progetto della rete compensatrice $R(s)$**

  Una volta verificate le specifiche univoche si passa alla progettazione di un controllore di secondo tentativo che soddisfi le *specifiche lasche* senza compromettere i risultati ottenuti con il controllore di primo tentativo utilizzato per le specifiche univoche.

  Per la sintesi della *rete compensatrice* $R(s)$ si utilizzano le cosiddette '*funzioni ausiliarie*' le quali possono essere di due tipologie:

  1. '*Funzioni anticipatrici*': aumentano di molto la fase della f.d.t. complessiva e di poco il modulo. La struttura generale è mostrata nella funzione 4.15.

  $$R_a(j\omega) = \frac{1 + \frac{j\omega}{\omega_a}}{1 + \frac{j\omega}{m_a\omega_a}} \tag{4.15}$$

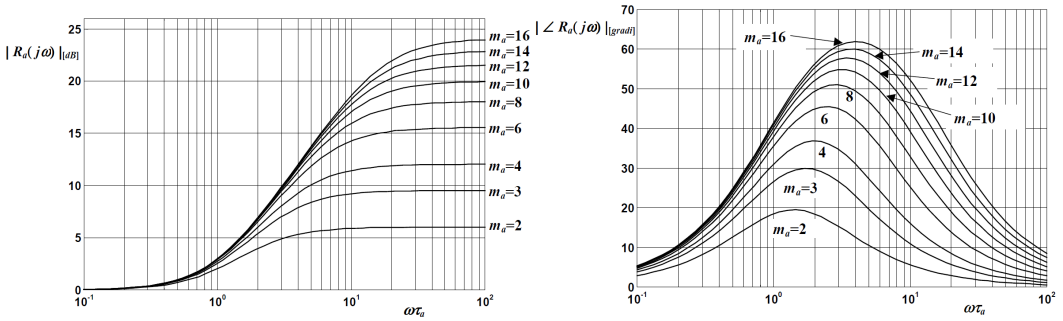
     Agendo sui valori di $m_a > 1$ e $\tau_a = \frac{1}{\omega_a}$ è possibile controllare l'entità dell'aumento di modulo e fase.

  2. '*Funzioni attenuatrici*': diminuiscono di molto il modulo della f.d.t. complessiva e di poco la fase. La struttura generale è mostrata nella funzione 4.16.

  $$R_i(j\omega) = \frac{1 + \frac{j\omega}{m_i\omega_i}}{1 + \frac{j\omega}{\omega_i}} \tag{4.16}$$

Agendo sui valori di $m_i > 1$ e $\tau_i = \frac{1}{\omega_i}$ è possibile controllare l'entità della diminuzione di modulo e fase.

I diagrammi di Bode di tali funzioni sono detti *diagrammi universali*, riportati in figura 4.4.



(a) Diagramma di Bode del modulo (b) Diagramma di Bode della fase

Figura 4.4: Diagrammi universali delle funzioni anticipatrici

I diagrammi delle reti attenuatrici sono analoghi ma speculari rispetto l'asse delle ascisse.

Si noti che, per non compromettere gli effetti del controllore di primo tentativo, le funzioni compensatrici $R(s)$ presentano un guadagno statico pari a zero e non possiedono né poli né zeri nell'origine del piano complesso. Questo è possibile grazie all'utilizzo esclusivamente di *termini binomi* (come nelle funzioni 4.15, 4.16) e *termini trinomi*.

Una volta trovata la giusta *rete compensatrice* si vanno a combinare insieme i controllori $\hat{G}(S)$ (Eq. 4.12) e $R(s)$ (Eq. 4.15, 4.16), ottenendo un controllore complessivo con la seguente espressione:

$$G(s) = \hat{G}(s)R(s) = \frac{K_G}{s^k}R(s) \tag{4.17}$$

Se il controllore progettato risulta soddisfare le specifiche in catena aperta, ovvero quelle derivabili dai diagrammi di Bode, si passa alla verifica delle specifiche in catena chiusa tramite la carta di Nichols, in modo tale da verificare che il sistema in catena chiusa rispetti effettivamente i requisiti richiesti.

4.2.2 Sintesi con luogo delle radici

La seconda tecnica di controllo analizzata è la sintesi basata sul concetto di *luogo delle radici*, anch'essa appartenente alla categoria di sintesi effettuate per tentativi. A differenza della tecnica precedentemente analizzata nella sezione 4.2.1, per lo studio del sistema da controllare viene utilizzato il *piano di Gauss* in cui vengono

rappresentate le posizioni occupate dei poli della funzione di trasferimento in catena chiusa $W(s)$ al variare di alcuni parametri definiti, invece, in catena aperta.

Per la realizzazione dei tale grafico all'interno del piano complesso, si parte dalla definizione della f.d.t. in catena aperta $F(s)$, espressa secondo una particolare rappresentazione detta *forma canonica poli-zeri*, riportata nell'espressione 4.18..

$$F(s) = K \frac{\prod_{i=1}^{m} (s - z_i)}{\prod_{i=1}^{n} (s - p_i)} \tag{4.18}$$

La costante K è detta *coefficiente di guadagno in catena diretta*, che in generale è diversa dalla costante di guadagno statico K_F. Mediante la seguente rappresentazione è possibile ricavare facilmente il numero e la posizione di tutti gli n poli ed m zeri della f.d.t. in catena diretta.

Dalla relazione 4.8 è possibile ricavare la f.d.t. in catena chiusa, sostituendo al posto di $F(s)$ l'espressione 4.18, ottenendo quanto segue.

$$W(s) = K \frac{\prod_{i=1}^{m} (s - z_i)}{\prod_{i=1}^{n} (s - p_i) + K \prod_{i=1}^{m} (s - z_i)} \tag{4.19}$$

Deve necessariamente risultare $n > m$ altrimenti si otterrebbe una funzione impropria fisicamente irrealizzabile.

Avendo trovato l'espressione della funzione di trasferimento in catena diretta $W(s)$, ponendo il denominatore uguale a zero si ottiene il cosiddetto *polinomio caratteristico* le cui radici rappresentano nient altro che i poli della stessa $W(s)$.

$$\prod_{i=1}^{n} (s - p_i) + K \prod_{i=1}^{m} (s - z_i) = 0 \tag{4.20}$$

Essendo la 4.20 una funzione nelle variabili s e K, per semplicità verrà indicata mediante l'espressione semplificata:

$$f(s, K) = 0 \quad \text{Polinomio caratteristico di W(s)} \tag{4.21}$$

Possiamo quindi affermare che il *luogo delle radici* è il luogo dei punti del piano complesso percorsi dalle radici del polinomio caratteristico $f(s, K) = 0$, cioè dai poli della funzione di trasferimento in catena chiusa $W(s)$, al variare del coefficiente di guadagno K della funzione di trasferimento in catena diretta $F(s)$.

Come anticipato, in generale $K \neq K_F$ ma esiste una relazione che permette il passaggio da una costante all'altra, ovvero:

$$K_F = \lim_{s \to 0} F(s) = \frac{K \prod_{i=1}^{m} (-z_i)}{\prod_{i=1}^{n} (-p_i)} \tag{4.22}$$

Risolvere in maniera esatta l'equazione 4.20 non è possibile in quanto non si conoscono a priori i poli e gli zeri della $F(s)$. Si può però tracciare qualitativamente l'andamento con cui variano le radici in funzione di K sfruttando alcune proprietà fondamentali alla base del concetto di *luogo delle radici*.

Proprietà fondamentali del luogo delle radici

1. Ad ogni radice di $f(s, K) = 0$ corrisponde un percorso sul piano di Gauss al variare di K, detto *ramo* del luogo delle radici.
2. Il luogo delle radici è diviso in *luogo positivo*, per valori di $K > 0$ e *luogo negativo*, per valori di $K < 0$.
3. Per $K \to 0$, gli n poli in catena chiusa tendono agli n poli in catena aperta. Ciò significa che per tracciare il luogo positivo e negativo si parte sempre dai punti $p_1, \ldots, p_n$ ovvero dai poli della $F(s)$.
4. Il luogo delle radici è simmetrico rispetto l'asse reale, in quanto le radici possono del polinomio caratteristico possono essere o reali o complesse coniugate.
5. Tutto l'asse reale appartiene al luogo delle radici: una parte appartiene al luogo positivo, la restante al luogo negativo.
6. Appartengono al luogo positivo quelle porzioni di asse reale che lasciano alla propria destra un numero dispari di poli e/o zeri di $F(s)$, contati con la loro molteplicità.

Oltre tali proprietà di base, esistono altri concetti fondamentali alla base del tracciamento del luogo delle radici. Essi riguardano il comportamento delle radici in presenza di di *punti singolari* e *punti regolari*, nonché l'andamento del luogo per $K \to \pm\infty$.

Punti singolari e punti regolari : al variare di K alcune radici del polinomio caratteristico possono coincidere nello stesso punto, dando luogo al cosiddetto *punto singolare*. Essi coincidono quindi con le radici multiple di $f(s, K)) = 0$, mentre gli altri punti del luogo vengono detti *punti regolari*. Le condizioni analitiche che li descrivono sono le seguenti:

$$\text{Punti singolari} \to \begin{cases} f(\bar{s}, K) = 0 \\ \left.\dfrac{\partial f(s, K)}{\partial s}\right|_{s=\bar{s}} = 0 \end{cases} \qquad \text{Punti regolari} \to \begin{cases} f(\bar{s}, K) = 0 \\ \left.\dfrac{\partial f(s, K)}{\partial s}\right|_{s=\bar{s}} \neq 0 \end{cases}$$

Comportamento per $K \to \pm\infty$: per $K \to +\infty$ per il luogo positivo e $K \to -\infty$ per il luogo negativo si ha che:

1. m rami tendono agli zeri z_i della $F(s)$. Se uno zero ha molteplicità $l > 1$ allora tendono su di esso l rami del luogo.
2. $(n - m)$ rami tendono ad $(n - m)$ semirette, dette *asintoti del luogo* il cui punto di origine è chiamato *centro degli asintoti*, indicato con s_0 e situato sul punto dell'asse reale avente la seguente espressione:

$$s_0 = \frac{\sum_{i=1}^{n} p_i - \sum_{i=1}^{m} z_i}{n - m}$$

Il suo valore sarà sempre reale in quanto nella sommatoria le eventuali parti immaginarie di poli e/o zeri complessi coniugati si elideranno a vicenda.

Stando a quanto detto fin ora, sfruttando i concetti e le proprietà introdotte è possibile tracciare in maniera qualitativa il luogo delle radici di una qualsiasi funzione di trasferimento espressa nella forma 4.18. Tuttavia esistono delle condizioni che consentono di verificare il corretto tracciamento dei punti sul piano di Gauss. In particolare, ogni punto del luogo deve avere modulo e fase che rispetti le cosiddette *condizioni di modulo* e *condizioni di fase*.

$$\textbf{Condizione di modulo} \to |K| = \frac{\prod_{i=1}^{n} |s - p_i|}{\prod_{i=1}^{m} |s - z_i|}$$

$$\textbf{Condizione di fase} \to \sum_{i=1}^{n} \angle(s - p_i) - \sum_{i=1}^{m} \angle(s - z_i) = \begin{cases} (2h+1)\pi & K > 0 \\ 2h\pi & K < 0 \end{cases}$$

Entrambe le condizioni sono ricavabili dall'equazione del polinomio caratteristico di $W(s)$, riportata nella 4.20.

Capitolo 5

Sintesi di regolatori lineari per il controllo dell'assetto del minidrone *Parrot Mambo*

Nel seguente capitolo si affronterà la fase di progettazione di regolatori lineari per il controllo dell'assetto in volo del minidrone *Parrot Mambo*. I gradi di libertà responsabili della sua regolazione sono quelli relativi agli angoli di inclinazione rispetto gli assi del *Body Frame*, i rispettivi angoli di *Roll, Pitch* e *Yaw*, argomentati nella parte introduttiva dell'elaborato nella sotto-sezione 1.3.2. I controllori per tali gradi di libertà verranno sviluppati mediante le tecniche di controllo lineare presentate nella sezione 4.2 del capitolo precedente, ovvero *sintesi in frequenza* e *sintesi mediante luogo delle radici*. Il modello matematico di partenza per l'implementazione dei sistemi di controllo è quello riportato nella sezione 3.2, espresso in *spazio di stato* e ottenuto tramite un processo di linearizzazione del modello reale non lineare (la fase di linearizzazione è argomento del capitolo 3). Da tale modello complessivo verranno selezionate solamente le variabili di interesse relative ai vari gradi di libertà interessati, tralasciando il comportamento assunto delle altre.

Durante la fase di sviluppo dei controllori verrà fatto uso dello strumento *Control System Designer* messo a disposizione in ambiente MATLAB e utile per la progettazione di controller per sistemi dinamici *SISO* (single input-single output) con feedback dallo stato. Tramite la *Command Window* di MATLAB è possibile accedere a tale strumento digitando il comando *'sisotool'*. Un'interfaccia grafica molto intuitiva consente la scelta della struttura dello schema di controllo desiderato e permette l'inserimento delle funzioni di trasferimento all'interno dei relativi blocchi del modello, con la successiva visualizzazione dei corrispettivi diagrammi di Bode, Nyquist, Nichols, luogo delle radici, etc.

5.1 Sintesi in frequenza del controllore per l'angolo di *roll*

Per la sintesi del regolatore lineare utilizzato nel controllo dell'angolo di *roll*, ovvero dell'angolo di inclinazione ϕ attorno l'asse x o *asse di rollio*, è stata utilizzata come tecnica di controllo la *sintesi in frequenza*, detta anche *sintesi in* ω.

La prima cosa da fare è definire la funzione di trasferimento del processo $P(s)_{roll}$ da controllare, nel nostro caso rappresentato dalla dinamica dell'angolo di *roll*. Dal

modello in spazio di stato del minidrone *Parrot Mambo* presentato nella sezione 3.2, è possibile ricavare dalle matrici standard complessive A, B, C, D solamente i coefficienti relativi alle componenti delle variabili di stato, ingresso e uscita che influenzano la dinamica del *roll*. In particolare, facendo riferimento alla struttura di tali vettori presente nelle figure 3.5, 3.6, 3.7, le componenti rilevanti per lo studio della dinamica del *roll* sono:

- Vettore degli stati: posizione angolare ϕ e velocità angolare p del *roll*, ovvero le componenti **phi theta psi(1)** e **p,q,r(1)**.
- Vettore degli ingressi: la spinta esercitata da ciascun motore, ovvero le componenti **Actuators(1)**, **Actuators(2)**, **Actuators(3)** e **Actuators(4)**.
- Vettore delle uscite: l'angolo di Eulero relativo al *roll*, ovvero la componente **Euler(1)**.

Di conseguenza, i corrispettivi coefficienti selezionati delle matrici standard complessive A, B, C, D presenti in figura 3.1, 3.2, 3.3, 3.4, portano alla formazione delle seguenti "**matrici standard ridotte per l'angolo di *roll***":

$$A_{roll} = \begin{bmatrix} 0 & 1 \\ 0 & -2.1715 \end{bmatrix} \quad B_{roll} = \begin{bmatrix} 0 \\ 0.4202 \end{bmatrix} \quad C_{roll} = \begin{bmatrix} 1 & 0 \end{bmatrix} \quad D_{roll} = \begin{bmatrix} 0 \end{bmatrix}$$

Il sistema in *spazio di stato* associato a tali matrici $A_{roll}, B_{roll}, C_{roll}, D_{roll}$ risulta quindi avere la seguente forma:

$$\begin{cases} \dot{x} = \begin{bmatrix} 0 & 1 \\ 0 & -2.1715 \end{bmatrix} x + \begin{bmatrix} 0 \\ 0.4202 \end{bmatrix} u \\ y = \begin{bmatrix} 1 & 0 \end{bmatrix} x \end{cases} \tag{5.1}$$

con $\quad x = \begin{bmatrix} \text{phi theta psi(1)} \\ \text{p,q,r(1)} \end{bmatrix} \quad$ e $\quad u = \text{Actuators(i)} \quad$ con $\quad i = 1, 2, 3, 4.$

Essendo $D = 0$, il sistema risulta essere della forma 2.2 e la relativa funzione di trasferimento sarà necessariamente un funzione (lineare) strettamente propria, ovvero con con Grado numeratore $<$ Grado denominatore. In termini tecnici $P(s)_{roll}$ ha un numero di poli maggiore rispetto al numero degli zeri, in accordo con la *"fisica realizzabilità"* del processo.

Una volta individuate le matrici standard del processo relativo alla dinamica del *roll* è necessario andare a trovare la relativa funzione di trasferimento di $P(s)_{roll}$ associata al sistema in *spazio di stato* sopra riportato. In generale, una generica funzione di trasferimento espressa nel dominio di Laplace è rappresentata da un'espressione avente la seguente forma:

$$H(s) = \frac{Y(s)}{U(s)} \tag{5.2}$$

Essa, per definizione, è pari al rapporto tra l'uscita $Y(s)$ del sistema e l'ingresso $U(s)$, in altri termini è una funzione che lega in maniera diretta l'uscita misurata all'ingresso fornito in input al sistema.

Nel caso di sistemi (lineari) espressi mediante *rappresentazione in spazio di stato*, come nel nostro caso, essa risulta assumere una forma nota e dipendente dalle relative matrici standard associate al modello. La sua espressione è la seguente:

$$H(s) = D + C(sI - A)^{-1}B \tag{5.3}$$

Se il sistema in *s.s.* considerato è della tipologia *SISO* (single input-single output), allora il risultato ottenuto dalla 5.3 sarà una funzione del tipo:

$$H(s) = K\frac{\prod_{i=0}^{Z}(s - z_i)}{\prod_{i=0}^{P}(s - p_i)} \tag{5.4}$$

dove i valori z_i con $i = 1, \ldots, Z$ rappresentano gli zeri della funzione di trasferimento, i valori p_i con $i = 1, \ldots, P$ i poli e la costante K il guadagno, la cosiddetta *rappresentazione poli-zeri* già introdotta nella sezione 4.2.2.

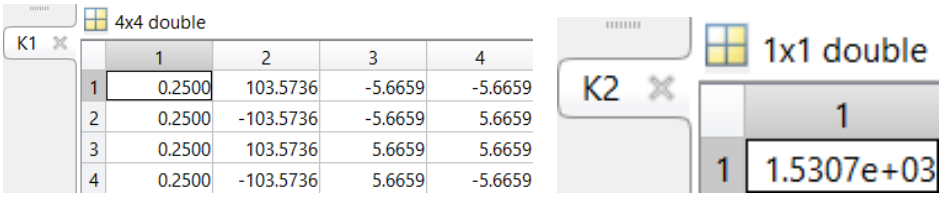
Nel nostro caso, essendo appunto il processo $P(s)_{roll}$ della tipologia *SISO*, la relativa f.d.t. troveremo essere della forma 5.4.

Prima di procedere con il calcolo della funzione di trasferimento associata alla dinamica del *roll* tramite l'applicazione diretta della 5.3, è necessario aggiungere due costanti moltiplicative K_1 e K_2 rappresentati rispettivamente il peso attraverso il quale lo sforzo di controllo viene ripartito tra i 4 attuatori e il coefficiente di spinta di ciascuno di essi, considerato per tutti identico (per dettagli si faccia riferimento ai blocchi 2.25 e 2.26). Per accedere al valore di tali costanti è sufficiente aprire il progetto *asbQuadcopter* e inserire i seguenti comandi all'interno della Command Window di MATLAB:

```
» K1 = Controller.Q2Ts;
» K2 = Vehicle.Motor.thrustToMotorCommand;
```

In questo modo verranno create all'interno del Workspace del progetto le due nuove variabili K_1 e K_2 i cui valori sono riportati nella figura 5.1.

Osservando la struttura della costante K_1, in figura 5.1a, essa risulta essere una matrice di dimensioni 4×4 le cui colonne rappresentano rispettivamente i pesi attribuiti allo sforzo di controllo dei 4 motori in fase di movimenti verticali lungo



(a) Struttura costante K_1 (b) Struttura costante K_2

Figura 5.1: Costanti moltiplicative blocco *Flight Controller*

z, movimenti attorno l'asse di imbardata, movimenti attorno l'asse di beccheggio e movimenti attorno l'asse di rollio.

Per la seconda costante K_2 rappresentante la spinta esercitata dagli attuatori, essendo per ipotesi costruttive considerabile identica per tutti e 4 i motori, la sua struttura è semplicemente costituita da un valore numerico (matrice 1×1), come riportato in figura 5.1b.

Dovendo considerare solamente le componenti relative allo studio della dinamica dell'angolo di *roll*, i valori scelti di K_1 e K_2 sono:

$$K_1 = 5.6659 \qquad K_2 = 1.5307 \times 10^3$$

Ottenuti i valori di tali costanti moltiplicative è ora possibile ricavare la funzione di trasferimento relativa alla dinamica del *roll*, il cui calcolo verrà eseguito sfruttando alcune funzioni dedicate presenti all'interno del *Control System Toolbox* di MATLAB. Digitando i seguenti comandi all'interno della Command Window si ottiene il risultato desiderato.

```
» A_roll = [0, 1; 0, -2.1715];
» B_roll = [0; 0.4202];
» C_roll = [1, 0];
» D_roll = [0];
» K1_roll = 5.6659;
» K2 = 1.5307e+03;
» sys_roll = ss(A_roll, B_roll, C_roll, D_roll);
» plant_roll = tf(sys_roll);
» Ps_roll = K1_roll * K2 * plant_roll
```

La funzione di trasferimento associata alla dinamica del *roll* ha quindi la seguente espressione:

$$\boldsymbol{P(s)_{roll} = \frac{3644}{s(s + 2.171)}} \tag{5.5}$$

Trovato il processo $P(s)_{roll}$ da controllare è ora possibile procedere con la sintesi vera e propria del controllore $G(s)_{roll}$ mediante la tecnica della *sintesi in frequenza*.

Partiamo con la definizione delle specifiche che il sistema a ciclo chiuso $W(s)_{roll}$ dovrà rispettare sotto l'azione del controllore $G(s)_{roll}$.

- **Specifiche univoche**
  - Tipologia di sistema desiderato: sistema di tipo $k = 1$
  - Condizione sull'errore di inseguimento a regime: $|e_1(t)| \leq 0.01$
  - Astatismo rispetto a disturbi costanti agenti in catena diretta in uscita dal processo $P(s)_{roll}$
- **Specifiche lasche**
  - Modulo alla risonanza: $M_r \leq 2dB$
  - Banda passante: $B_3 \approx 0.5Hz$

Essendo tali requisiti espressi in catena chiusa è necessario per prima cosa effettuare una conversione in specifiche espresse in catena aperta su cui tale tecnica di sintesi è basata.

Facendo riferimento a quanto discusso nella sotto-sezione 4.2.1, per le specifiche univoche otteniamo:

1. $F(s)_{roll} = G(s)_{roll}P(s)_{roll}$ ha almeno un polo in $s = 0$
2. $|e_1(t)| \leq 0.01 \Rightarrow \dfrac{K_D^2}{K_F} \leq 0.01$. Per $K_D = 1 \Rightarrow |e_1(t)| = \dfrac{1}{K_F} \leq 0.01$
3. Deve essere presente almeno un polo in $s = 0$ 'a monte' del disturbo, ovvero $F(s)_{roll} = G(s)_{roll}P(s)_{roll}$ deve avere almeno un polo in $s = 0$

Invece per le specifiche lasche, dalle relazioni 4.9, 4.10, 4.11 è facile ricavare sia le relative specifiche espresse in catena diretta che alcuni parametri fondamentali nel dominio del tempo t, in particolare:

1. $M_r \leq 2dB \Rightarrow m_\varphi \geq m_{\varphi min} \simeq 45° \Rightarrow \hat{s} \approx 0.7 = 70\%$
2. $B_3 \approx 0.5Hz \Rightarrow \omega_t \approx 2\frac{rad}{s} \Rightarrow t_s \approx 0.95sec$

Prima di procedere con lo sviluppo del controllore $G(s)_{roll}$ andiamo ad analizzare il comportamento in frequenza dell'attuale funzione di trasferimento in catena diretta, rappresentata unicamente dal processo $P(s)_{roll}$, mediante l'utilizzo del *Control System Designer* precedentemente introdotto. Inserendo l'espressione 5.5 all'interno del Toolbox si ottengono i grafici mostrati nella figura 5.2.

Dai grafici in figura 5.2 si osserva la natura stabile del sistema, avendo tutti i poli a parte reale minore-uguale a 0 ma è facile notare come le specifiche richieste non

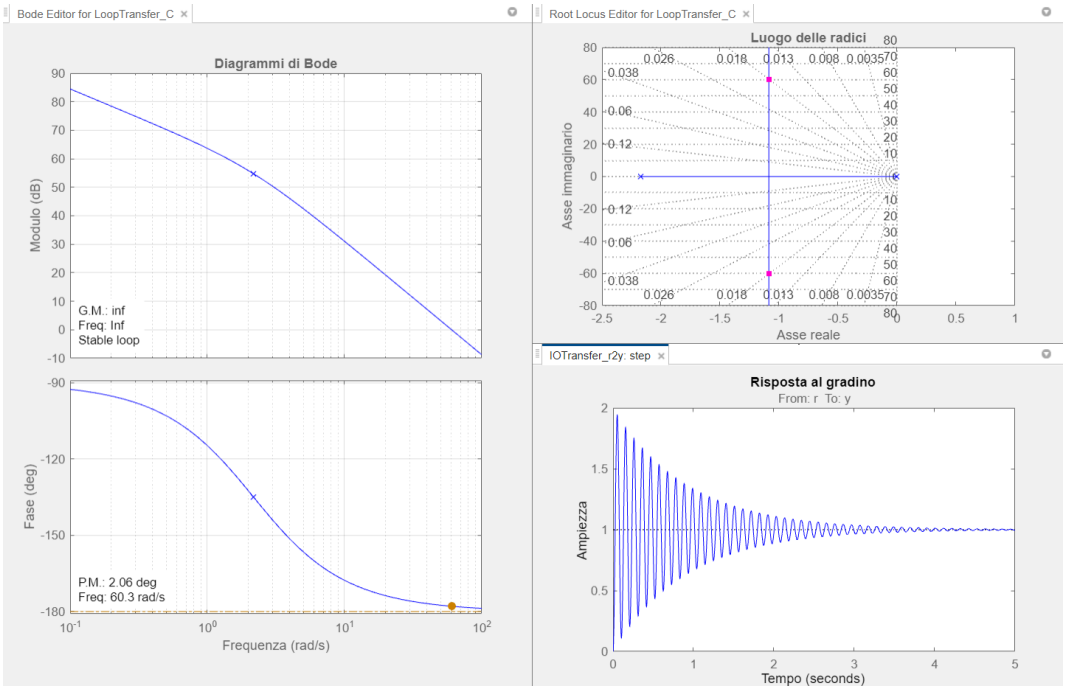


Figura 5.2: Analisi in frequenza processo $P(s)_{roll}$

siano soddisfatte, avendo infatti una pulsazione di attraversamento posizionata in corrispondenza della pulsazione $60.3\frac{rad}{s}$ ben distante dalla ω_t desiderata, pari a $2\frac{rad}{s}$. La risposta al gradino risulta, seppur stabile, non soddisfacente in quanto presenta una forte componente armonica, con un'elevata sovraelongazione e un tempo di assestamento altrettanto elevato. É quindi necessario inserire un controllore atto a migliorare tali caratteristiche in modo da rientrare nei requisiti desiderati.

Controllore di primo tentativo $\hat{G}(s)_{roll}$

Il controllore di primo tentativo ha la funzione di rendere verificate le specifiche univoche mediante un controllore della tipologia 4.12. In questo caso, essendo già presente un polo in $s = 0$ all'interno della funzione di trasferimento del processo $P(s)$, la prima è la terza specifica sono già soddisfatte e di conseguenza il controllore di prima istanza avrà la seguente forma:

$$\hat{G}(s)_{roll} = K_G$$

Dalle seconda specifica sull'errore di inseguimento a regime permanente si ottiene il valore della costante di guadagno K_G, ovvero:

$$|e_1(t)| = \frac{1}{K_F} \leq 0.01 \quad \Rightarrow \quad \frac{1}{K_G K_P} \leq 0.01 \quad \Rightarrow \quad K_G \geq \frac{100}{K_P}$$

Dall'espressione del processo $P(s)_{roll}$ (Eq. 5.5) espressa in *forma canonica di Bode* si ricava che $K_P = \frac{3644}{2.171} = 1678.489175$. Di conseguenza si ottiene dalla precedente

relazione:

$$K_G \geq 0.059577$$

Scegliendo $K_G = 0.06$ si ottiene quindi il controllore di primo tentativo $\hat{G}(s)_{roll} = 0.06$ e di conseguenza la relativa funzione di trasferimento in catena diretta avrà la seguente forma:

$$\hat{F}(s)_{roll} = \hat{G}(S)_{roll} P(s)_{roll} = \frac{0.06 \cdot 3644}{s(s + 2.171)} = \frac{218.64}{s(s + 2.171)} \tag{5.6}$$

Andando ad inserire tale controllore di prima istanza all'interno dello schema di controllo definito nel *Control System Designer* si ottengono i risultati mostrati nella figura 5.3.

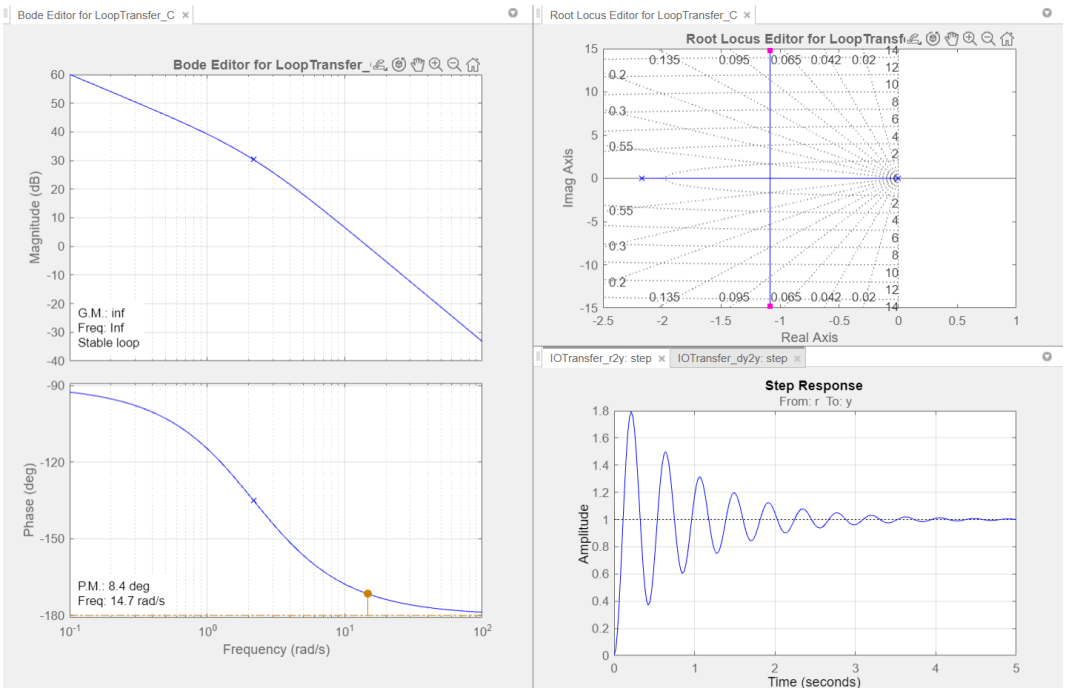
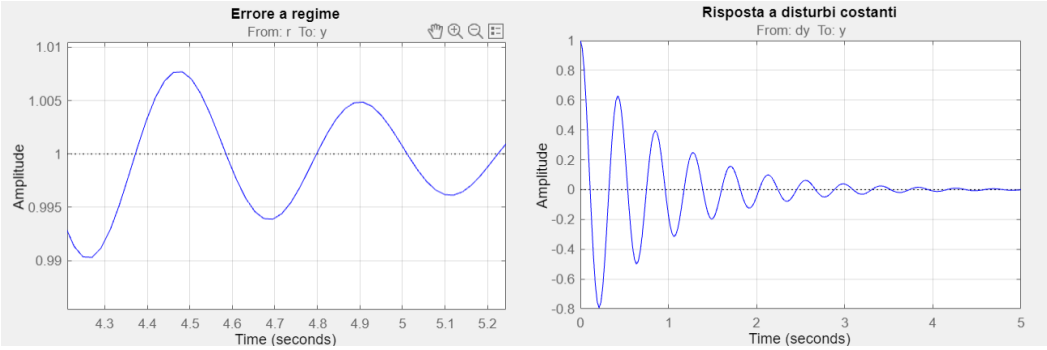


Figura 5.3: Analisi in frequenza $\hat{F}(s)_{roll}$

Si osserva un lieve miglioramento della risposta al gradino grazie ad un'attenuazione della componente armonica nel transitorio e ad un errore di inseguimento a regime dell'ordine di 10^{-3}, come si evince dalla figura 5.4a, in accordo con la seconda specifica univoca richiesta. Tuttavia, tempo di assestamento e sovraelongazione risultano ancora troppo elevati anche a fronte di un tempo di salita accettabile. Per questo motivo sono necessarie ancora delle correzioni al sistema di controllo per far rientrare $F(s)_{roll}$ nelle specifiche richieste.

Dai diagrammi di Bode si osserva che l'attuale pulsazione di attraversamento è situata a $14.7\frac{rad}{s}$, ancora distante dalla ω_t desiderata di $2\frac{rad}{s}$. Infatti, in corrispondenza della pulsazione di attraversamento voluta risulta $|\hat{F}(j\omega_t)_{roll}| \simeq 31.36dB$ e



(a) Grafico errore a regime per $\hat{F}(s)_{roll}$ (b) Risposta a disturbi costanti d_2

Figura 5.4: Errore a regime e astatismo della funzione $\hat{F}(s)_{roll}$

$\angle\hat{F}(j\omega_t) \simeq -132.66°$, mentre i valori desiderati sono $|F(j\omega_t)_{roll}| \simeq 0dB$ per il modulo e $\angle F(j\omega_t) \geq m_{\varphi min} - 180° \simeq -135°$ per la fase. Seppur la specifica in merito alla fase risulti al limite dell'accettabile, ciò non vale per il modulo il quale risulta avere ancora un valore troppo elevato rispetto a quello di riferimento.

Di conseguenza, per soddisfare le specifiche lasche è necessario integrare il controllore di primo tentativo $\hat{G}(s)_{roll}$ con una *rete compensatrice* in modo da attenuare il modulo in corrispondenza della pulsazione $\omega_t = 2\frac{rad}{s}$ di circa $31dB$ senza apportare troppe modifiche alla relativa fase, cercando se possibile di aumentarla in modo da migliorare il margine di fase m_φ e allontanare il sistema da posizioni di instabilità.

Scelta della rete compensatrice $R(s)_{roll}$

Per la progettazione della *sistema compensatore* $R(s)_{roll}$ si utilizzeranno le cosiddette *funzioni anticipatrici* $R_a(s)$ e *funzioni attenuatrici* $R_i(s)$, la cui struttura è stata già introdotta nel capitolo 4 nella sotto-sezione 4.2.1 precisamente nelle equazioni 4.15 e 4.16. Per la scelta delle costanti di tempo τ_a, τ_i e dei parametri caratteristici m_a, m_i si fa riferimento ai diagrammi di Bode di tali funzioni detti *diagrammi universali*, mostrati in figura 4.4, in cui vengono indicati i guadagni in modulo e fase in base alle diverse scelte dei rispettivi parametri τ ed m.

Dopo numerosi tentativi con l'obbiettivo di individuare i valori ottimali per il sistema sotto esame, i parametri scelti sono stati i seguenti:

- **Funzione anticipatrice**

  $\omega_t\tau_a = 1 \quad \Rightarrow \quad \tau_a = \frac{1}{\omega_t} = \frac{1}{2}sec = 0.5sec \qquad \text{con} \qquad m_a = 16$

  Da questi valori si ottiene la seguente funzione anticipatrice $R_a(s)$:

$$R_a(s) = \frac{1 + \tau_a \cdot s}{1 + \frac{\tau_a}{m_a} \cdot s} = \frac{1 + 0.5s}{1 + \frac{0.5}{16}s} = \frac{16(s+2)}{s+32}$$

- **Funzione attenuatrice**

  $\omega_t \tau_i = 400 \quad \Rightarrow \quad \tau_i = \frac{400}{\omega_t} = \frac{400}{2} sec = 200 sec \qquad \text{con} \qquad m_i = 53$

  Da questi valori si ottiene la seguente funzione attenuatrice $R_i(s)$:

$$R_i(s) = \frac{1 + \frac{\tau_i}{m_i} \cdot s}{1 + \tau_i \cdot s} = \frac{1 + \frac{200}{53} s}{1 + 200s} = \frac{\frac{1}{53}(s + 0.265)}{s + 0.005}$$

Quindi, la struttura del controllore complessivo $G(s)_{roll}$ è la seguente:

$$G(s)_{roll} = \hat{G}(s)_{roll} \cdot R_a(s) \cdot R_i(s) = \hat{G}(s)_{roll} \cdot R(s) = \frac{\frac{24}{1325}(s+2)(s+0.265)}{(s+32)(s+0.005)} \tag{5.7}$$

a cui segue la funzione di trasferimento in catena aperta $F(s)_{roll}$:

$$F(s)_{roll} = G(s)_{roll} \cdot P(s)_{roll} = \frac{66.005(s+2)(s+0.265)}{s(s+2.171)(s+32)(s+0.005)} \tag{5.8}$$

I risultati ottenuti durante l'analisi in frequenza della funzione di trasferimento in catena diretta $F(s)_{roll}$ sono mostrati nella figura 5.5.

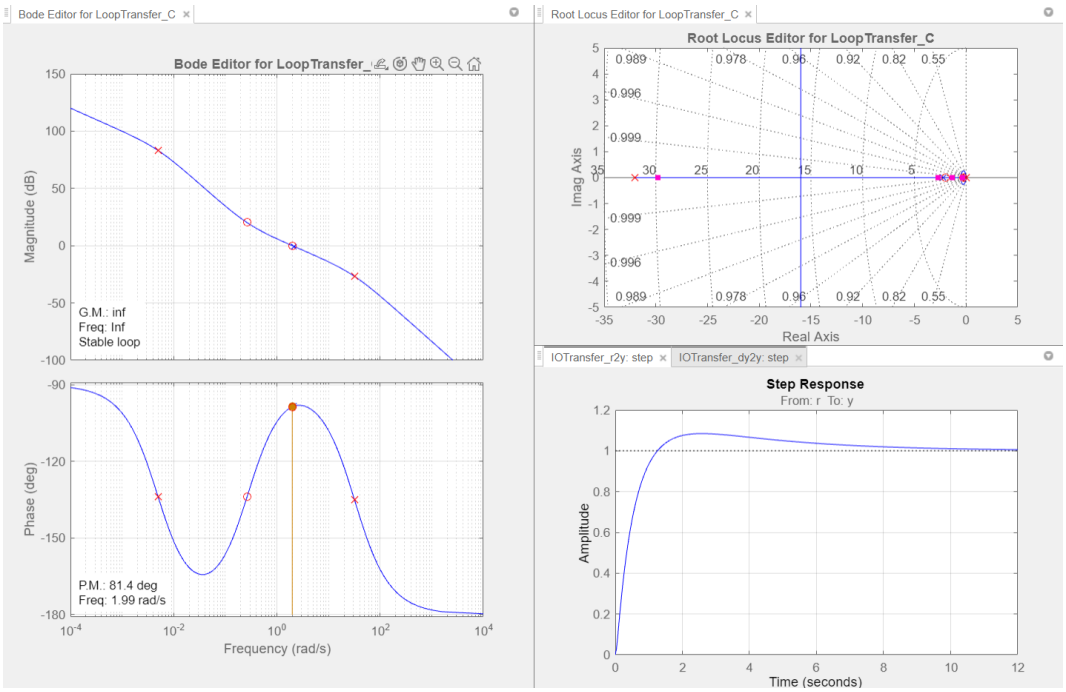


Figura 5.5: Analisi in frequenza della funzione in catena diretta $F(s)_{roll}$

Come si può osservare dalla 5.5 il nuovo sistema in catena diretta soddisfa anche le specifiche lasche. Infatti, l'attuale pulsazione di attraversamento è posizionata precisamente alla pulsazione ω_t desiderata, ovvero $\omega_t = 2\frac{rad}{s}$, con un margine di fase m_φ pari a 81.4° ben superiore rispetto al margine di fase minimo $m_{\varphi min} \simeq 45°$.

Per quanto riguarda le specifiche espresse nel dominio del tempo, anch'esse sono pienamente soddisfatte, osservando infatti un tempo di salita $t_s = 0.826sec$ e una sovraelongazione $\hat{s} = 8.45\%$ all'istante $t = 2.59sec$, in cui si misura un'ampiezza massima pari a 1.08. Quindi tali valori rientrano a pieno nei requisiti richiesti e di conseguenza il controllore sviluppato può considerarsi valido.

Nonostante il soddisfacimento delle specifiche lasche risultano ancora verificate anche le specifiche univoche, specie quella in merito all'errore di inseguimento $e_1(t)$. Infatti, risultando $K_F \simeq 100.71$ (si ottiene scrivendo la 5.8 in *forma canonica di Bode*) il valore finale dell'errore di inseguimento dopo la progettazione del controllore $G(s)_{roll}$ è:

$$|e_1(t)| \leq 0.01 \quad \Rightarrow \quad \frac{1}{K_F} \leq 0.01 \quad \Rightarrow \quad \frac{1}{100.71} \simeq 9.930 \times 10^{-3} \leq 0.01$$

Per concludere il processo di sintesi occorre verificare che siano soddisfatte anche le specifiche in catena chiusa mediante l'utilizzo della *carta di Nichols*, i cui risultati sono osservabili dalla figura 5.6.

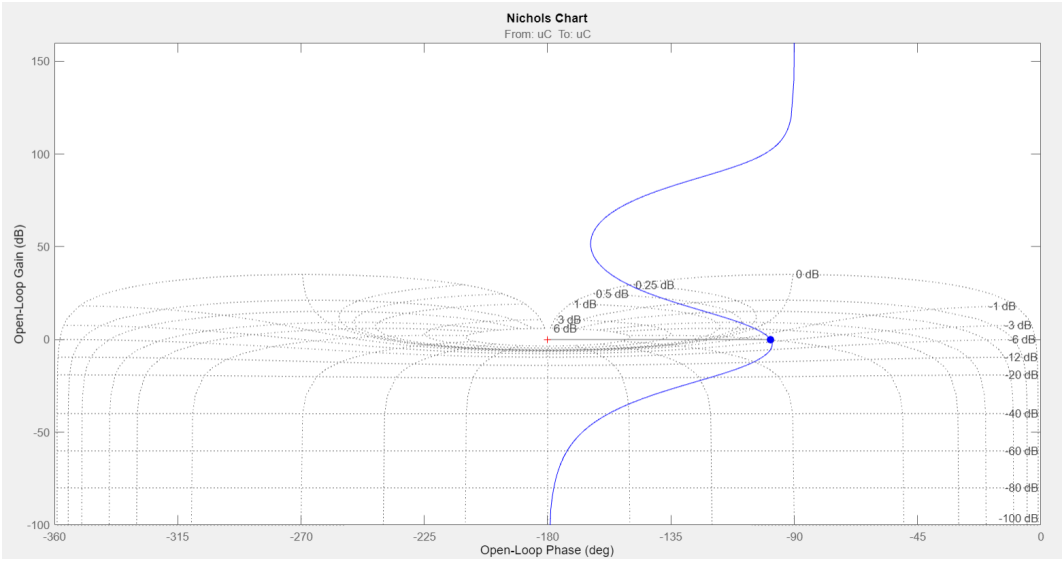


Figura 5.6: Diagramma di Nichols della funzione in catena chiusa $W(s)_{roll}$

É facile verificare che anche le specifiche in merito al modulo alla risonanza M_r e alla banda passante B_3 sono soddisfatte e di conseguenza possiamo considerare conclusa la sintesi del sistema di controllo per l'angolo di *roll*.

L'implementazione di tale controllore all'interno del progetto *asbQuadcopter* di *Simulink* è argomento del capitolo 6.

5.2 Sintesi in frequenza del controllore per l'angolo di *yaw*

Per la sintesi del regolatore lineare utilizzato nel controllo dell'angolo di *yaw*, ovvero dell'angolo di inclinazione ψ attorno l'asse z o *asse di imbardata*, è stata utilizzata come tecnica di controllo la *sintesi in frequenza*, come per il controllo del *roll*. Essendo la logica di progettazione analoga al al caso precedentemente affrontato, per il seguente processo di sintesi non verranno analizzate nel dettaglio tutte le caratteristiche e le fasi progettuali in quanto una spiegazione rigorosa è già presente nella sezione 5.1. Quindi, per maggiori dettagli in merito ai controllori sviluppati mediante tale tecnica di controllo, si faccia riferimento alla sezione appena citata e alla sotto-sezione 4.2.1, contenente tutte le caratteristiche da tenere a mente per la progettazione di controllori tramite l'analisi in frequenza.

Procedendo con la progettazione del regolatore lineare, la prima cosa da fare è definire la funzione di trasferimento del processo $P(s)_{yaw}$ da controllare, nel nostro caso rappresentato dalla dinamica dell'angolo di *yaw*.

Facendo riferimento alla struttura dei vettori contenenti le variabili di stato, ingresso e uscita complessive del sistema in *spazio di stato* relativo al minidrone *Parrot Mambo* (Figure 3.5, 3.6, 3.7), le componenti rilevanti per lo studio della dinamica dell'angolo di *yaw* sono:

- Vettore degli stati: posizione angolare ψ e velocità angolare r dello *yaw*, ovvero le componenti **phi theta psi(3)** e **p,q,r(3)**.
- Vettore degli ingressi: la spinta esercitata da ciascun motore, ovvero le componenti **Actuators(1)**, **Actuators(2)**, **Actuators(3)** e **Actuators(4)**.
- Vettore delle uscite: l'angolo di Eulero relativo allo *yaw*, ovvero la componente **Euler(3)**.

Di conseguenza, i corrispettivi coefficienti selezionati dalle matrici standard complessive A, B, C, D, riportate in figura 3.1, 3.2, 3.3, 3.4, portano alla formazione delle seguenti "**matrici standard ridotte per l'angolo di *yaw***":

$$A_{yaw} = \begin{bmatrix} 0 & 1 \\ 0 & -6.8807 \times 10^{-9} \end{bmatrix} \quad B_{yaw} = \begin{bmatrix} 0 \\ -0.0115 \end{bmatrix} \quad C_{yaw} = \begin{bmatrix} 1 & 0 \end{bmatrix} \quad D_{yaw} = \begin{bmatrix} 0 \end{bmatrix}$$

Il sistema in *spazio di stato* associato a tali matrici $A_{yaw}, B_{yaw}, C_{yaw}, D_{yaw}$ risulta quindi avere la seguente forma:

$$\begin{cases} \dot{x} = \begin{bmatrix} 0 & 1 \\ 0 & -6.8807 \times 10^{-9} \end{bmatrix} x + \begin{bmatrix} 0 \\ -0.0115 \end{bmatrix} u \\ y = \begin{bmatrix} 1 & 0 \end{bmatrix} x \end{cases} \tag{5.9}$$

$$\text{con} \quad x = \begin{bmatrix} \text{phi theta psi(3)} \\ \text{p,q,r(3)} \end{bmatrix} \quad \text{e} \quad u = \text{Actuators(i)} \quad \text{con} \quad i = 1, 2, 3, 4.$$

Una volta individuate le matrici standard relative alla dinamica dello *yaw*, è necessario trovare la relativa funzione di trasferimento del processo $P(s)_{yaw}$ da analizzare. Per calcolarla, come già visto nella sintesi del controllore per l'angolo di *roll*, è necessario considerare anche il contributo delle costanti moltiplicative K_1 e K_2, i cui valori sono riportati nella figura 5.1. Esse rappresentano rispettivamente il peso attraverso il quale lo sforzo di controllo viene ripartito tra i 4 attuatori e il coefficiente di spinta di ciascuno di essi, considerato identico per tutti.

Dovendo considerare solamente le componenti relative allo studio della dinamica dell'angolo di *yaw*, i valori scelti sono necessariamente:

$$K_1 = -103.5736 \quad K_2 = 1.5307 \times 10^3$$

Ottenuti i valori di tali costanti moltiplicative è ora possibile calcolare la funzione di trasferimento relativa al processo $P(s)_{yaw}$, ottenibile digitando i seguenti comandi all'interno della Command Window di MATLAB:

```
» A_yaw = [0, 1; 0, -6.8807e-09];
» B_yaw = [0; -0.0115];
» C_yaw = [1, 0];
» D_yaw = [0];
» K1_yaw = -103.5736;
» K2 = 1.5307e+03;
» sys_yaw = ss(A_yaw, B_yaw, C_yaw, D_yaw);
» plant_yaw = tf(sys_yaw);
» Ps_yaw = K1_yaw * K2 * plant_yaw
```

La funzione di trasferimento associata alla dinamica dello *yaw* ha quindi la seguente espressione:

$$\boldsymbol{P(s)_{yaw} = \frac{1823}{s(s + 6.881 \times 10^{-9})} \quad \Rightarrow \quad P(s)_{yaw} \approx \frac{1823}{s^2}} \tag{5.10}$$

sfruttando l'approssimazione $6.881 \times 10^{-9} \approx 0$.

In fase di sintesi verrà utilizzata l'espressione approssimata mentre per la verifica dei requisiti si considererà invece l'espressione completa, in modo tale da non introdurre errori legati a troncamenti o semplificazioni e per verificare il corretto funzionamento del controllore sviluppato sul processo reale (non approssimato).

Trovato il processo $P(s)_{yaw}$ da controllare è ora possibile procedere con la sintesi del controllore $G(s)_{yaw}$.

Le specifiche che il sistema a ciclo chiuso $W(s)_{yaw}$ dovrà rispettare sotto l'azione del controllore sono:

- **Specifiche univoche**
  - Tipologia di sistema desiderato: sistema di tipo $K = 1$
  - Condizione sull'errore di inseguimento a regime: $|e_1(t)| \leq 0.01$
  - Astatismo rispetto disturbi costanti agenti in catena diretta in uscita dal processo $P(s)_{yaw}$
- **Specifiche lasche**
  - Modulo alla risonanza: $M_r \leq 2dB$
  - Banda passante: $B_3 \approx 2Hz$

Effettuando una conversione da specifiche in catena chiusa a specifiche in catena aperta si ottiene per le specifiche univoche:

1. $F(s)_{yaw}$ ha almeno un polo in $s = 0$
2. $|e_1(t)| = \frac{1}{K_F} \leq 0.01$
3. Deve essere presente almeno un polo in $s = 0$ 'a monte' del disturbo, ovvero in $F(s)_{yaw}$

mentre, per le specifiche lasche:

1. $M_r \leq 2dB \Rightarrow m_\varphi \geq m_{\varphi min} \simeq 45° \Rightarrow \hat{s} \approx 0.7 = 70\%$
2. $B_3 \approx 2Hz \Rightarrow \omega_t \approx 8\frac{rad}{s} \Rightarrow t_s \approx 0.24sec$

Definite le specifiche che il sistema a ciclo chiuso dovrà rispettare, andiamo ora a visualizzare il comportamento in frequenza del processo $P(s)_{yaw}$ privo di controllore, riportato in figura 5.7.

Dalla figura è facile notare la natura stabile del processo, ma a causa di un margine di fase $m_\varphi \simeq 0$ il sistema è molto vicino alla posizione di transizione da stabilità a instabilità, ovvero il cosiddetto *punto critico* in corrispondenza del quale si osserva un margine di fase $m_\varphi = 0$. L'influenza di tale posizione di criticità si riscontra anche nella risposta al gradino, che come si può notare assume un comportamento puramente armonico senza mai stabilizzarsi al valore di riferimento.

Andiamo quindi a progettare un controllore atto a risolvere tali problematiche e a rendere il sistema a ciclo chiuso rientrante nei requisiti richiesti.

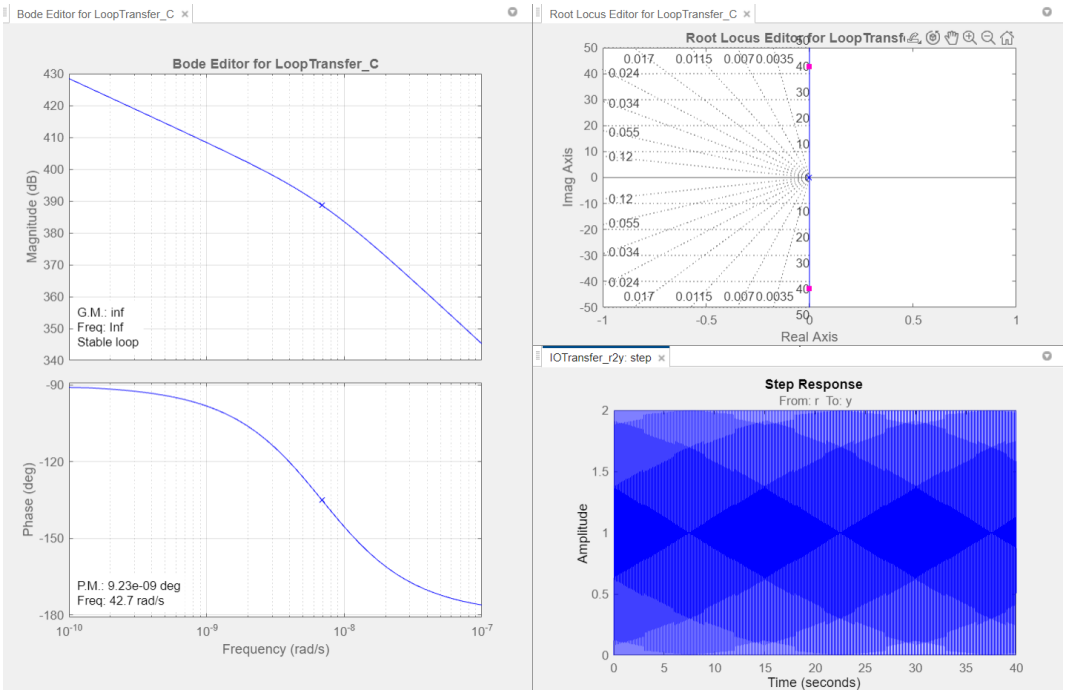


Figura 5.7: Analisi in frequenza processo $P(s)_{yaw}$

Controllore di primo tentativo $\hat{G}(s)_{yaw}$

Essendo la prima e la terza specifica univoca già soddisfatte ($P(s)_{yaw}$ ha un polo in $s = 0$) il controllore di prima istanza sarà della forma $\hat{G}(s)_{yaw} = K_G$ dove la costante di guadagno statico del controllore K_G si ricava dalla seconda specifica univoca, in particolare:

$$\frac{1}{K_F} \leq 0.01 \quad \Rightarrow \quad \frac{1}{K_G K_P} \leq 0.01 \quad \Rightarrow \quad K_G \geq \frac{100}{K_P}$$

In questo caso il guadagno $K_P = 1823$ e di conseguenza si ottiene:

$$K_G \geq 0.0548546$$

Scegliamo per semplicità $K_G = 0.06$ ottenendo il controllore di primo tentativo $\hat{G}(s)_{yaw} = 0.06$ e di conseguenza la funzione di trasferimento in catena diretta di prima istanza (considerando l'espressione del processo non approssimata) sarà:

$$\hat{F}_{yaw}(s) = \frac{109.38}{s(s + 6.881 \times 10^{-9})} \tag{5.11}$$

Inserendo tale controllore $\hat{G}(s)_{yaw}$ nello schema di controllo definito in ambiente MATLAB si osserva il comportamento riportato in figura 5.8.

Analizzando la figura 5.8, con l'applicazione del controllore $\hat{G}(s)_{yaw}$ si è conservata la stabilità del sistema a ciclo chiuso a fronte però di un aumento della componente armonica nella risposta al gradino, dovuta ad una diminuzione del guadagno statico

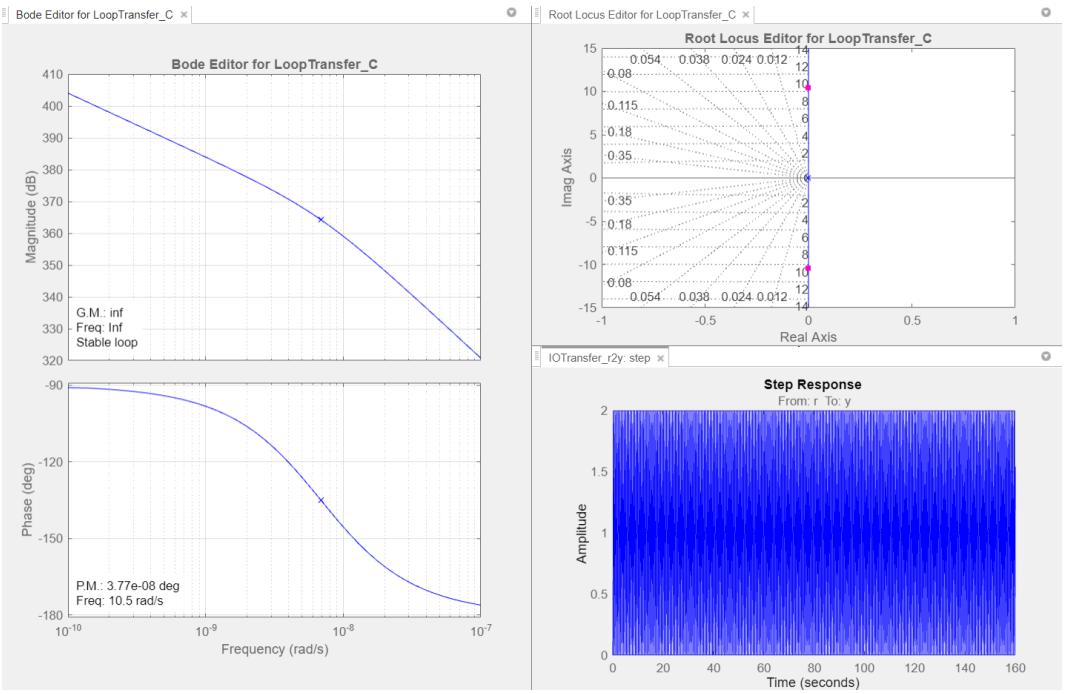


Figura 5.8: Analisi in frequenza $\hat{F}(s)_{yaw}$

del sistema in catena diretta. In realtà, a causa di un margine di fase prossimo a 0° il sistema è ancora molto vicino al *punto critico* e di conseguenza piccolissime variazioni di modulo e fase potrebbero portare il sistema in una posizione di instabilità. Anche per la risposta a disturbi costanti agenti in uscita dal processo il discorso è analogo: una forte componente armonica, simile a quella della risposta al gradino, non consente di ottenere astatismo in tempi accettabili.

É quindi necessario progettare una rete di controllo atta a compensare tale comportamento armonico e a far rientrare il sistema nelle specifiche lasche, attualmente non verificate. Infatti, in corrispondenza della pulsazione di attraversamento desiderata $\omega_t = 8\frac{rad}{s}$ i valori di modulo e fase misurati risultano essere rispettivamente $|\hat{F}(j\omega_t)| \simeq 4.65dB$ e $\angle\hat{F}(j\omega_t) \simeq -180°$, quando quelli voluti sono $|F(j\omega_t) \simeq 0dB$ per il modulo e $\angle F(j\omega_t) \geq m_{\varphi min} - 180° \simeq 135°$ per la fase.

Scelta della rete compensatrice $R(s)_{yaw}$

Per la progettazione del *sistema compensatore* $R(s)_{yaw}$ si utilizzeranno una *funzione anticipatrice* per ottenere un aumento di fase di almeno 45° e una *funzione attenuatrice* in modo da diminuire il modulo della funzione di trasferimento in catena diretta di circa $5dB$ senza però apportare ingenti modifiche al valore della fase. Il risultato ottenuto sarà lo spostamento della pulsazione di attraversamento (attualmente posizionata a $10.5\frac{rad}{s}$) al valore desiderato $\omega_t = 8\frac{rad}{s}$ e un margine di fase m_φ tale da allontanare il più possibile il sistema dal *punto critico* e quindi da possibili posizioni di instabilità.

Dopo numerose prove con l'obbiettivo di individuare le migliori funzioni da utilizzare per la formazione del presente sistema di controllo, sfruttando i *diagrammi universali*, i valori dei parametri selezionati sono stati:

- **Funzione anticipatrice**

  $\omega_t \tau_a = 3 \quad \Rightarrow \quad \tau_a = \frac{3}{\omega_t} = \frac{3}{8} sec = 0.375 sec \qquad \text{con} \qquad m_a = 20$

  Da questi valori si ottiene la seguente funzione anticipatrice $R_a(s)$:

$$R_a(s) = \frac{1 + \tau_a \cdot s}{1 + \frac{\tau_a}{m_a} \cdot s} = \frac{1 + 0.375s}{1 + \frac{0.375}{20} s} = \frac{20(s + \frac{8}{3})}{s + \frac{160}{3}}$$

- **Funzione attenuatrice**

  $\omega_t \tau_i = 200 \quad \Rightarrow \quad \tau_i = \frac{200}{\omega_t} = \frac{200}{8} sec = 25 sec \qquad \text{con} \qquad m_i = 5.35$

  Da questi valori si ottiene la seguente funzione attenuatrice $R_i(s)$:

$$R_i(s) = \frac{1 + \frac{\tau_i}{m_i} \cdot s}{1 + \tau_i \cdot s} = \frac{1 + \frac{500}{107} s}{1 + 25s} = \frac{\frac{20}{107}(s + 0.214)}{s + 0.04}$$

La struttura del controllore complessivo $G(s)_{yaw}$ sarà quindi la seguente:

$$G(s)_{yaw} = \hat{G}(s)_{yaw} \cdot R_a(s) \cdot R_i(s) = \hat{G}(s)_{yaw} \cdot R(s) = \frac{\frac{24}{107}(s + \frac{8}{3})(s + 0.214)}{(s + \frac{160}{3})(s + 0.04)} \quad (5.12)$$

a cui segue la funzione di trasferimento in catena aperta $F(s)_{yaw}$:

$$F(s)_{yaw} = \frac{\frac{43752}{107}(s + \frac{8}{3})(s + 0.214)}{s(s + 6.881 \times 10^{-9})(s + \frac{160}{3})(s + 0.04)} \quad (5.13)$$

I risultati ottenuti durante l'analisi in frequenza della funzione di trasferimento in catena diretta $F(s)_{yaw}$ sono mostrati nella figura 5.9.

Osservando la figura 5.9, si può notare come il sistema in catena diretta finale soddisfa anche le specifiche lasche. Infatti, la pulsazione di attraversamento è posizionata precisamente in corrispondenza di quella desiderata ($8 \frac{rad}{s}$), con un margine di fase m_φ pari a 61.8°, superiore al valore limite $m_{\varphi min} \simeq 45°$.

Si sono ottenute ottime prestazioni anche per i parametri espressi nel dominio del tempo. Infatti, i valori misurati tramite MATLAB mostrano un tempo di salita $t_s = 0.15 sec$, una sovraelongazione $\hat{s} = 21\%$ e un tempo di assestamento $t_a = 1.06 sec$. Quindi, tutti i valori rientrano a pieno nei requisiti richiesti.

Per concludere il processo di sintesi occorre verificare che siano soddisfatte anche le specifiche in catena chiusa mediante l'utilizzo della *carta di Nichols*, i cui risultati sono mostrati nella figura 5.10.

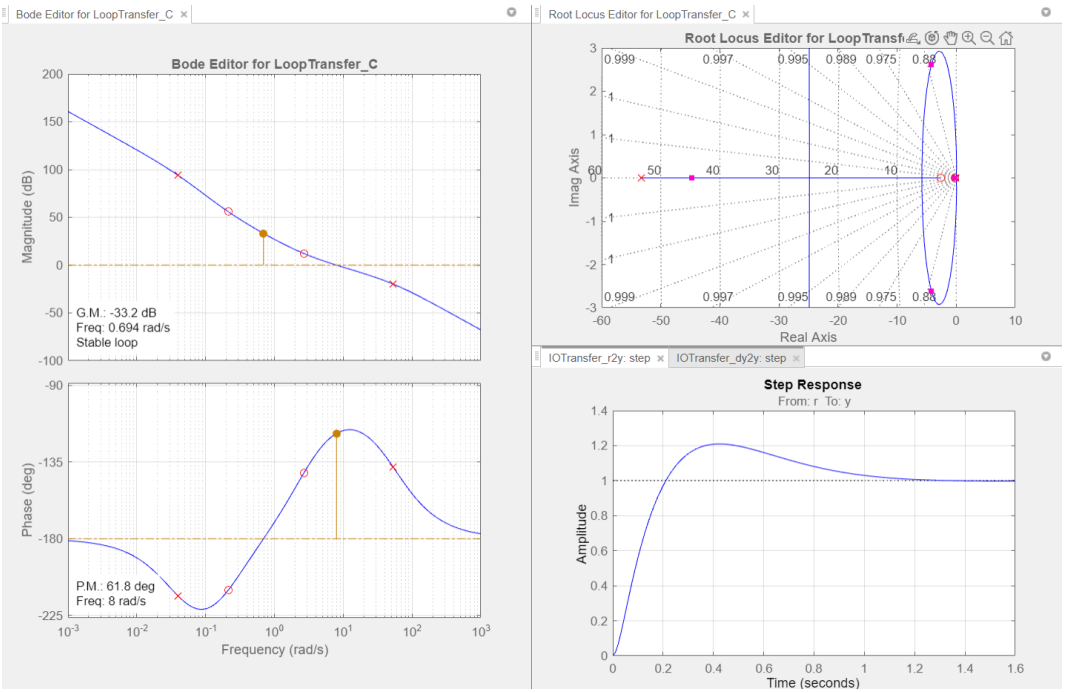


Figura 5.9: Analisi in frequenza della funzione in catena diretta $F(s)_{yaw}$

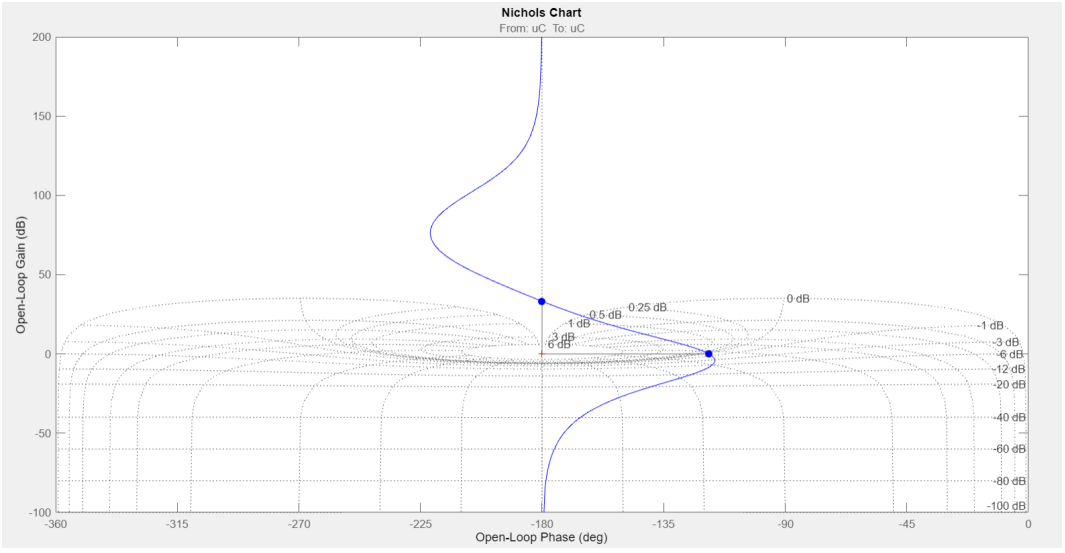


Figura 5.10: Diagramma di Nichols della funzione in catena chiusa $W(s)_{yaw}$

É facile verificare come anche le specifiche relative al modulo alla risonanza M_r e alla banda passante B_3 sono soddisfatte e di conseguenza possiamo considerare concluso il processo di sintesi.

L'implementazione di tale controllore all'interno del progetto *asbQuadcopter* di *Simulink* è argomento del capitolo 6.

5.3 Sintesi con luogo delle radici del controllore per l'angolo di *pitch*

Per la sintesi del regolatore lineare utilizzato nel controllo dell'angolo di *pitch*, ovvero dell'angolo di inclinazione θ attorno l'asse x o *asse di beccheggio*, è stata utilizzata come tecnica di controllo la *sintesi tramite luogo delle radici*. Le specifiche che il sistema a ciclo chiuso dovrà rispettare saranno simili a quelle già introdotte nelle sezioni 5.1, 5.2 ma per quanto riguarda la logica di sintesi essa risulta essere leggermente differente in quanto questa metodologia non lavora nel dominio della frequenza bensì sfrutta il *piano di Gauss* o *piano complesso* per rappresentare il luogo dei punti occupati dai poli della funzione di trasferimento in catena chiusa $W(s)$ al variare del coefficiente di guadagno K relativo alla funzione di trasferimento in catena diretta $F(S)$ (in generale diverso da K_F). Per maggiori dettagli si faccia riferimento alla sotto-sezione4.2.2.

Procedendo con la sintesi, la prima cosa da fare è definire la funzione di trasferimento del processo $P(s)_{pitch}$ da controllare. É sufficiente fare riferimento alla struttura dei vettori di stato, ingresso e uscita relativi al modello in *spazio di stato* del minidrone *Parrot Mambo*, la cui struttura è riportata nelle figure 3.5, 3.6, 3.7. Le componenti rilevanti da selezionare per il presente sistema di controllo sono:

- Vettore degli stati: posizione angolare θ e velocità angolare q del *pitch*, ovvero le componenti **phi theta psi(2)** e **p,q,r(2)**.
- Vettore degli ingressi: la spinta esercitata da ciascun motore, ovvero le componenti **Actuators(1), Actuators(2), Actuators(3), Actuators(4)**.
- Vettore delle uscite: l'angolo di Eulero relativo al *pitch*, ovvero la componente **Euler(2)**.

Facendo quindi riferimento alle matrici standard del sistema complessivo A, B, C, D, riportate nelle figure 3.1, 3.2, 3.3, 3.4, è possibile ricavare i rispettivi coefficienti che portano alla formazione delle seguenti "**matrici standard ridotte per l'angolo di *pitch***":

$$A_{pitch} = \begin{bmatrix} 0 & 1 \\ 0 & -1.6192 \end{bmatrix} \quad B_{pitch} = \begin{bmatrix} 0 \\ 0.3133 \end{bmatrix} \quad C_{pitch} = \begin{bmatrix} 1 & 0 \end{bmatrix} \quad D_{pitch} = \begin{bmatrix} 0 \end{bmatrix}$$

Il sistema in *spazio di stato* associato a tali matrici $A_{pitch}, B_{pitch}, C_{pitch}, D_{pitch}$ risulta quindi avere la seguente forma:

$$\begin{cases} \dot{x} = \begin{bmatrix} 0 & 1 \\ 0 & -1.6192 \end{bmatrix} x + \begin{bmatrix} 0 \\ 0.3133 \end{bmatrix} u \\ y = \begin{bmatrix} 1 & 0 \end{bmatrix} x \end{cases} \tag{5.14}$$

$$\text{con} \quad x = \begin{bmatrix} \text{phi theta psi(2)} \\ \text{p,q,r(2)} \end{bmatrix} \quad \text{e} \quad u = \text{Actuators(i)} \quad \text{con} \quad i = 1, 2, 3, 4.$$

Per il calcolo delle funzione di trasferimento associata al processo $P(s)_{pitch}$ bisogna considerare il contributo delle costanti moltiplicative K_1 e K_2, rappresentanti rispettivamente il peso con cui lo sforzo di controllo viene ripartito tra i 4 attuatori e il coefficiente di spinta di ciascuno di essi, considerato identico per tutti. Nel seguente caso, i valori selezionati sono necessariamente:

$$K_1 = 5.6659 \quad K_2 = 1.5307 \times 10^3$$

Digitando i seguenti comandi all'interno della Command Window di MATLAB si ottiene il risultato desiderato.

```
» A_pitch = [0, 1; 0, -1.6192];
» B_pitch = [0; 0.3133];
» C_pitch = [1, 0];
» D_pitch = [0];
» K1_pitch = 5.6659;
» K2 = 1.5307e+03;
» sys_pitch = ss(A_pitch, B_pitch, C_pitch, D_pitch);
» plant_pitch = tf(sys_pitch);
» Ps_pitch = K1_pitch * K2 * plant_pitch
```

La funzione di trasferimento associata alla dinamica del *pitch* ha quindi la seguente espressione:

$$\boldsymbol{P(s)_{pitch} = \frac{2717}{s(s + 1.619)}} \tag{5.15}$$

Procediamo quindi con la sintesi del controllore $G(s)_{pitch}$ partendo con la definizione delle specifiche che il sistema a ciclo chiuso dovrà rispettare:

- **Specifiche univoche**
  - Tipologia di sistema desiderato: sistema di tipo $K = 1$

  - Astatismo rispetto disturbi costanti agenti in catena diretta in uscita dal processo $P(s)_{pitch}$.

- **Specifiche lasche**
  - i poli p_i della funzione di trasferimento in catena chiusa devono avere tutti parte reale minore di -1, ovvero $\Re(p_i) \leq -1 \quad \forall i$

Controllore di primo tentativo $\hat{G}(s)_{pitch}$

Dalle definizione delle specifiche univoche in merito alla tipologia di sistema desiderato e all'astatismo rispetto disturbi costanti agenti sull'uscita del processo, risulta quindi necessario che la funzione di trasferimento $F(s)_{pitch}$ abbia almeno un polo in $s = 0$. Poiché è già presente nella struttura del processo $P(s)_{pitch}$ entrambe le specifiche risultano a priori soddisfatte e di conseguenza la struttura del controllore di prima istanza $\hat{G}(s)_{pitch}$ sarà del tipo:

$$\hat{G}(s)_{pitch} = K$$

dove K è il *coefficiente di guadagno in catena diretta*, legato alla costante K_F dalla relazione 4.22. Di conseguenza la funzione di trasferimento in catena diretta di primo tentativo $\hat{F}(s)_{pitch}$ avrà la seguente forma:

$$\hat{F}(s)_{pitch} = \hat{G}(S)_{pitch} \cdot P(s)_{pitch} = K \frac{2717}{s(s + 1.619)}$$

Con l'ausilio del *Control System Designer* di MATLAB andiamo a visualizzare il luogo delle radici del polinomio caratteristico $f(s, K)$ della funzione di trasferimento in catena chiusa $W(s)_{pitch}$, in figura 5.11.

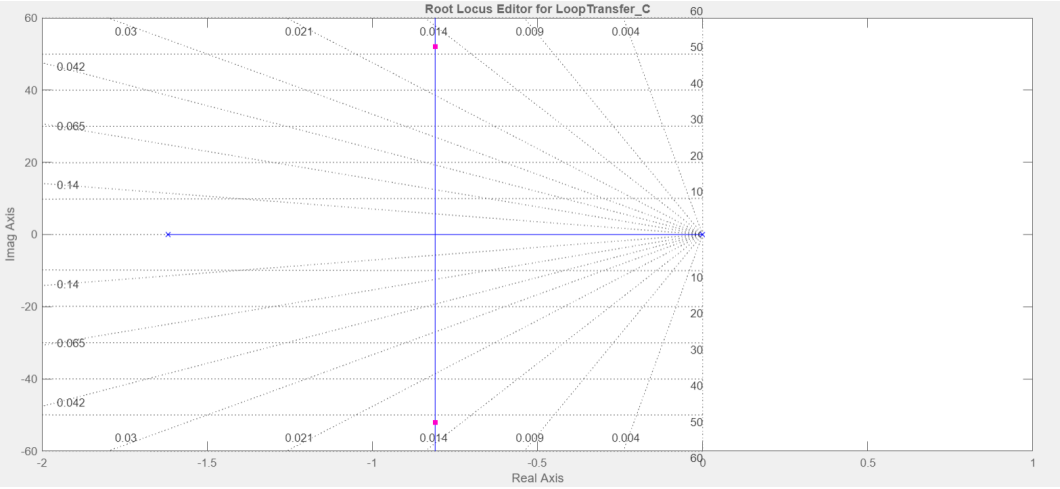


Figura 5.11: Luogo delle radici relativo a $\hat{F}(s)_{pitch}$

Dalla figura 5.11 è facile notare la presenza di due poli stabili e nessuno zero per la funzione di trasferimento in catena diretta, in particolare $p_1 = 0$ e $p_2 = -1.619$. Di

conseguenza si ha $n = 2$ e $m = 0$, ottenendo quindi $n - m = 2$. Da questo possiamo affermare che il luogo delle radici sarà costituito da un totale di $n = 2$ rami entrambi tendenti a degli asintoti al variare di K, il cui centro s_0 risiede sull'asse reale nel punto:

$$s_0 = \frac{\sum_{i=1}^{n} p_i - \sum_{i=1}^{m} z_i}{n - m} = \frac{0 - 1.619}{2 - 0} = -0.8095 \approx -0.81$$

Dalla figura è facile individuare il loro valore, in particolare risulta $\varphi_0^- = 0$ e $\varphi_1^- = \frac{\pi}{2}$ per il luogo negativo e $\varphi_0^+ = \pi$ e $\varphi_1^+ = \frac{3}{2}\pi$ per il luogo positivo.

Per il soddisfacimento delle specifiche lasche in merito al posizionamento nella regione desiderata dei poli a ciclo chiuso è quindi necessario spostare il centro degli asintoti il una posizione diversa da quella attuale, la quale non permette per nessun valore di K di posizionare tutti i poli della $W(s)_{pitch}$ nella regione $\Re(p_i) \leq -1$.

Andiamo quindi a progettare una funzione compensatrice $R(s)_{pitch}$ che consenta di spostare il centro degli asintoti s_0 in una posizione appartenente al semipiano desiderato.

Scelta della rete compensatrice $R(s)_{pitch}$

La struttura della rete compensatrice dovrà essere tale da mantenere costante e al valore minimo lo scarto poli-zeri della funzione di trasferimento in catena diretta $F(s)_{pitch}$, in modo tale da non aggiungere ulteriori rami al luogo delle radici e non complicare ulteriormente il modello complessivo. Poiché l'attuale scarto poli-zeri risulta essere pari a 2, di conseguenza una buona scelta progettuale è quella di inserire una rete compensatrice con un numero di poli uguale al numero di zeri, così da lasciare inalterata le differenza $n - m$. Per non introdurre eccessivi poli e zeri nella struttura complessiva, la rete compensatrice $R(s)_{pitch}$ sarà quindi del tipo:

$$R(s)_{pitch} = \frac{s - z_1}{s - p_1}$$

in cui sono presenti $n = 1$ poli ed $m = 1$ zeri. La scelta dei rispettivi valori dipende dal posizionamento desiderato del centro degli asintoti s_0, il quale come già anticipato deve cadere nel semipiano sinistro in cui $\Re(s) \leq -1$.

Per la scelta di p_1 e z_1 deve quindi essere necessariamente verificata la seguente condizione:

$$s_0 = \frac{\sum_{i=1}^{n} p_i - \sum_{i=1}^{m} z_i}{n - m} = \frac{0 - 1.619 + p_1 - z_1}{3 - 1} = \frac{-1.619 + p_1 - z_1}{2} < -3 \qquad (5.16)$$

Con l'obbiettivo di individuare i valori ottimali per la progettazione della rete compensatrice, che rispettino la condizione 5.16 appena enunciata, le posizioni scelte per la coppia z_1 - p_1 sono state:

$$z_1 = -3 \qquad p_1 = -60$$

dalle quali si ottiene la seguente funzione compensatrice:

$$R(s)_{pitch} = \frac{s+3}{s+60} \tag{5.17}$$

Possiamo quindi definire la funzione di trasferimento del controllore $G(s)_{pitch}$:

$$G(s)_{pitch} = \hat{G}(s)_{pitch} \cdot R(s)_{pitch} = K\frac{s+3}{s+60}$$

a cui segue la seguente struttura della funzione in catena diretta $F(s)_{pitch}$:

$$F(s)_{pitch} = G(s)_{pitch} \cdot P(s)_{pitch} = K\frac{2717(s+3)}{s(s+1.619)(s+60)}$$

La scelta del *coefficiente di guadagno in catena diretta* K è stata fatta in funzione del posizionamento desiderato dei poli in catena chiusa. In particolare, il Toolbox *Control System Designer* mette a disposizione una funzionalità che consente di spostare in maniera interattiva direttamente sul luogo delle radici i poli della $W(s)$, con il conseguente calcolo automatico della costante K relativa a tali posizioni scelte. Con l'obbiettivo di posizionare tutti i poli in catena chiusa nella regione di piano a parte reale minore di -1, il coefficiente di guadagno in catena diretta scelto è stato $K = 0.001$. Di conseguenza il sistema di controllo finale ha la seguente espressione:

$$G(s)_{pitch} = \frac{0.02(s+3)}{s+60} \tag{5.18}$$

In figura 5.12 vieni mostrato il luogo delle radici dopo l'applicazione del controllore $G(s)_{pitch}$.

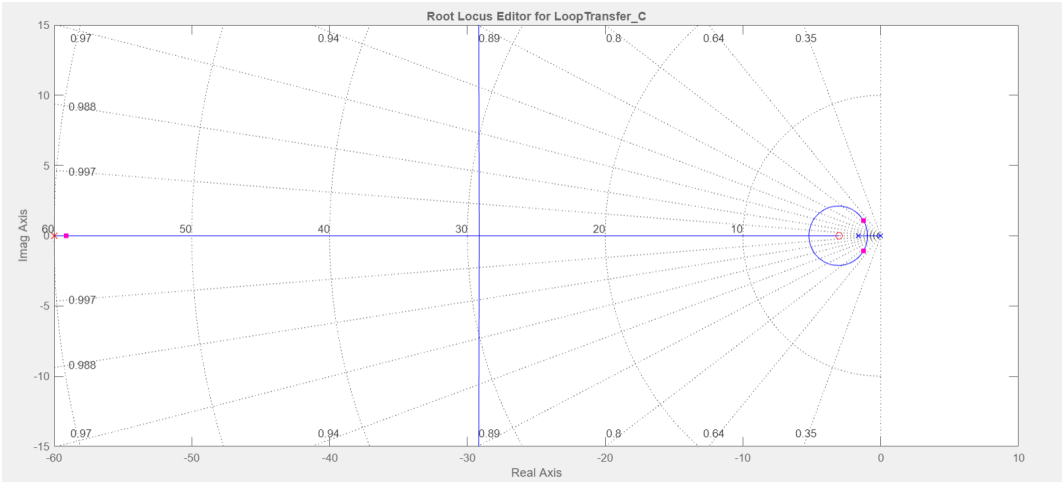


Figura 5.12: Luogo delle radici relativo a $F(s)_{pitch}$

Come si osserva dalla figura 5.12 dopo l'applicazione del controllore progettato sono verificate anche le specifiche lasche. Infatti, tutti i poli in catena chiusa sono posizionati nella regione desiderata a parte reale < -1, in particolare si ha un polo puramente reale nella posizione -59.1 e due poli complessi coniugati rispettivamente nelle posizioni $-1.26-1.08i$ e $-1.26+1.08i$ (in figura rappresentati con dei quadratini fucsia). L'attuale centro degli asintoti è situato nella posizione $s_0 = -29.3095$, in pieno accordo con la condizione 5.16.

Per terminare il processo di sintesi andiamo ora a riportare il grafico relativo alla risposta al gradino del sistema in modo tale da valutarne le prestazione in termini di parametri espressi nel dominio del tempo (Figura 5.13).

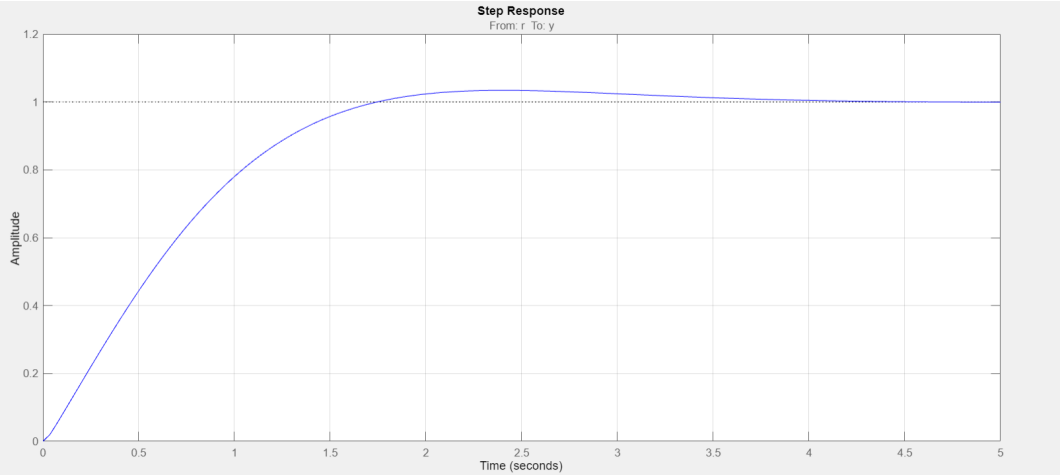


Figura 5.13: Risposta al gradino del sistema $W(s)_{pitch}$

Si osserva un tempo di salita $t_s = 1.17sec$ e una sovraelongazione $\hat{s} = 3.4\%$ che consentono al sistema complessivo di avere ottime prestazioni, anche grazie ad un tempo di assestamento al valore di regime pari a $t_a = 3.16sec$.

L'implementazione di tale controllore all'interno del progetto *asbQuadcopter* di *Simulink* è argomento del capitolo 6.

5.4 Considerazioni sui controllori sviluppati

Tutti i sistemi di controllo precedentemente sviluppati possono a primo impatto apparire inappropriati per il sistema dinamico a cui essi devono essere applicati, ovvero per il controllo della dinamica d'assetto del drone durante il volo. Infatti, osservando i risultati ottenuti tramite il *Control System Designer* di MATLAB si può notare che, per la maggior parte di essi, il tempo di risposta agli stimoli esterni (ad esempio un ingresso a gradino o un disturbo costante agente in catena diretta) risulta essere abbastanza elevato rispetto alle variazioni della dinamica di volo del quadricottero. Questo aspetto risulta essere evidente specialmente per il controllore sviluppato per la regolazione dell'angolo di *roll*, argomento della sezione 5.1, in cui, facendo

riferimento alla figura 5.5, si osserva una risposta molto lenta del controllore rispetto ad un ingresso a gradino unitario, discorso analogo per l'ottenimento dell'astatistmo rispetto a disturbi agenti in catena diretta. In realtà, la scelta di un controllore di questa tipologia è stata necessaria a causa della natura stessa degli attuatori montati a bordo drone *Parrot Mambo*, i quali rispetto a stimoli troppo rapidi si è osservato avere comportamenti indesiderati che portavano ad instabilità o addirittura incidenti durante le simulazioni di volo. Per questo motivo, dopo aver fatto numerose prove con sistemi di controllo più rapidi, ottenendo scarsi risultati, il giusto compromesso si è ottenuto con un controllore più 'lento' che però ha permesso di ottenere ottime prestazioni di volo, sia per riferimenti costanti che variabili.

Lo stesso discorso vale anche per gli altri controllori implementati, in cui si nota una risposta leggermente più 'rapida' ma comunque oggettivamente 'lenta' rispetto alla dinamica del sistema da controllare. Anche in questo caso, dopo numerosi sistemi di controllo testati, i rispettivi regolatori presentati sono frutto del giusto compromesso tra prestazioni di volo e sforzo di controllo richiesto agli attuatori.

Capitolo 6

Implementazione dei controllori sviluppati in ambiente *Simulink*

Nel seguente capitolo verrà eseguita l'implementazione dei regolatori lineari, sviluppati nel capitolo 5, all'interno del progetto *asbQuadcopter* di *Simulink*, analizzato nel capitolo 2 del presente elaborato. Verranno illustrate tutte le modifiche eseguite nel progetto al fine di consentire la corretta visualizzazione dei segnali mediante degli oscilloscopi virtuali e la metodologia utilizzata per la generazione dei comandi di riferimento. Successivamente saranno riportate una serie di simulazioni di volo eseguite sia prima che dopo la sostituzione dei controllori *PID* con i controllori lineari di propria progettazione, con lo scopo di analizzare e confrontare le diverse prestazioni di volo ottenute, sia nel caso di riferimenti costanti (mantenimento della posizione di *Hovering*) che variabili (movimenti durante il volo rispetto tutti i gradi di libertà). Si precisa che, anche avendo sintetizzato i controllori lineari sulla base del modello in spazio di stato del minidrone *Parrot Mambo*, argomentato nella sezione 3.2, le simulazioni di volo sono state comunque eseguite utilizzando il modello non lineare del quadricottero, in modo da valutare l'effettivo funzionamento dei sistemi di controllo sul modello che più si avvicina alla dinamica reale di un minidrone in volo.

6.1 Modifiche eseguite all'interno del progetto *asbQuadcopter*

Le principali modifiche eseguite all'interno del progetto, oltre la sostituzione dei controllori *PID*, sono state fatte per consentire una corretta analisi dei risultati sperimentali ottenuti. In particolare, i blocchi interessati da tali modifiche sono stati sostanzialmente tre: *Command*, *Visualization* e *gravity feedforward/equilibrium thrust*, la cui struttura originale è stata descritta rispettivamente nelle sezioni 2.5, 2.6 e 2.4.5.

Di seguito vengono mostrati brevemente i cambiamenti eseguiti in ognuno di questi.

Modifiche blocco *Command* : facendo riferimento alla struttura originale, riportata in figura 2.32, la sostanziale modifica fatta è stata la sostituzione del blocco responsabile della generazione dei segnali di riferimento: per semplicità di

utilizzo, il blocco originale *Signal Editor* è stato sostituito con un blocco *Signal Builder*, avente un'interfaccia più intuitiva per la costruzione dei riferimenti. La nuova struttura modificata è mostrata nella figura 6.1.

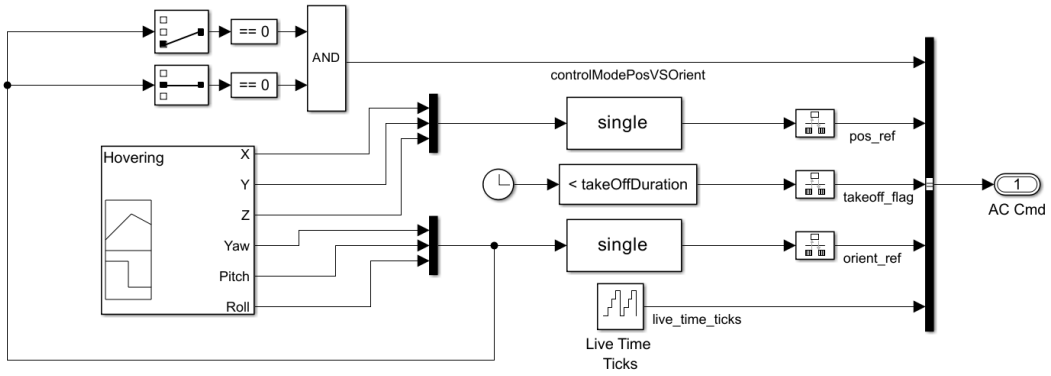


Figura 6.1: Blocco *Command* modificato con *Signal Builder*

Per la corretta interazione con tutto il resto del progetto, è stato necessario inserire dei blocchi *Cast to Single* e *Rate Transition*, oltre che due blocchi *Mux* per raggruppare in soli due vettori (*pos_ref* e *orient_ref*) le componenti relative ai riferimenti di posizione e orientamento nello spazio.

Modifiche blocco *Visualization* : con riferimento alla struttura originale riportata nella figura 2.34, le modifiche eseguite sono state fatte in modo tale da poter visualizzare tramite gli oscilloscopi virtuali solamente i segnali rilevanti per lo scopo dell'elaborato, ovvero quelli in merito ai riferimenti e al valore delle rispettiva variabili di stato e uscita del sistema.

In figura 6.2 è rappresentato il nuovo blocco modificato.

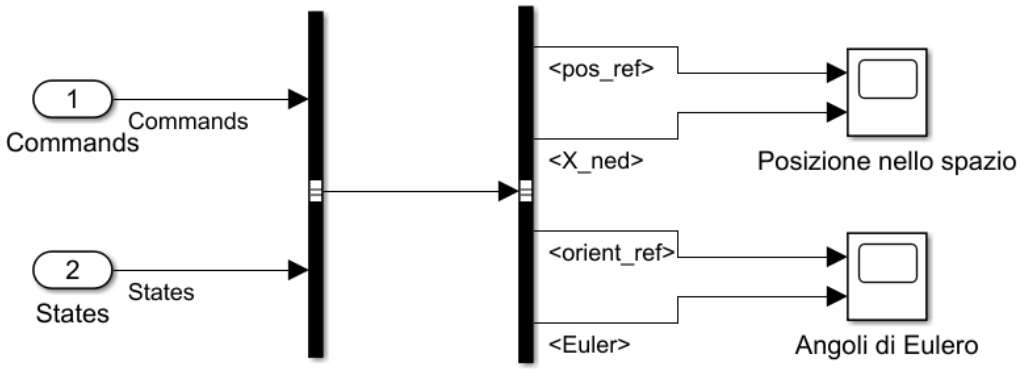


Figura 6.2: Blocco *Visualization* modificato

Dalla prima variabile di ingresso *Commands* sono state selezionate solamente le componenti relative alla posizione di riferimento nello spazio X, Y, Z e all'orientamento di riferimento, ovvero i rispettivi angoli di inclinazione *roll, pitch* e *yaw*.

Invece, dalla seconda variabile di ingresso *States* si sono scelte solamente le componenti relative alla posizione reale nello spazio e agli angoli di Eulero (angoli di rotazione attorno gli assi del *Body Frame*), in modo tale da poter confrontare quest'ultimi con i lori riferimenti. Si precisa che la visualizzazione della posizione misurata (reale) nello spazio, contenuta nella variabile *X_ned*, fa riferimento alle coordinate del drone all'interno dell'ambiente virtuale in cui è inserito, dove di default i rispettivi valori di X, Y, Z sono rispettivamente $57, 95, -0.046$ (coordinate di riferimento iniziali). Di conseguenza nel caso di mantenimento della posizione di *Hovering*, i valori misurati e visualizzati, relativi alla posizione nello spazio, saranno quelli appena riportati.

Si ricorda che, come già indicato nella sezione 2.6, per l'utilizzo della modalità di visualizzazione '*Scopes*' è necessario settare la variabile d'ambiente $VSS\_VISUALIZATION = 0$. In alternativa, con lo scopo di visualizzare contemporaneamente sia i grafici dei segnali che la rappresentazione *3D* del volo tramite *Simulink 3D*, sono stati comunque inseriti dei blocchi *Scope* all'interno dei sotto-sistemi contenenti i controllori, la cui struttura verrà successivamente mostrata.

Modifiche blocco *gravity feedforward/equilibrium thrust* : quest'ultimo blocco, contenente il sistema di controllo per l'altitudine Z (la cui struttura è presente nella figura 2.30), è stato lievemente modificato in modo da migliorare le prestazioni di volo complessive del drone. Infatti, durante le simulazioni avendo notato delle oscillazioni anomale in fase di decollo per tutti i gradi di libertà, oltre che un'elevata sovraelongazione della Z rispetto il suo riferimento, si è trovata soluzione all'interno di tale blocco. In particolare, modificando il valore del guadagno *takeoff_gain* (utilizzato per ottenere un rapido decollo del drone nel primo secondo di volo) di default settato a 0.45 si sono riscontrati dei netti miglioramenti in volo. Il nuovo valore scelto è *takeoff_gain*$= 0.13$ che ha consentito di ottenere un tempo di decollo comunque molto rapido senza però influire troppo sei restanti gradi di libertà e senza ottenere una sovraelongazione troppo elevata della posizione Z rispetto al suo valore di riferimento.

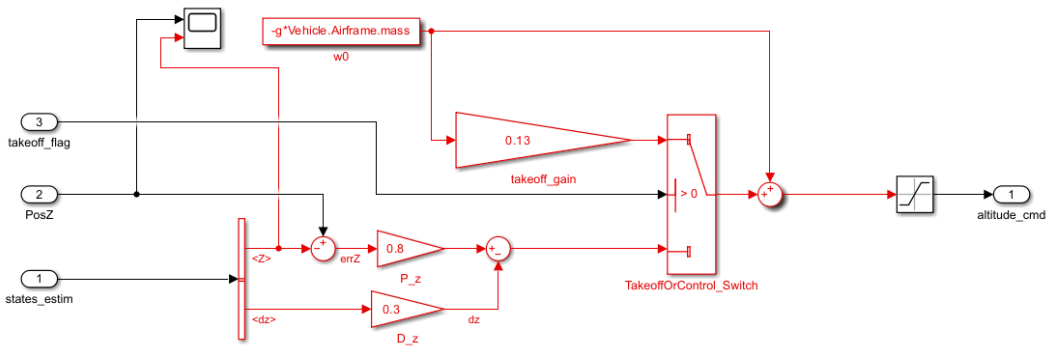


Figura 6.3: Blocco *gravity feedforward/equilibrium thrust* modificato

6.2 Sostituzione dei controllori *PID* con i nuovi sistemi di controllo lineari

Dopo aver illustrato le principali modifiche effettuate all'interno del modello *Simulink* di riferimento, andiamo ora ad illustrare l'implementazione vera e propria dei sistemi di controllo precedentemente sviluppati nel capitolo 5. Prima di mostrare la nuova struttura dei blocchi interessati, poiché i controllori lineari sono stati sviluppati a tempo continuo, cioè nella variabile s, è necessario eseguire una conversione a tempo discreto (nella variabile z) delle relative funzioni di trasferimento, in modo tale da rendere coerente la comunicazione con tutti sotto-sistemi del modello complessivo che per loro natura sono ovviamente componenti discrete. Per ottenere la conversione, si è utilizzata la funzione "*c2d*" del *Control System Toolbox* di MATLAB che a partire dalla funzione di trasferimento a tempo continuo da convertire, dal tempo di campionamento utilizzato e dal metodo di conversione adoperato, consente di ottenere in uscita il risultato desiderato. In particolare, la struttura del comando risulta essere:

$$sys\_d = c2d(sys\_c, Ts, method)$$

in cui:

- sys_d: sistema a tempo discreto risultante
- sys_c: sistema a tempo continuo da convertire
- Ts: tempo di campionamento
- $method$: metodo adoperato per la conversione

Per quanto riguarda il parametro Ts è stato scelto come valore $Ts = 0.005$, in accordo con il tempo di campionamento utilizzato da tutte le componenti del modello *asbQuadcopter*, definito all'interno della variabile d'ambiente Ts caricata nel Workspace di MATLAB in fase di avvio del progetto. Come metodo di conversione si è utilizzato quello di default, ovvero $method =$ '*zoh*' (*Zero-order hold*) in cui si assume che gli ingressi di controllo siano costanti a tratti nel tempo di campionamento Ts, ottenendo in uscita un segnale 'a scalini', con 200 SPS (*Sample-Per-Second*) nel nostro caso.

Facendo riferimento alla struttura a tempo continuo delle funzioni di trasferimento dei controllori per gli angoli di *roll, yaw* e *pitch*, sintetizzati nelle sezioni 5.1, 5.2, 5.3, le corrispettive f.d.t. a tempo discreto calcolate sono:

Struttura controllore angolo di *roll* a tempo discreto

$$G_{roll}(z) = \frac{0.018113(z - 0.9987)(z - 0.9909)}{(z - 1)(z - 0.8521)} \tag{6.1}$$

Struttura controllore angolo di *yaw* a tempo discreto

$$G_{yaw}(z) = \frac{0.2243(z - 0.9989)(z - 0.9884)}{(z - 1)(z - 0.7659)} \tag{6.2}$$

Struttura controllore angolo di *pitch* a tempo discreto

$$G_{pitch}(z) = \frac{0.02(z - 0.987)}{(z - 0.7408)} \tag{6.3}$$

Passiamo ora all'implementazione dei sistemi di controllo lineari in cui verranno utilizzati i valori delle funzioni di trasferimento discrete appena illustrate.

6.2.1 Implementazione dei nuovi controllori per gli angoli di *pitch* e *roll*

Ricordando quanto analizzato nel capitolo 2, i sistemi di controllo per gli angoli di *pitch* e *roll*, grazie alla quale è possibile regolare l'assetto del drone durante il volo, sono contenuti all'interno del blocco *Attitude*, argomento della sotto-sezione 2.4.4. Facendo riferimento alla figura 2.29 contenente la struttura originale del controllore *PID*, l'implementazione del nuovo sistema di controllo lineare è di seguito mostrata.

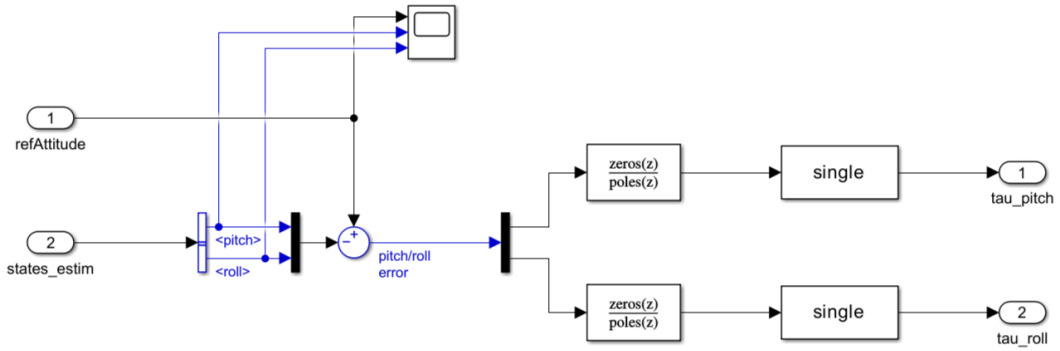


Figura 6.4: Nuovi controllori per gli angoli di *pitch* e *roll* - nuova struttura blocco *Attitude*

Osservando la figura 6.4 si nota come la nuova struttura del blocco *Attitude* risulta essere composta da due blocchi *Discrete Zero-Pole* in sostituzione ai blocchi *Gain* dei precedenti controllori *PID*. Al loro interno sono contenute le relative funzioni di trasferimento dei controllori sintetizzati nelle sezioni 5.1 e 5.3, opportunamente discretizzate. I relativi valori di zeri, poli e guadagno inseriti, sono quelli riportati nelle espressioni 6.3 e 6.1. La presenza del blocco *Casto to Single* è necessaria per mantenere la coerenza tra i tipi di dati in quanto i segnali di uscita *tau_pitch* e *tau_roll* devono essere di tipo *single*, mentre gli ingressi ai blocchi *Discrete Zero-Pole* devono necessariamente essere di tipo *double* (in questo caso il cast dei segnali avviene direttamente all'interno del blocco *sum* settando al valore *double* il parametro "*Output*

data type"). Per quanto riguarda i segnali p e q presenti nella struttura originale dei controllori *PID* rappresentanti rispettivamente le velocità angolari di *roll* e *pitch*, essi sono stati trascurati in quanto non utilizzati da un controllore con questa struttura. Si noti infine la presenza di un blocco *Scope* che consente la visualizzazione dei segnali in merito alla stima degli stati e ai riferimenti, in modo tale da poter eseguire delle analisi anche durante la visualizzazione 3D del volo (si faccia riferimento al paragrafo *Modifiche blocco Visualization* della sezione 6.1 per maggiori informazioni).

Per la visualizzazione dei risultati ottenuti in fase di simulazione con i presenti sistemi di controllo, si faccia riferimento alla sezione 6.3.

6.2.2 Implementazione del nuovo controllore per l'angolo di *yaw*

Il sistema di controllo per l'angolo di *yaw* è contenuto all'interno del blocco *Yaw* del *Flight Control System*, la cui struttura originale di tipo *PD* è stata argomentata nella sotto-sezione 2.4.2. Con riferimento alla struttura originale contenuta in figura 2.27, l'implementazione del nuovo sistema di controllo lineare è di seguito mostrata.

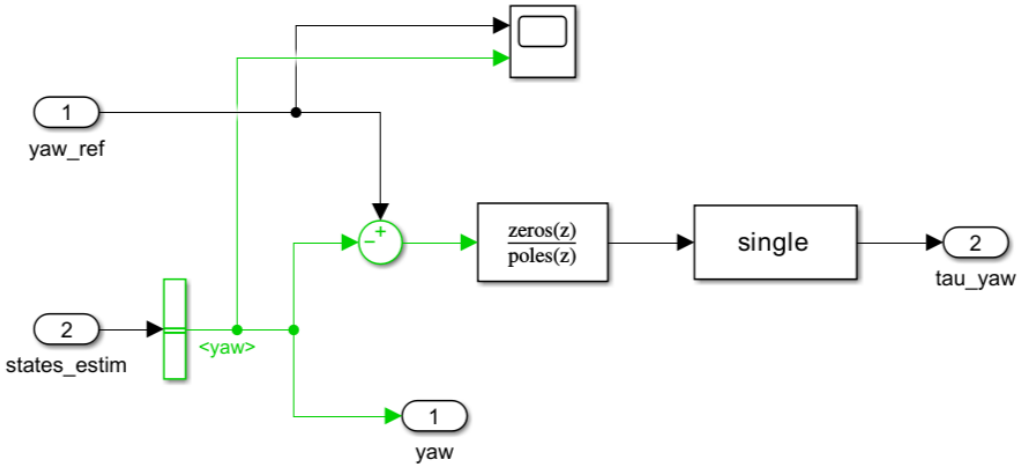


Figura 6.5: Nuovo controllore per l'angolo di *yaw* - nuova struttura blocco *Yaw*

Anche in questo caso la nuova struttura del controllore risulta essere composta da un blocco *Discrete Zero-Pole* contenente la funzione di trasferimento relativa al controllore dell'angolo di *yaw*, i cui rispettivi valori di zeri, poli e guadagno sono quelli riportati nell'equazione 6.2. Rispetto al caso precedente, dove nello stesso blocco erano presenti due sistemi di controllo differenti (uno per il *pitch* e uno per il *roll*), nel blocco in questione troviamo solamente un unico sistema di controllo e di conseguenza non è stato necessario utilizzare un *demultiplexer* per disaccoppiare i relativi segnali di errore in ingresso al controllore. Per quanto riguarda le restanti caratteristiche della struttura riportata in figura 6.5, valgono le stesse considerazioni fatte precedentemente nel caso dei controllori per gli angoli di *pitch* e *roll*.

Per la visualizzazione dei risultati ottenuti in fase di simulazione con il presente sistema di controllo, si faccia riferimento alla sezione 6.3.

6.3 Analisi e confronto delle prestazioni ottenute in fase di simulazione di volo

Dopo aver analizzato e mostrato l'implementazione dei controllori sviluppati, andiamo ora a riportare i risultati di volo ottenuti in fase di simulazione sia utilizzando i controllori *PID* originali che quelli di propria progettazione. Nella prima fase di test si sono utilizzati segnali di riferimento di tipo costante per tutti i gradi di libertà e una volta accertata la stabilità durante il mantenimento della posizione di *hovering*, si è passati ad una seconda fase di test in cui sono stati adoperati segnali di riferimento di tipo variabile (ingressi a rampa con diversi valori di pendenza).

6.3.1 Caso di riferimenti costanti

La definizione dei segnali di riferimento utilizzati è avvenuta all'interno del blocco *Command*, tramite l'utilizzo di un *Signal Builder* introdotto nel primo paragrafo della sezione 6.1. La struttura dei segnali definiti è di seguito riportata.

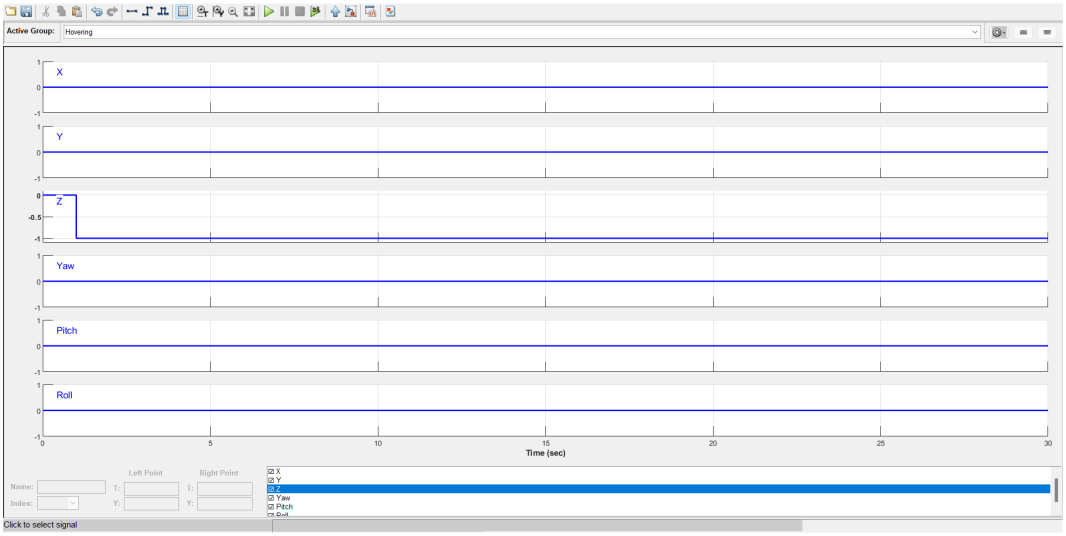


Figura 6.6: Blocco *Signal Builder* nel caso di riferimenti costanti

Poiché la durata per le simulazioni nel caso di riferimenti costanti è stata settata a 30 secondi, i segnali di riferimento sono stati definiti utilizzando un *time range* dello stesso valore, come si può notare dalla figura 6.6. Lo scopo di un test con segnali di questo tipo è stato quello di valutare la 'bontà' dei controllori sviluppati in fase di mantenimento della posizione di *hovering*, confrontando i risultati ottenuti con quelli nel caso di utilizzo dei controllori *PID* originali. Di seguito vengono mostrati gli andamenti di tutti i gradi di libertà rapportati ai loro riferimenti (.

Confronto andamento posizione X

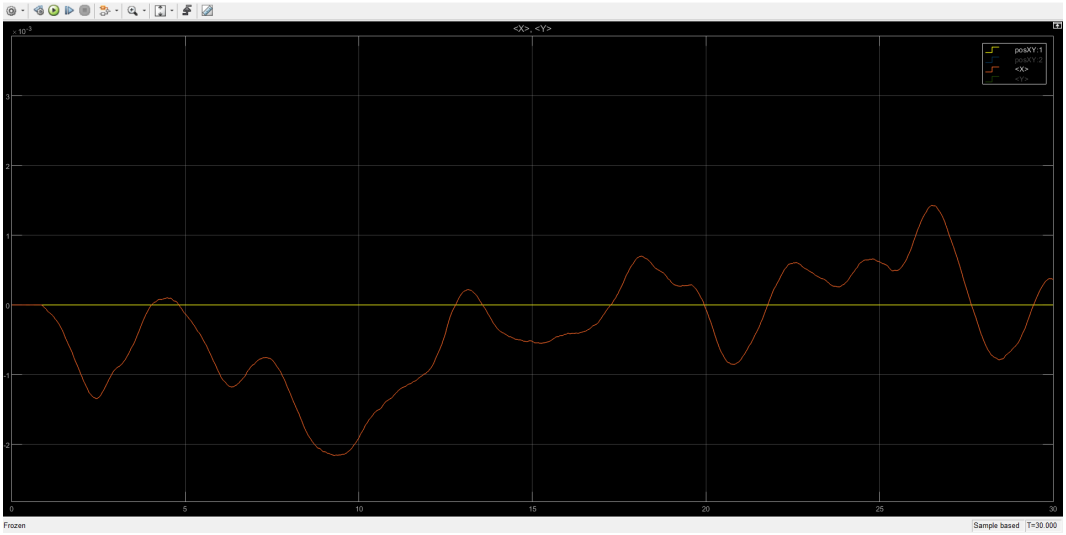


Figura 6.7: Andamento posizione X controllori *PID* originali - riferimenti costanti

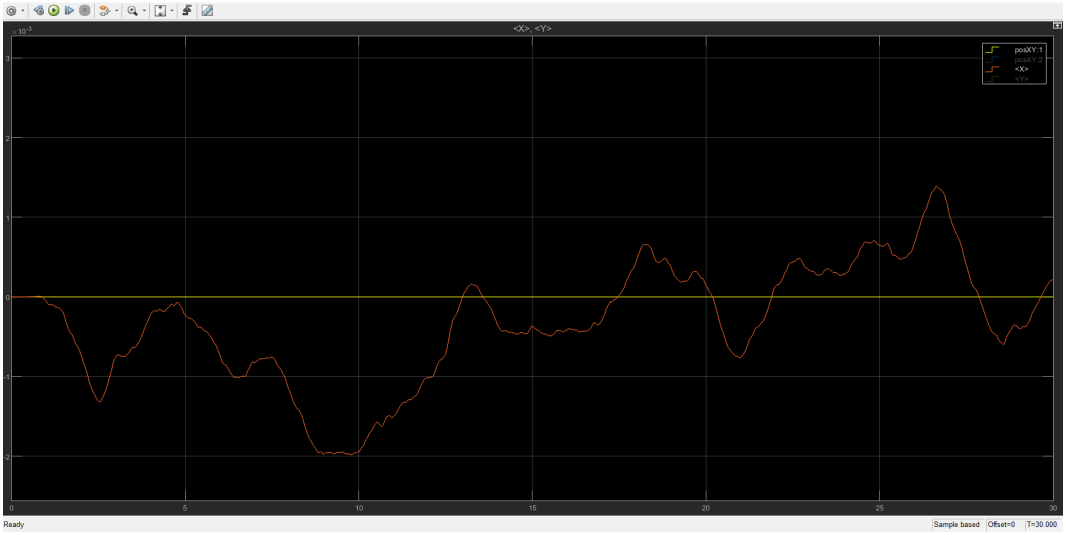


Figura 6.8: Andamento posizione X nuovi controllori lineari - riferimenti costanti

I risultati ottenuti nel caso dell'andamento della posizione X per riferimenti costanti, sono molto simili sia se si utilizza il controllore *PID* originale che quello lineare di propria progettazione. L'errore rispetto al valore di riferimento è in entrambi i casi dell'ordine di 10^{-3}.

Confronto andamento posizione Y

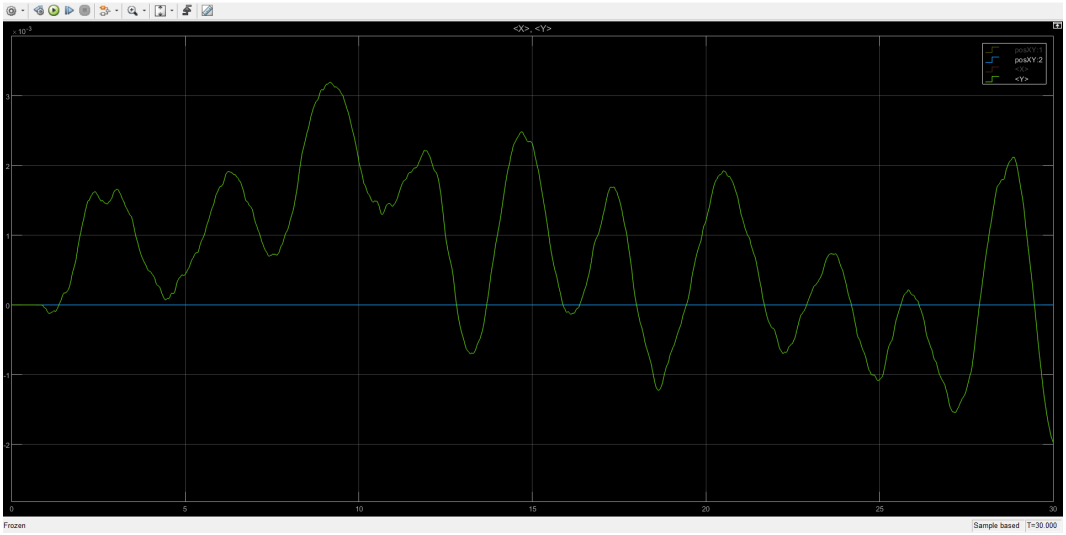


Figura 6.9: Andamento posizione Y controllori *PID* originali - riferimenti costanti

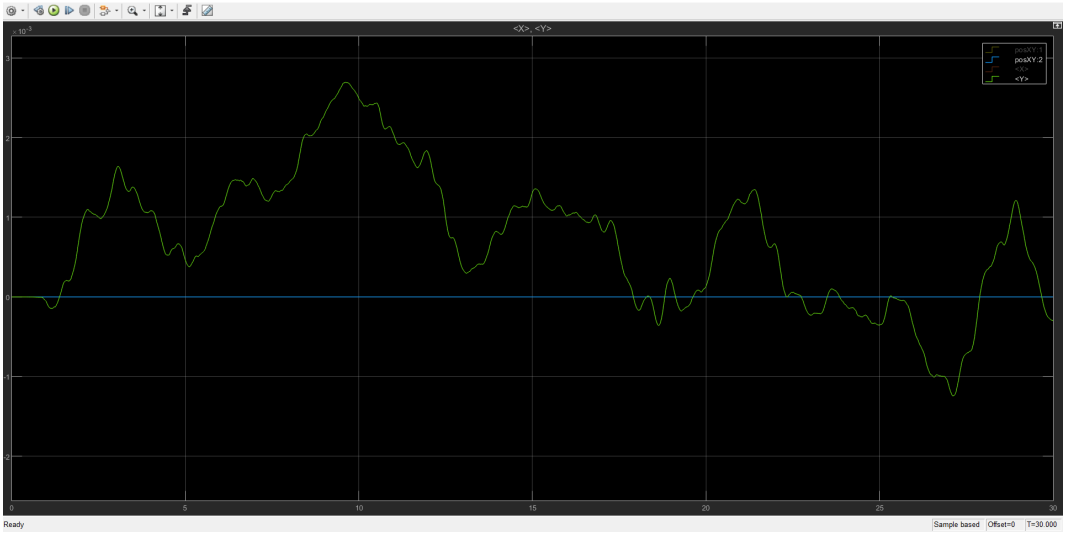


Figura 6.10: Andamento posizione Y nuovi controllori lineari - riferimenti costanti

Anche nel caso della posizione Y i risultati ottenuti sono simili in entrambi i casi. Si noti come i nuovi controllori consentano un'attenuazione delle oscillazioni presenti invece con i controllori *PID* originali. L'errore rispetto al riferimento è sempre dell'ordine di 10^{-3} per entrambi i sistemi di controllo.

Confronto andamento posizione *Z*

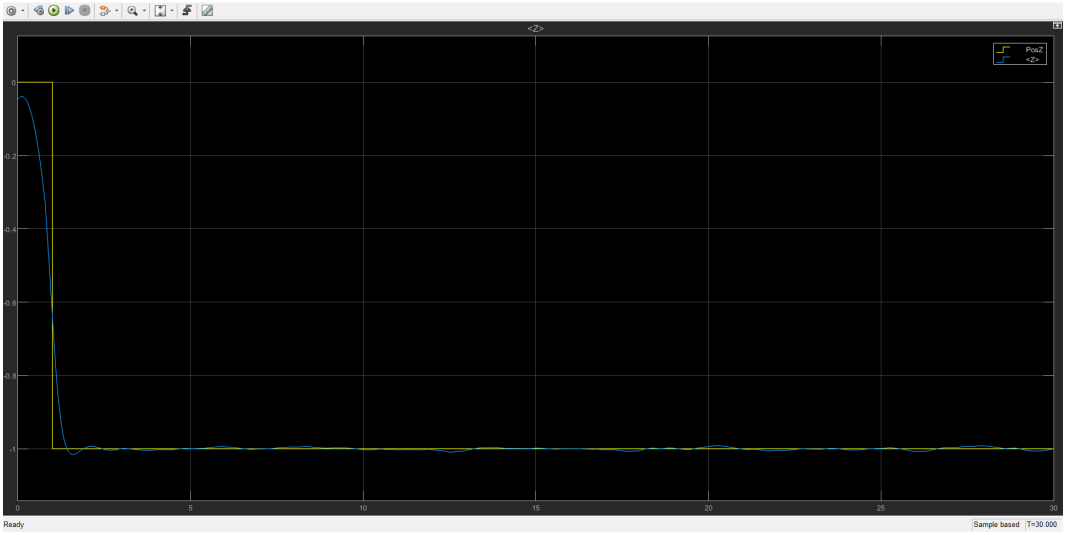


Figura 6.11: Andamento posizione Z controllori *PID* originali - riferimenti costanti

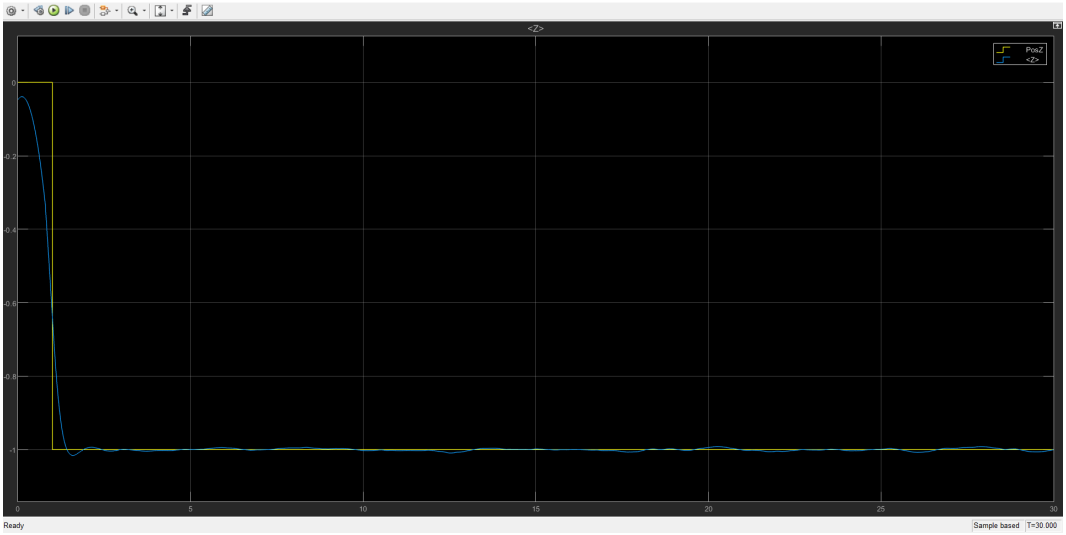


Figura 6.12: Andamento posizione Z nuovi controllori lineari - riferimenti costanti

Nel caso della variabile Z, gli andamenti sono identici sia nel caso di utilizzo dei controllori *PID* che dei regolatori lineari, in quanto il relativo sistema di controllo non è stato modificato e i gradi di libertà interessati dalla sostituzione dei controllori non influiscono sul suo comportamento.

Si precisa che per l'ottenimento di una bassa sovraelongazione in fase di decollo è stata apportata una modifiche al relativo sistema di controllo, come descritto nel terzo paragrafo della sezione 6.1.

Confronto andamento angolo di *roll*

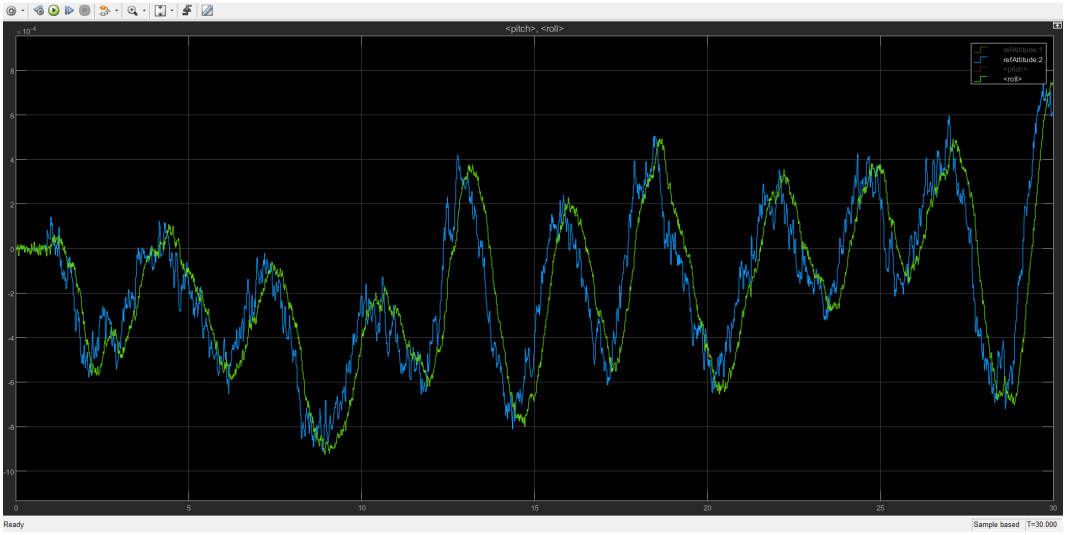


Figura 6.13: Andamento angolo di *roll* controllori *PID* originali - riferimenti costanti

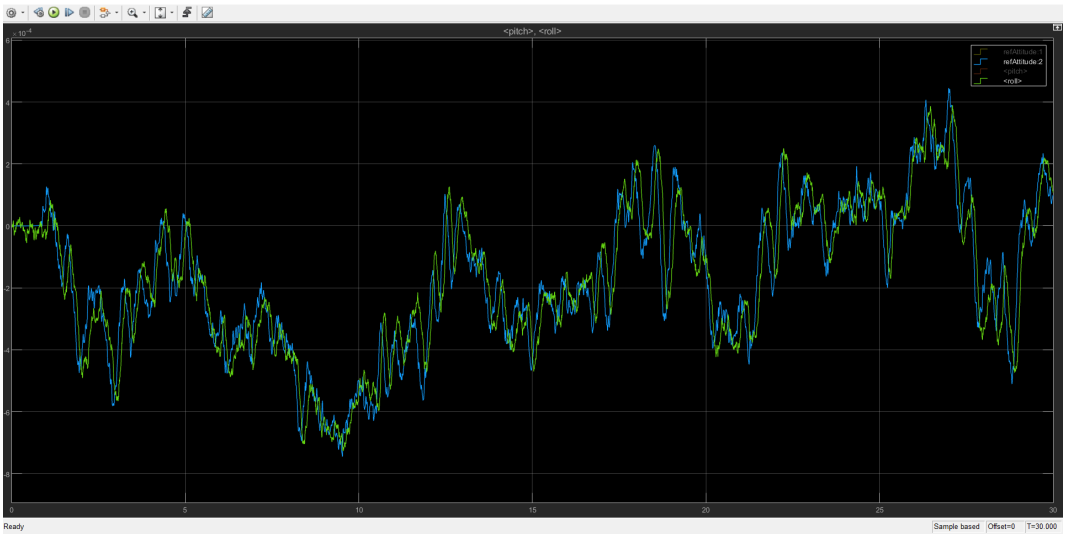


Figura 6.14: Andamento angolo di *roll* nuovi controllori lineari - riferimenti costanti

L'errore ottenuto è in entrambi i casi dell'ordine di 10^{-4}, con una ridotta componente oscillatoria osservabile nel caso di utilizzo dei controllori lineari. In generale, il comportamento è pressoché paragonabile in entrambe le situazioni.

Confronto andamento angolo di *pitch*

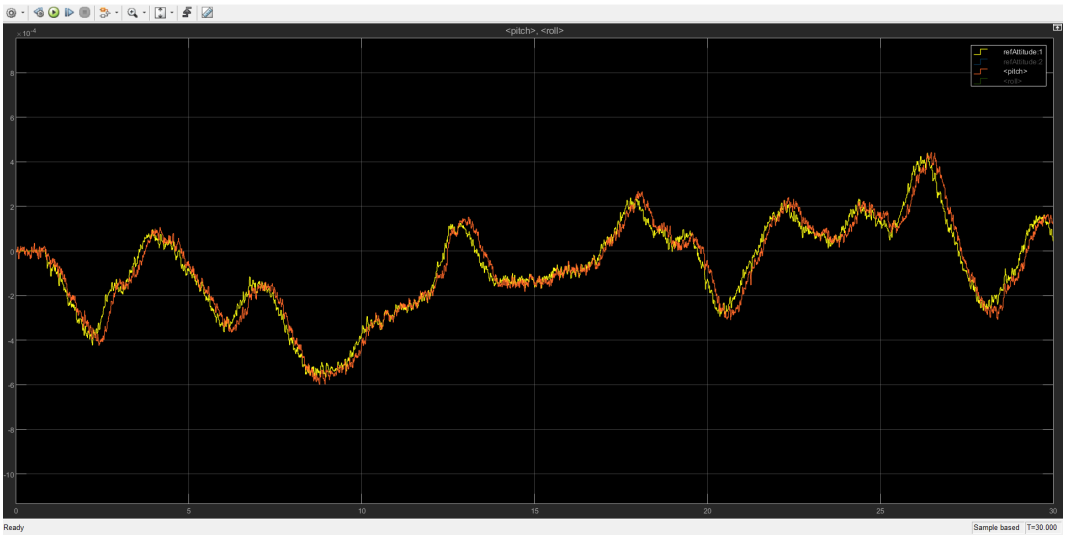


Figura 6.15: Andamento angolo di *pitch* controllori *PID* originali - riferimenti costanti

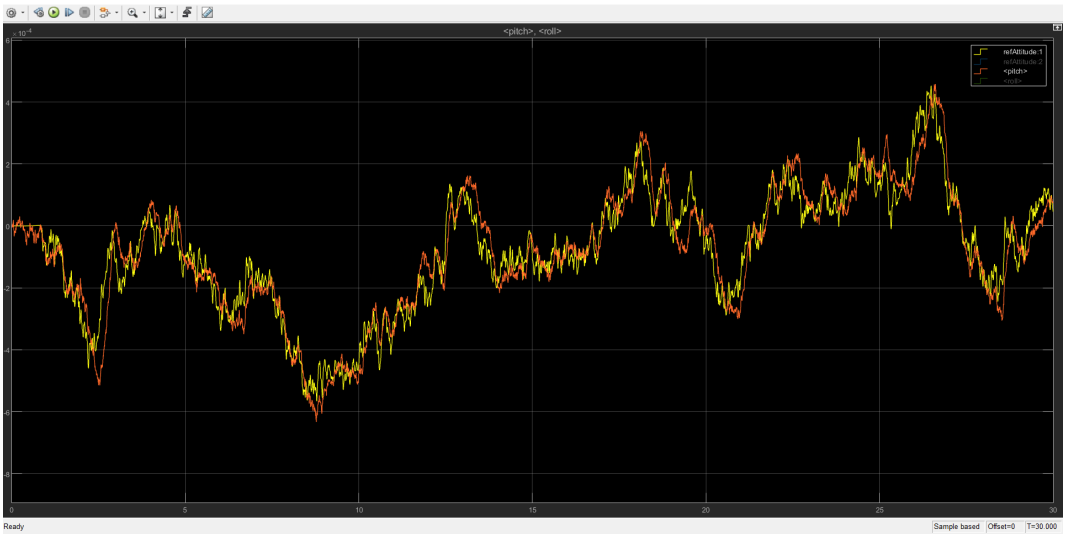


Figura 6.16: Andamento angolo di *pitch* nuovi controllori lineari - riferimenti costanti

Anche l'errore per l'angolo di *pitch* è dell'ordine di 10^{-4}. Con entrambe le configurazioni il risultato ottenuto è considerabile identico.

Confronto andamento angolo di *yaw*

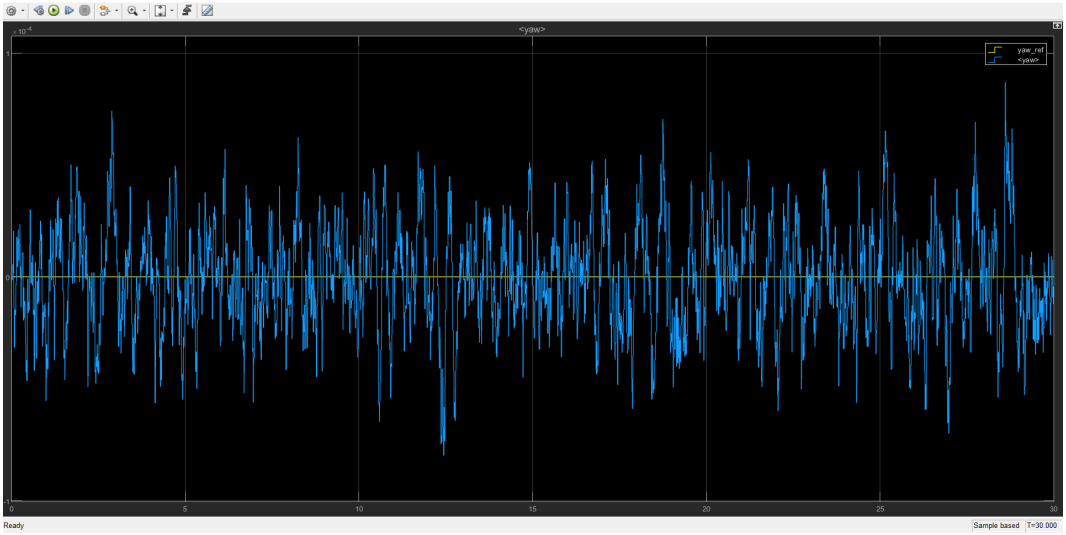


Figura 6.17: Andamento angolo di *yaw* controllori *PID* originali - riferimenti costanti

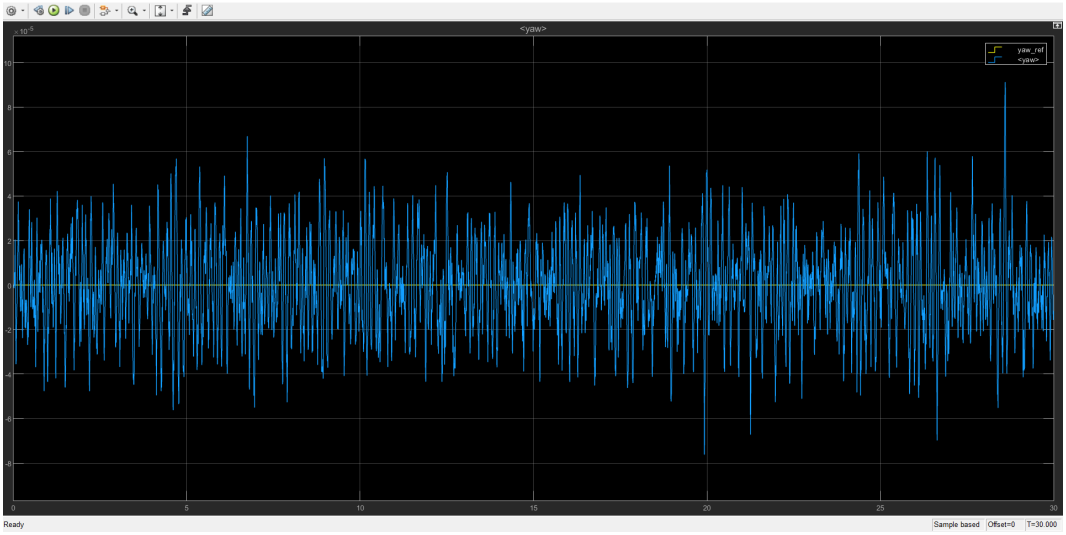


Figura 6.18: Andamento angolo di *yaw* nuovi controllori lineari - riferimenti costanti

Nel caso dell'angolo di *yaw* si sono ottenuti dei lievi miglioramenti rispetto i controllori originali. Infatti, l'errore nel caso di utilizzo dei controllori lineari è dell'ordine di 10^{-5} mentre per i *PID* esso è superiore di circa un ordine di grandezza (10^{-4}).

6.3.2 Caso di riferimenti variabili

Dopo aver analizzato e confrontato le prestazioni di volo ottenute in fase di mantenimento della posizione di *hovering*, passiamo ora ad esaminare i risultati raggiunti durante l'utilizzo di riferimenti variabili che consentano di eseguire spostamenti in volo lungo tutti i gradi di libertà.

La struttura dei segnali definiti è di seguito riportata.

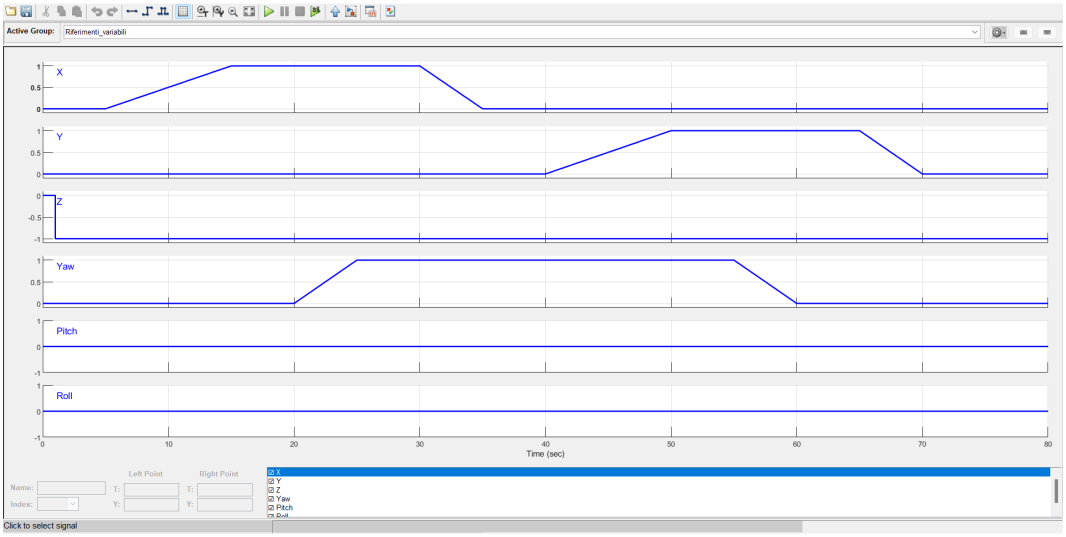


Figura 6.19: Blocco *Signal Builder* nel caso di riferimenti variabili

Poiché la durata per le simulazioni nel caso di riferimenti variabili è stata settata a 80 secondi, i segnali di riferimento sono stati definiti utilizzando un *time range* dello stesso valore, come si può notare dalla figura 6.19.

La scelta di utilizzare pendenze variabili per gli ingressi a rampa è stata fatta con l'obbiettivo di poter valutare la risposta del sistema in diverse situazioni. Si precisa che, oltre ad aver testato il drone con i seguenti segnali di riferimento, sono state eseguite anche altre simulazioni con segnali differenti rispetto tutti i gradi di libertà, ottenendo in tutti i casi delle ottime prestazioni di volo.

Per i segnali presenti nella 6.19, vengono di seguito mostrati gli andamenti di tutti i gradi di libertà rapportati ai loro riferimenti (si veda la legenda per la distinzione tra i vari segnali).

Confronto andamento posizione X

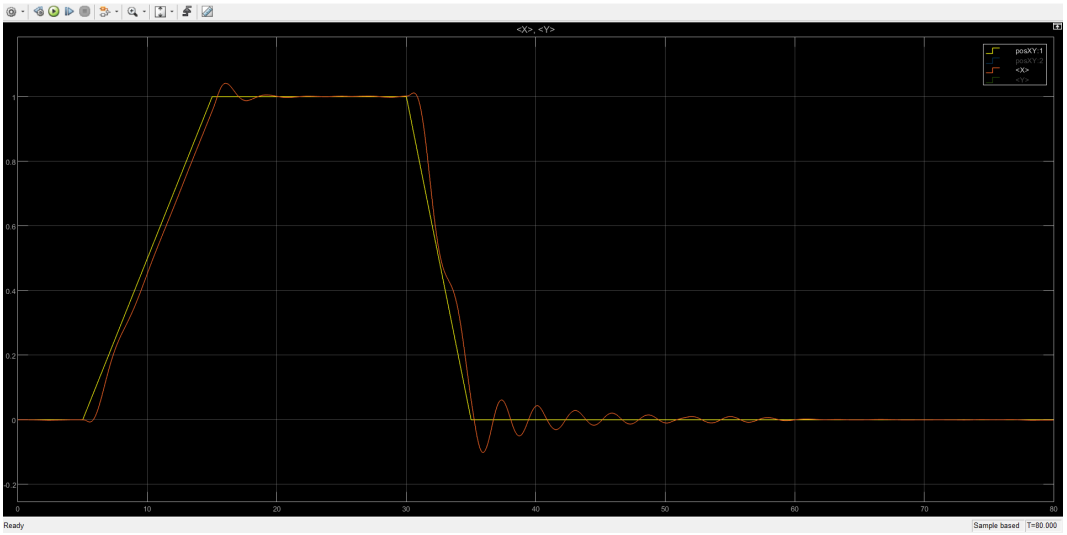


Figura 6.20: Andamento posizione X controllori *PID* originali - riferimenti variabili

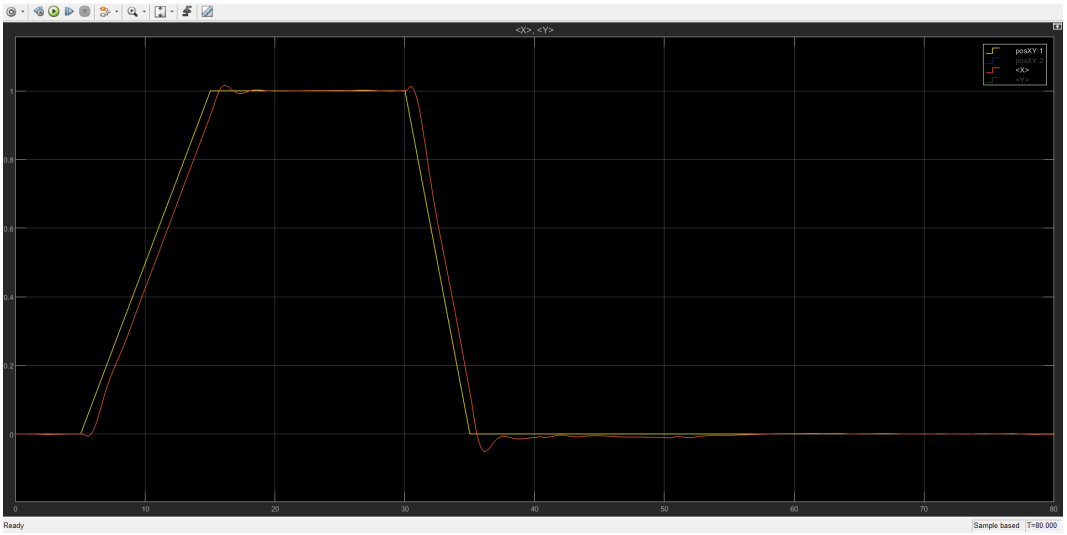


Figura 6.21: Andamento posizione X nuovi controllori lineari - riferimenti variabili

L'utilizzo dei controllori lineari nel caso di riferimenti variabili ha portato considerevoli miglioramenti alla dinamica di volo della posizione X (controllata tramite l'angolo di *pitch*). Si osservi una netta riduzione delle oscillazioni in fase di brusche variazioni di segnale (accelerazioni/frenate improvvise) ed una fedeltà di risposta maggiore rispetto ai *PID* originali.

Confronto andamento posizione Y

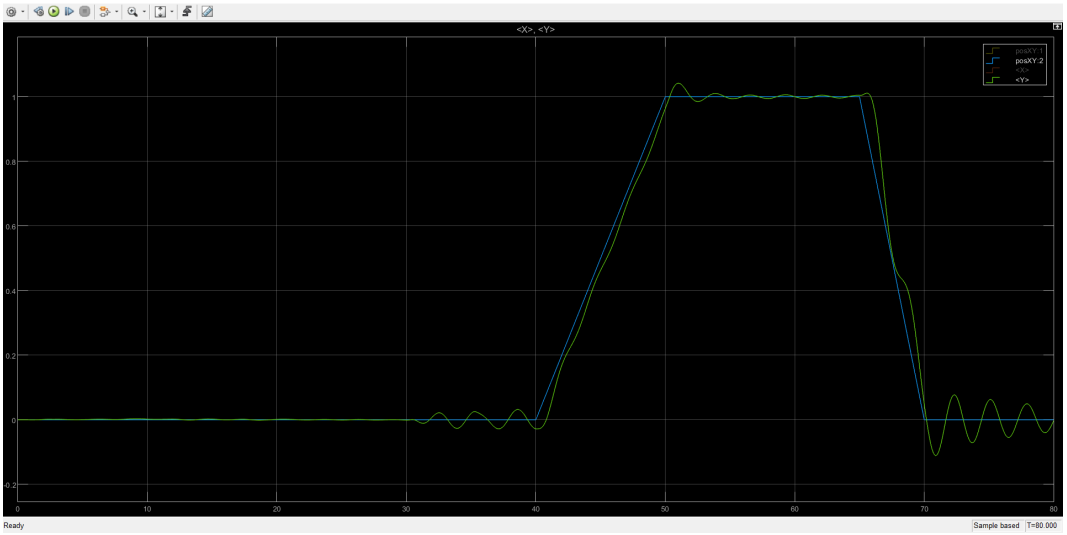


Figura 6.22: Andamento posizione Y controllori *PID* originali - riferimenti variabili

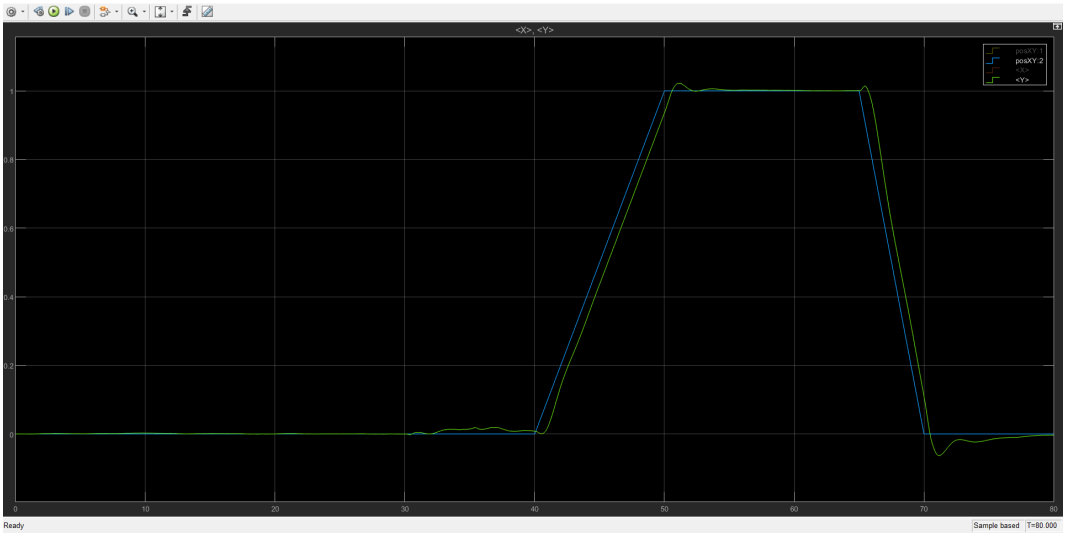


Figura 6.23: Andamento posizione Y nuovi controllori lineari - riferimenti variabili

Anche in questo caso è facile notare i miglioramenti ottenuti sostituendo i controllori originali con quelli di propria progettazione (il controllo della Y avviene mediante l'angolo di *roll*). Le oscillazioni precedentemente presenti sono state del tutto attenuate, ottenendo di conseguenza un aumento della fedeltà di risposta, analogamente alla posizione X.

Confronto andamento posizione *Z*

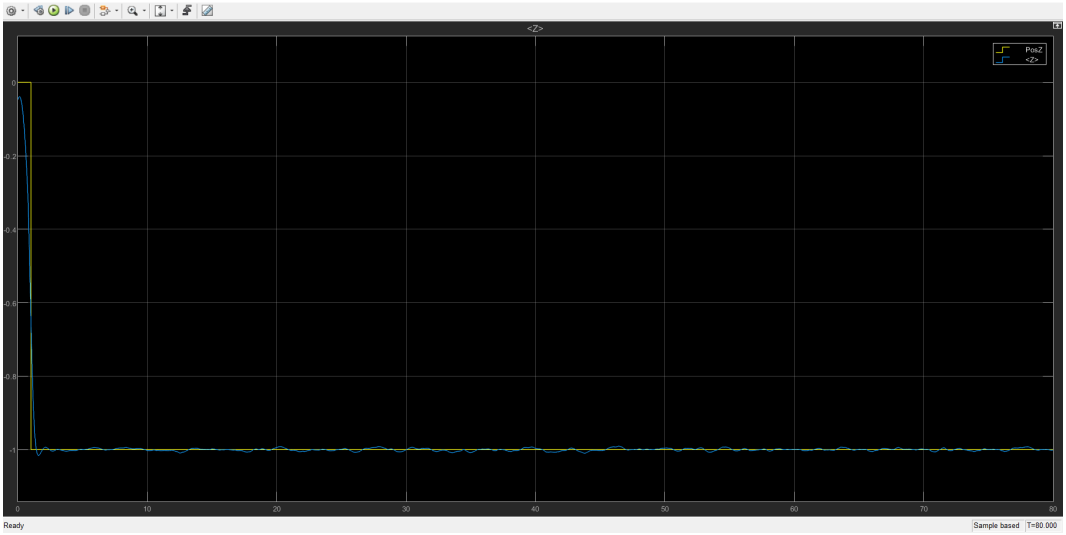


Figura 6.24: Andamento posizione Z controllori *PID* originali - riferimenti variabili

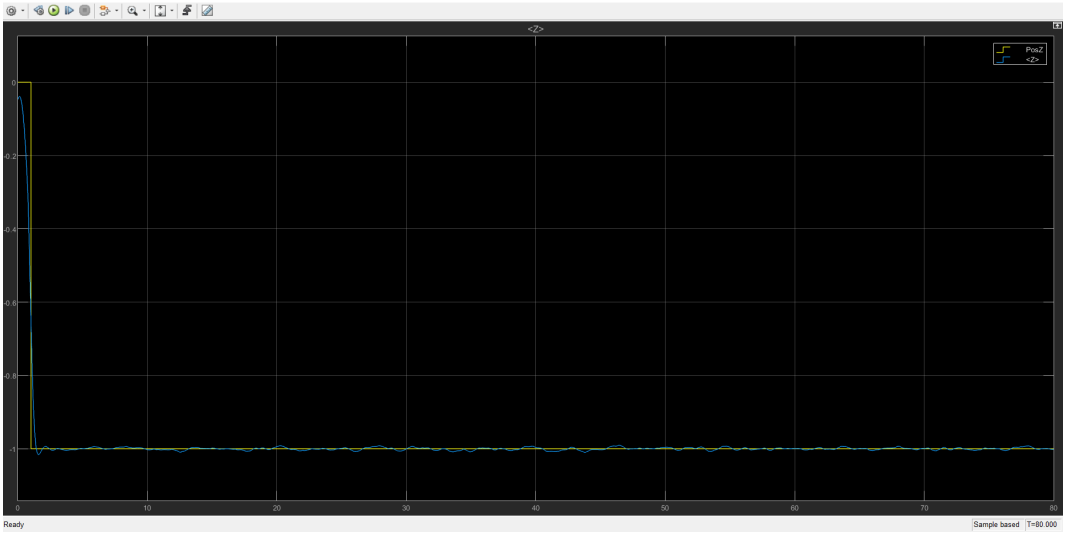


Figura 6.25: Andamento posizione Z nuovi controllori lineari - riferimenti variabili

Nel caso della variabile Z, gli andamenti sono identici sia nel caso di utilizzo dei controllori *PID* che dei regolatori lineari, in quanto il relativo sistema di controllo non è stato modificato e i gradi di libertà interessati dalla sostituzione dei controllori non influiscono sul suo comportamento.

Confronto andamento angolo di *roll*

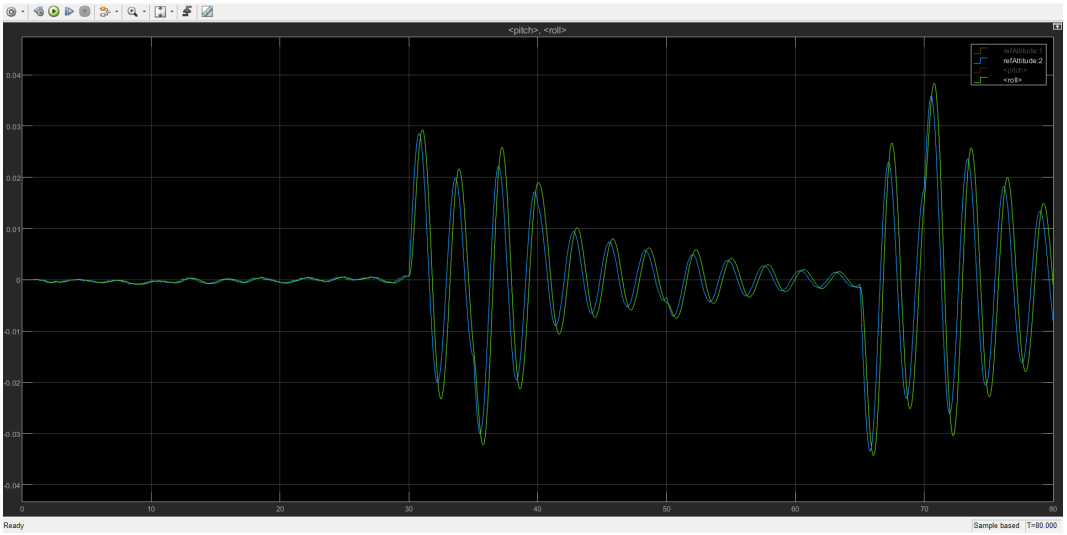


Figura 6.26: Andamento angolo di *roll* controllori *PID* originali - riferimenti variabili

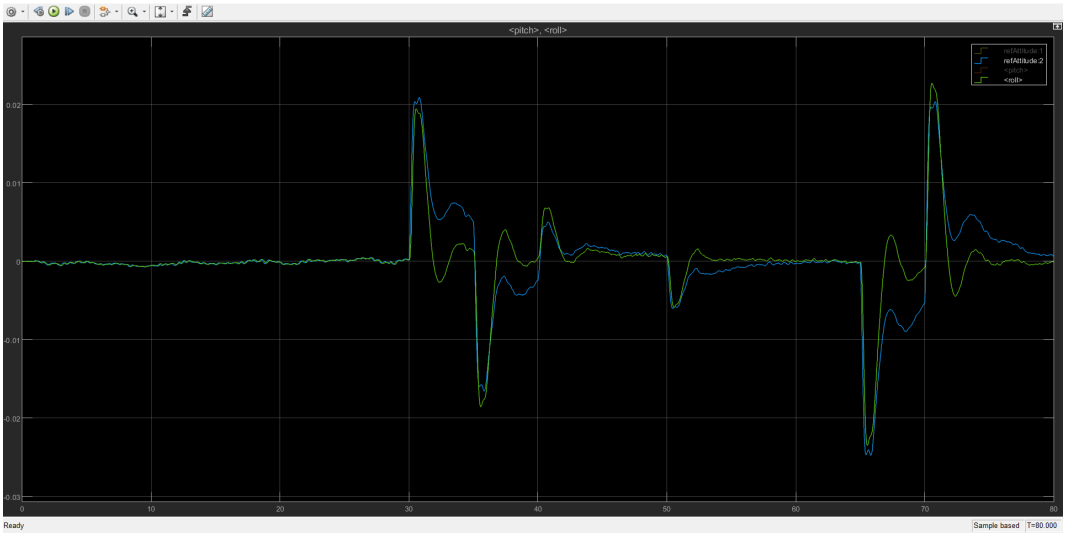


Figura 6.27: Andamento angolo di *roll* nuovi controllori lineari - riferimenti variabili

L'andamento dell'angolo di roll risulta essere migliorato grazie ad una riduzione del contenuto armonico, dovuto ai controllori *PID*. Seppur in alcuni punti l'errore di inseguimento risulti essere maggiore rispetto alla configurazione precedente, i risultati di volo ottenuti sono invece nettamente superiori, come confermato dall'andamento della posizione Y nel caso di riferimenti variabili

Confronto andamento angolo di *pitch*

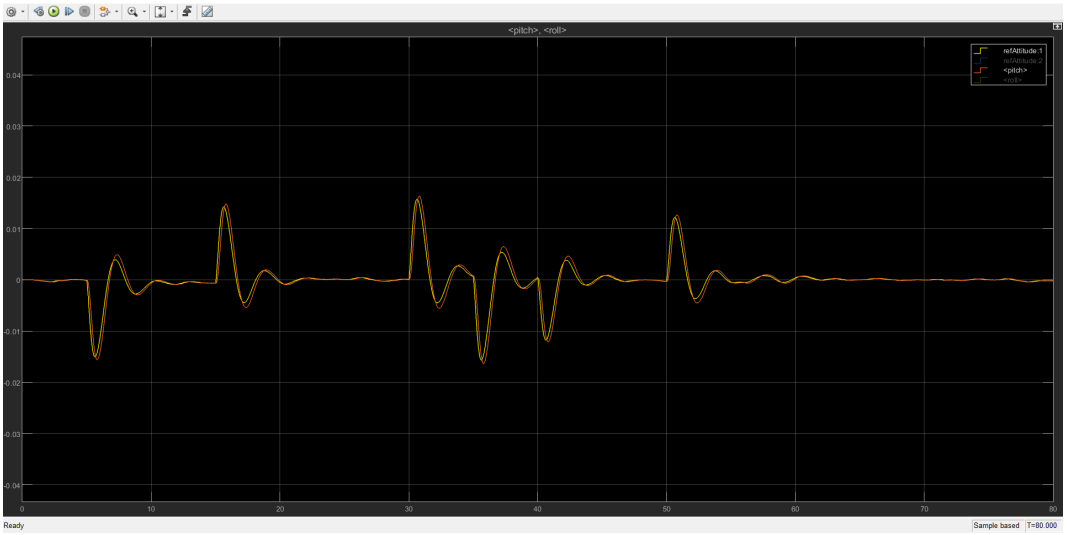


Figura 6.28: Andamento angolo di *pitch* controllori *PID* originali - riferimenti variabili

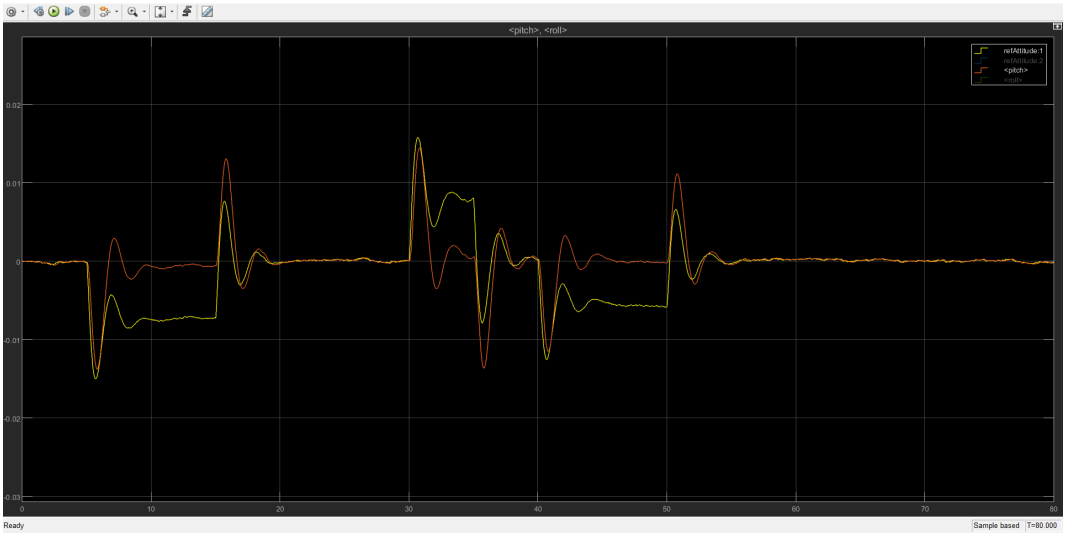


Figura 6.29: Andamento angolo di *pitch* nuovi controllori lineari - riferimenti variabili

Il discorso è analogo al caso dell'angolo di *roll*: l'applicazione dei sistemi di controllo lineare ha apportato notevoli miglioramenti alla dinamica di volo, consentendo una maggior fedeltà di risposta (della posizione X che dipende da tale angolo) anche a fronte di un errore di inseguimento lievemente maggiore rispetto la configurazione con controllori *PID*.

Confronto andamento angolo di *yaw*

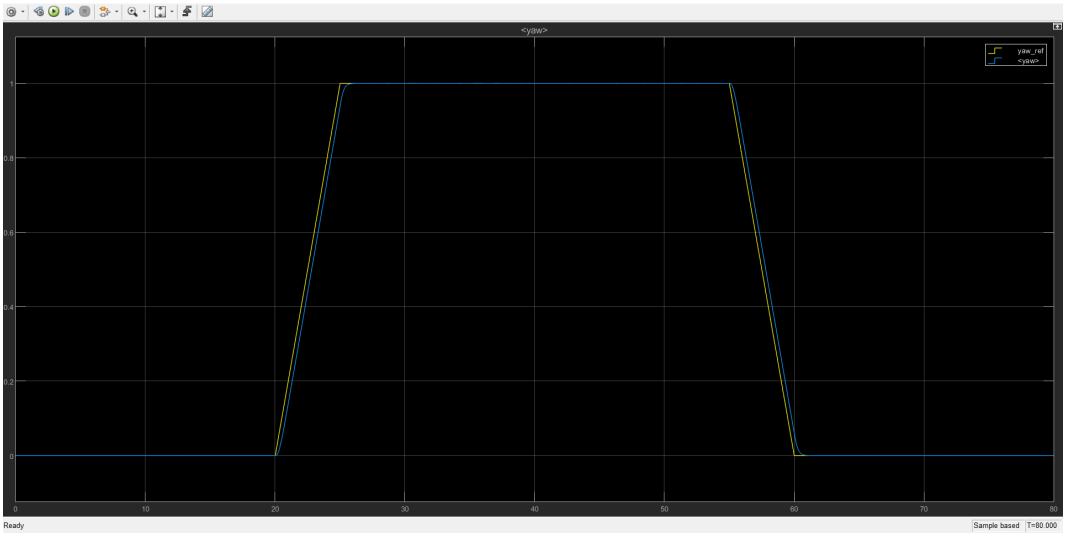


Figura 6.30: Andamento angolo di *yaw* controllori *PID* originali - riferimenti variabili

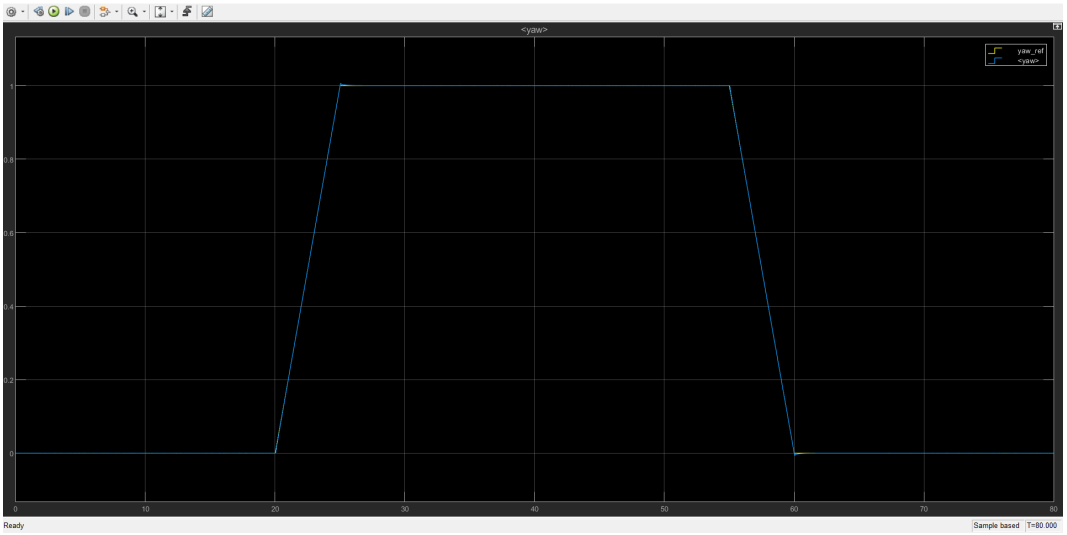


Figura 6.31: Andamento angolo di *yaw* nuovi controllori lineari - riferimenti variabili

Anche per l'angolo di *yaw* si sono ottenuti dei miglioramenti rispetto ai precedenti controllori, già di per sé molto soddisfacenti. La risposta del sistema è molto rapida e precisa tale da consentire al drone di seguire con un' elevatissima precisione i riferimenti impartiti (il segnale di riferimento è quasi coincidente con quello della stima dello stato).

6.4 Considerazioni sulle simulazioni effettuate e i risultati ottenuti

Facendo riferimento ai risultati di volo precedentemente riportati nelle sottosezioni 6.3.1 e 6.3.2, possiamo affermare che, in generale, si sono ottenuti dei netti miglioramenti delle prestazioni sia nel caso di riferimenti costanti che variabili. Se nel mantenimento della posizione di *hovering* le prestazioni sono in entrambi i casi paragonabili, quando si utilizzano segnali di riferimento variabili si ottengono consistenti miglioramenti della dinamica di volo utilizzando i nuovi controllori lineari di propria progettazione. Infatti, rispetto ai *PID* originali, implementando i nuovi sistemi di controllo è stato possibile eliminare le pronunciate oscillazioni dovute a forti variazioni di segnale, ottenendo di conseguenza maggiore stabilità durante generici spostamenti in volo più o meno rapidi e un aumento della fedeltà di risposta complessiva.

Per una migliore visualizzazione delle prestazioni di volo relative ai riferimenti precedentemente riportati nelle figure 6.6 e 6.19, sono state eseguite delle registrazioni video ad alta definizione mediante l'utilizzo del Toolbox *Simulink 3D Animation*, disponibili al seguente *link*.

Capitolo 7

Integrazione dei controllori sviluppati con algoritmi per l'inseguimento di traiettorie

Nel seguente capitolo verranno presentati i risultati di volo ottenuti combinando i controllori lineari, sviluppati nel capitolo 5, con algoritmi di visione e "piloting" tali da consentire al mini-drone *Parrot Mambo* l'inseguimento di generiche traiettorie, con l'obbiettivo di valutare la risposta dei sistemi di controllo implementati a generici segnali di riferimento variabili, impartiti dagli stessi algoritmi (i riferimenti non provengono più dal blocco *Signal Builder*). In questo caso, verrà utilizzato il progetto *parrotMinidroneCompetition*, dedicato nello specifico allo sviluppo di sistemi di controllo per l'inseguimento di traiettorie, opportunamente modificato con gli algoritmi di propria progettazione.

Verranno brevemente descritti i principi di funzionamento degli algoritmi utilizzati per il controllo della traiettoria, per poi passare all'implementazione dei controllori lineari all'interno del nuovo ambiente ed infine mostrare i risultati di volo ottenuti, confrontandoli con quelli dei precedenti sistemi di controllo, che anche in questo caso sono della tipologia *PID*.

Si precisa che il lavoro di progettazione e implementazione degli algoritmi, è stato svolto in collaborazione con un altro progetto di tesi, realizzato da Andrea Caduceo e orientato in maniera specifica alla parte di visione e logica di pilotaggio per l'inseguimento di traiettorie. Per maggiori dettagli in merito, si faccia riferimento a [1].

7.1 Algoritmi di controllo della traiettoria

Andiamo ad analizzare brevemente le tecniche mediante la quale vengono catturate e processate le immagini provenienti dalla telecamera inferiore del *Parrot Mambo*, utilizzate per far seguire al drone la traiettoria definita.

Le fasi principali di cattura ed elaborazione delle immagini (*Image Processing System*) relative alla traiettoria da inseguire sono sostanzialmente tre:

1. Filtraggio dei colori: **filtro HSV**

2. Definizione dei bordi del percorso da seguire: **algoritmo di Canny**

3. Determinazione dell'angolo di inclinazione del drone rispetto al percorso: **trasformata di Hough**

A partire dai risultati ottenuti dall'elaborazione delle immagini sarà possibile determinare la "zona di pilotaggio" a cui corrispondono determinati segnali di controllo. L'insieme di tutte le azioni di controllo relative alle diverse zone di pilotaggio definisce il ***Path Planning***, ovvero la logica di "piloting".

7.1.1 Image Processing System

Come già introdotto, l'*Image Processing System* è quella parte che si occupa dell'elaborazione delle immagini e la determinazione della zona di pilotaggio. All'interno del progetto *parrotMinidroneCompetition* tale sistema è definito all'interno del blocco *Flight Control System*, in cui è presente anche il sotto-blocco *Control System* dove sono implementati i controllori dei vari gradi di libertà e i vari sistemi utilizzati per la stima degli stati.

La struttura del blocco *Image Processing System* in cui sono stati implementati i vari algoritmi di visione sviluppati è di seguito mostrata.

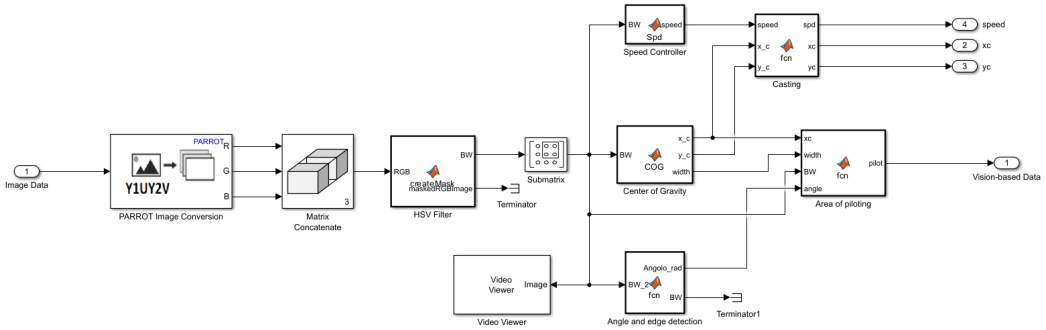


Figura 7.1: Blocco *Image Processing System* - inseguimento di traiettoria

Si precisa che la struttura originale del presente blocco, ottenuta al primo avvio del progetto, non corrisponde con quella appena riportata.

Si osservi come le immagini provenienti dalla fotocamera del drone in formato Y1UY2V vengano convertite in immagini di tipo RGB, ottenute tramite il concatenamento delle relative matrici R, G, e B calcolate dal blocco *PARROT Image Conversion*. Successivamente tramite il blocco *HSV Filter* è possibile passare dallo spazio RGB allo spazio HSV, che consente una migliore analisi delle immagini rispetto alla precedente rappresentazione e permette di isolare il colore del percorso dal resto dell'ambiente. A partire dall'immagine filtrata e opportunamente ridimensionata si vanno a definire i bordi e l'angolo di sfasamento del percorso rispetto la traiettoria del drone, rispettivamente mediante l'algoritmo di Canny e la trasformata di Hough, la cui implementazione è presente all'interno del blocco *Angle and edge detection*. Tramite le informazioni calcolate da tale blocco (angolo θ di sfasamento) è possibile determinare la "zona di pilotaggio" la cui scelta viene effettuata dalla logica presente

nel blocco *Area of piloting*. Esempi di zone di pilotaggio sono la fase di atterraggio, di allineamento al percorso, di decollo, etc.

Per maggiori dettagli sul funzionamento dei blocchi illustrati e di quelli non descritti si faccia riferimento al progetto di tesi [1], svolto in collaborazione con l'autore del presente elaborato, dove è presente un'analisi rigorosa della logica di visione ed elaborazione delle immagini relative alla traiettoria da seguire.

7.1.2 Path Planning

Come anticipato precedentemente, a partire dai risultati ottenuti dall'elaborazione delle immagini provenienti dalla telecamera del drone saranno determinate le relative "zone di pilotaggio". La logica di piloting vera e propria è definita all'interno del *Path Planning*, un particolare blocco contenuto all'interno del *Control System* (Figura 7.2), responsabile della generazione dei segnali di riferimento atti a consentire al drone di seguire la traiettoria disegnata. Tali riferimenti sono contenuti all'interno della struttura dati *UpdateReferenceCmds*, la quale sarà poi data in ingresso al blocco *Controller* dove sono presenti i sistemi di controllo di tutti i gradi di libertà, che consentono di calcolare il relativo sforzo di controllo da inviare ad ogni attuatore del drone, mediante la variabile *motorCmds*. É proprio all'interno di quest'ultimo blocco che successivamente andremo a sostituire i controllori originali con quelli di propria progettazione.

La struttura generale del blocco *Control System* è di seguito riportata:

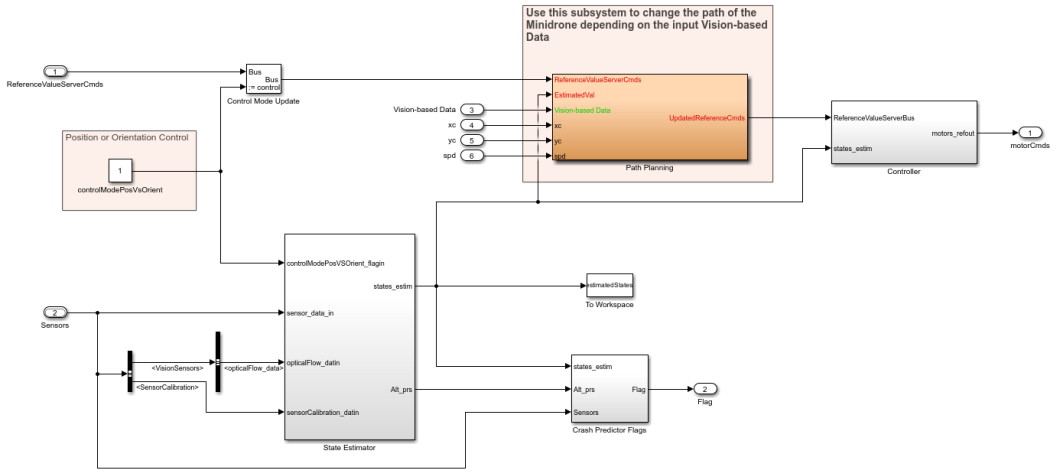


Figura 7.2: Blocco *Control System* - inseguimento di traiettoria

Si noti la presenza di tutti i blocchi appena citati.

In questo caso, soffermiamo la nostra attenzione sulla struttura blocco *Path Planning* in cui sono contenuti gli algoritmi di piloting relativi alle diverse zone di pilotaggio. In particolare, esso prende in ingresso quattro variabili principali, calcolate nella fase di processamento delle immagini:

- *Vision-based Data*: contiene l'attuale "zona di piloting", definita dalla variabile *pilot*
- x_c: coordinata ascisse del baricentro del percorso
- y_c: coordinata ordinate del baricentro del percorso
- *spd*: velocità attuale di volo

Tramite i dati appena citati, l'algoritmo consente di calcolare la giusta posizione di riferimento (*Pos_ref*) e il giusto angolo di orientamento di riferimento (*Orient_ref*), tali da consentire al drone il corretto inseguimento della traiettoria.

Entrando all'interno del blocco *Path Planning* troviamo il blocco in cui è definita la logica vera e propria del calcolo dei nuovi segnali di riferimento per tutti i gradi di libertà interessati (x, y, z, yaw). In particolare, essa è definita all'interno del blocco *Piloting* la cui struttura è di seguito riportata:

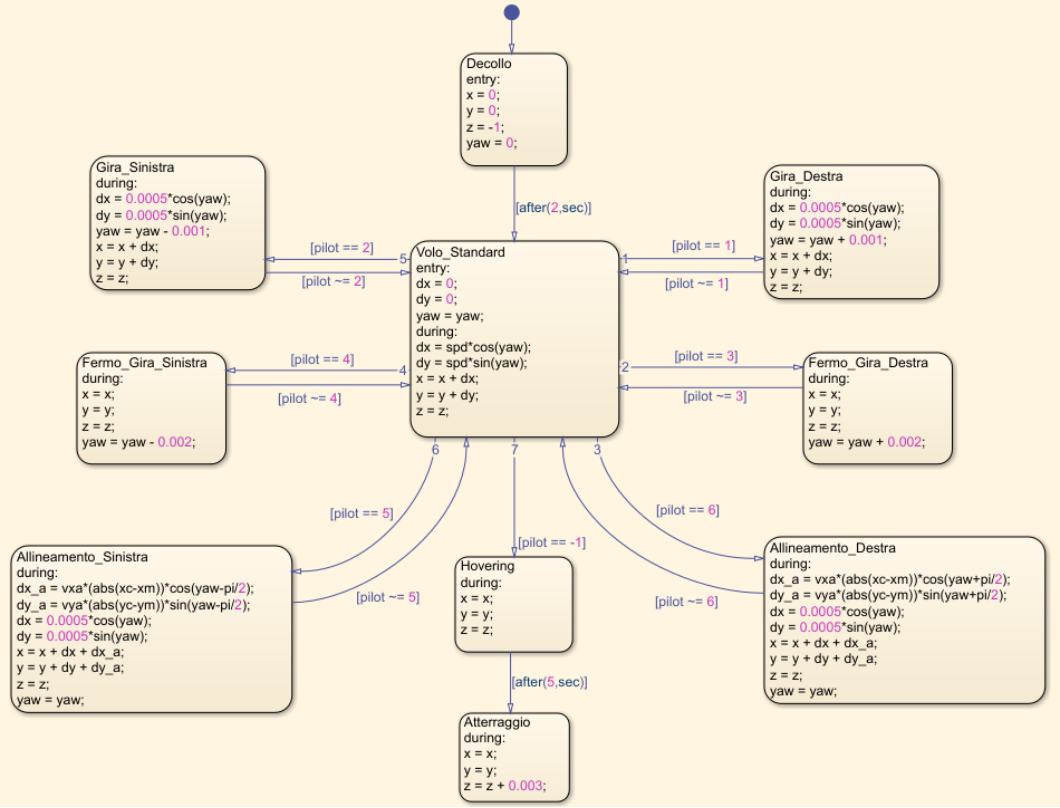


Figura 7.3: Blocco *Path Planning/Piloting* - inseguimento di traiettoria

Dalla figura è facile notare come la logica di pilotaggio è stata gestita ad "eventi" mediante i valori assunti dalla variabile *pilot*. Ciò è stato possibile sfruttando i diagrammi *State Flow* messi a disposizione da MATLAB.

Per maggiori dettagli sull'implementazione degli algoritmi di piloting si faccia riferimento, anche in questo caso, alla tesi [1].

7.2 Sostituzione dei *PID* con i nuovi controllori lineari

Procediamo ora con l'implementazione dei nuovi controllori lineari in sostituzioni ai *PID* originali. Concentriamo quindi la nostra attenzione sul blocco *Controller* (Figura 7.4)in cui sono presenti tutti i sistemi di necessari al controllo di gradi di libertà del drone *Parrot Mambo*. Essendo la dinamica di volo identica in entrambi i progetti utilizzati, la sintesi e l'implementazione di controllori lineari segue la stessa logica utilizzata nei capitoli 5 e 6. Per maggiori dettagli sulla progettazione dei controllori si faccia riferimento alle sezioni 5.1, 5.2 e 5.3, mentre per la fase di implementazione, le strutture dei controllori utilizzate all'interno del progetto *parrotMinidroneCompetition* sono identiche a quelle sviluppate all'interno del progetto *asbQuadcopter*. Per maggiori dettagli si faccia riferimento alle sotto-sezioni 6.2.1 e 6.2.2.

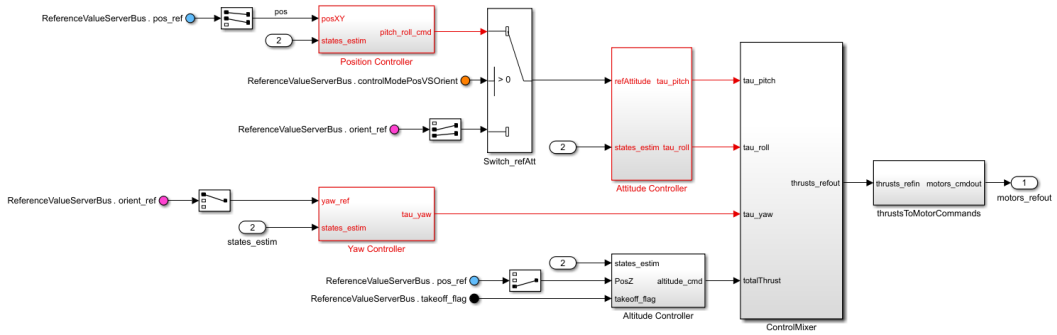


Figura 7.4: Blocco *Controller* - inseguimento di traiettoria

Si ricorda che i gradi di libertà interessati dalla sostituzione dei controllori sono *roll, pitch* e *yaw*, ritenuti di fondamentale importanza in quanto tramite essi è possibile controllare l'intera dinamica di volo del drone (le posizioni x e y vengono regolate attraverso gli angoli di *pitch* e *roll*). Di conseguenza i blocchi interessati dalle modifiche sono *Attitude Controller* e *Yaw Controller*, la cui nuova struttura interna è identica a quella già riportata nelle figure 6.4 e 6.5.

Andiamo ora a mostrare e confrontare i risultati ottenuti durante le simulazioni di volo relativi ad un generico percorso avente la seguente forma:

Figura 7.5: Esempio percorso - T7

Confronto andamento posizione *X*

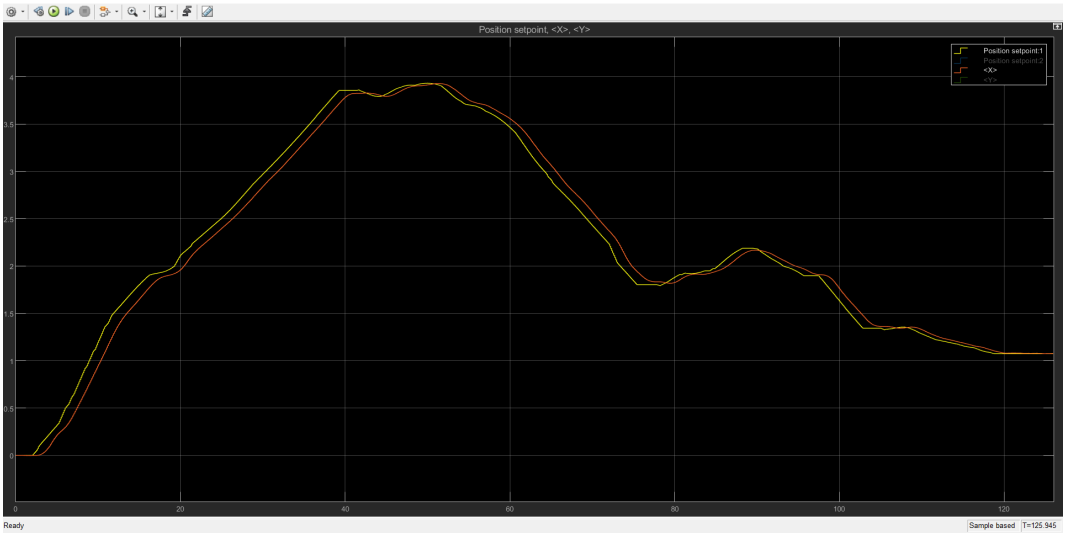


Figura 7.6: Andamento *PID* posizione X - inseguimento di traiettoria

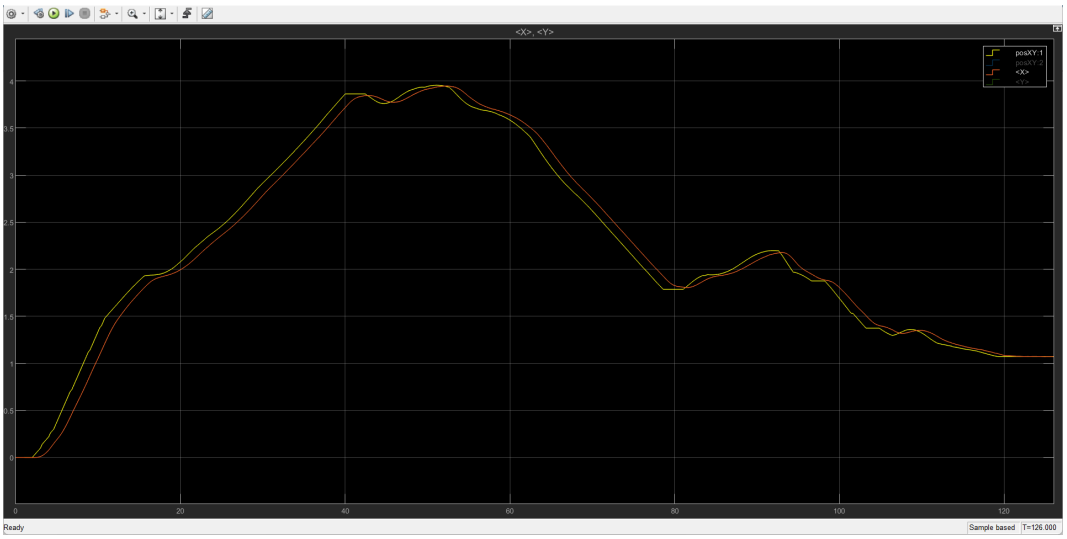


Figura 7.7: Andamento nuovi controllori posizione X - inseguimento di traiettoria

Nel seguente percorso scelto, per l'andamento della posizione X, controllata tramite l'angolo di *pitch*, non si notano grandi differenze tra i due sistemi di controllo utilizzati.

In altre situazioni testate, l'utilizzo dei nuovi controllori ha invece consentito di ottenere dei miglioramenti nelle prestazioni.

Confronto andamento posizione *Y*

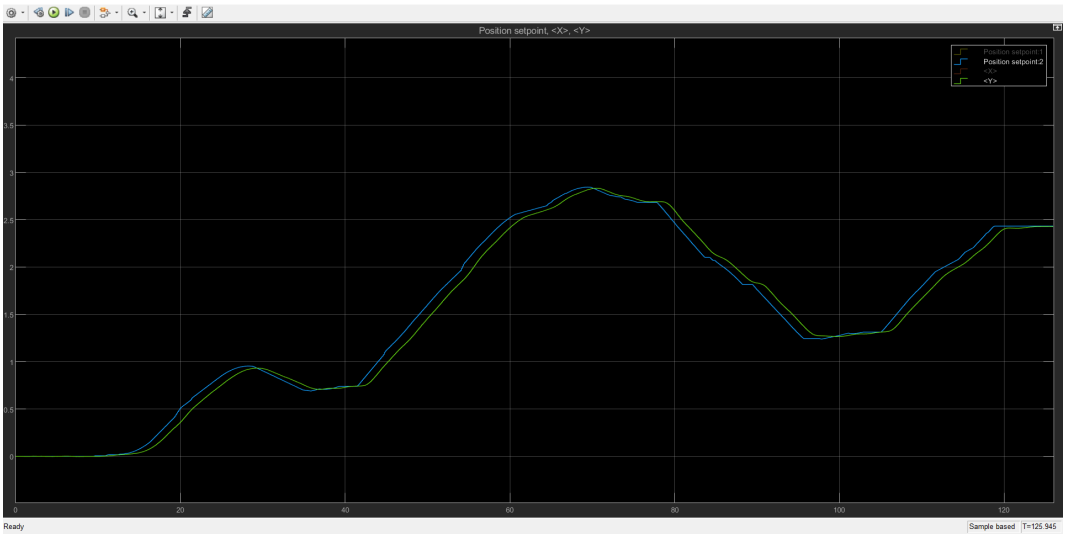


Figura 7.8: Andamento *PID* posizione Y - inseguimento di traiettoria

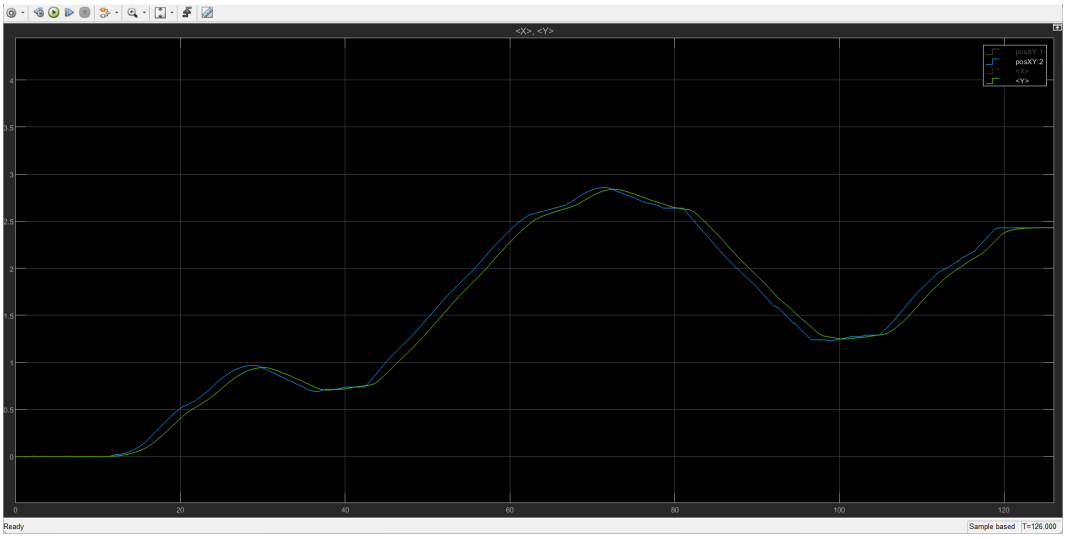


Figura 7.9: Andamento nuovi controllori posizione Y - inseguimento di traiettoria

Anche per il seguente caso, valgono le stesse considerazioni fatte per la posizione X.

Confronto andamento angolo di *yaw*

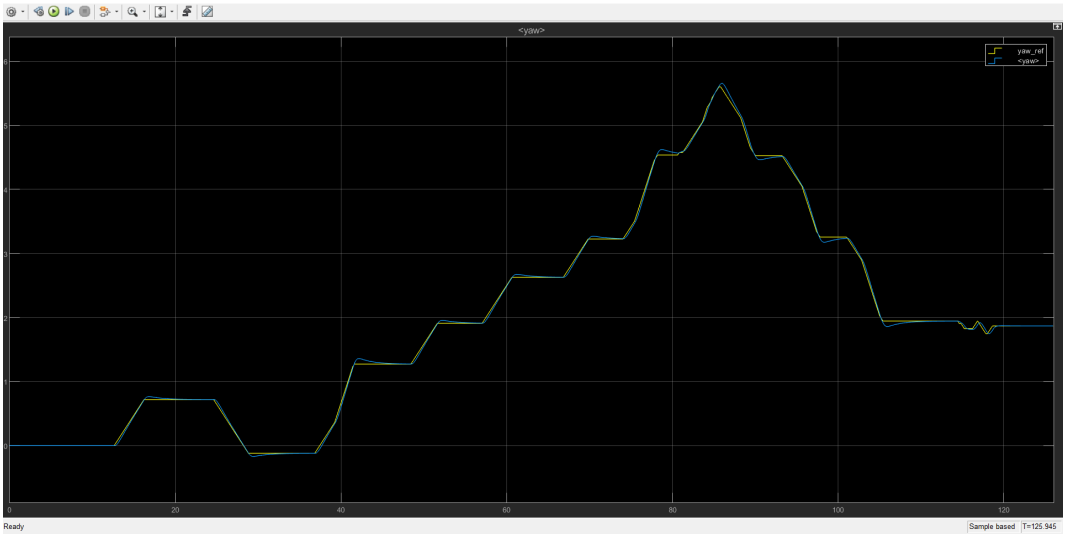


Figura 7.10: Andamento *PID* angolo di *yaw* - inseguimento di traiettoria

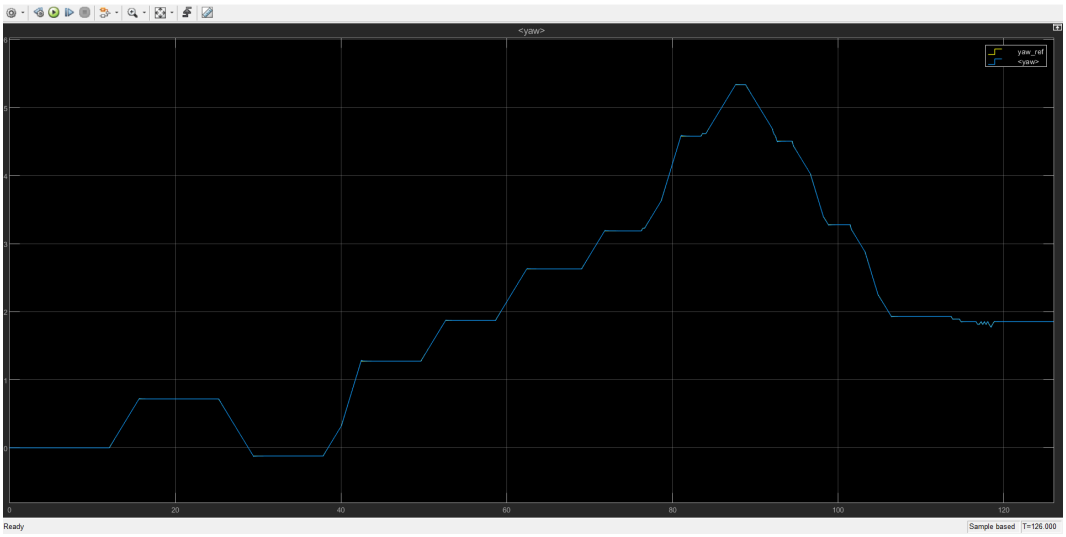


Figura 7.11: Andamento nuovi controllori angolo di *yaw* - inseguimento di traiettoria

Per l'angolo di *yaw* si sono invece ottenuti risultati migliori rispetto all'utilizzo dei controllori *PID*. Come si può infatti notare, i nuovi controllori lineari risultano essere più "pronti" a rispondere alle variazioni di segnale e di conseguenza consentono di ottenere una fedeltà di risposta maggiore. Ciò si evince facilmente osservando i due andamenti e notando che nel secondo caso il segnale di riferimento è praticamente coincidente con quello della stima dello stato.

Si sono riportati solamente gli andamenti relativi alle posizioni X e Y e all'angolo di *yaw*, in quanto sono i gradi di libertà interessati dalle variazioni di segnale, tramite i quali è possibile analizzare la risposta complessiva del sistema.

I risultati ottenuti nelle numerose simulazioni eseguite su differenti percorsi, hanno consentito di registrare in alcuni casi dei netti miglioramenti delle prestazioni rispetto i precedenti controllori *PID*.

Possiamo quindi concludere affermando che i sistemi di controllo lineari sviluppati sono considerabili di valida applicazione anche in situazioni più critiche, come nel caso di inseguimento di traiettorie generiche.

Per una migliore visualizzazione di alcune delle simulazioni eseguite, anche in questo caso sono disponibili al seguente link delle registrazioni video ad alta qualità in cui è possibile constatare la qualità dei controllori e degli algoritmi di inseguimento sviluppati. Se si desidera visualizzare le prestazioni e le problematiche riscontrate invece con i controllori *PID* originali e con una vecchia versione degli algoritmi implementati, si faccia riferimento al seguente link.

Conclusioni

L'obbiettivo preposto del seguente elaborato era quello di progettare dei sistemi di controllo di tipo lineare per tutti i gradi di libertà rotazionali di un quadricottero che, nel caso analizzato relativo al minidrone *Parrot Mambo*, migliorassero le prestazioni di volo rispetto agli originali controllori di tipo *PID*. Con riferimento ai risultati ottenuti durante le fasi di testing, si può affermare che le finalità imposte in partenza sono state ampiamente raggiunte. Con l'implementazione dei nuovi sistemi di controllo lineari si sono infatti riscontrati dei netti miglioramenti nella dinamica di volo (di natura non lineare), specie in presenza di segnali di riferimento variabili, ottenendo nel complesso un sistema più "pronto" alle brusche variazioni di segnale e "fedele" rispetto ai riferimenti impartiti. Avendo inoltre testato i controllori anche in circostanze diverse dall'ambiente in cui essi sono stati sviluppati, ovvero nel controllo della dinamica di volo per l'inseguimento di traiettorie generiche mediante algoritmi di visione, si può garantire con certezza la loro validità e corretta integrazione con i restanti sotto-sistemi del modello complessivo.

Come possibili sviluppi futuri, avendo svolto il lavoro esclusivamente in simulazione, si potrebbe passare all'implementazione dei sistemi di controllo direttamente all'interno del *Flight Control System* del drone reale, in modo da valutarne le prestazioni sperimentali rispetto quelle ottenute in simulazione, specie nel caso di integrazione con algoritmi di visione per l'inseguimento di traiettorie.

Bibliografia

[1] Andrea Caduceo. Studio e sviluppo di algoritmi di visione per l'inseguimento di traiettorie per droni. Tesi triennale, UNIVPM - Università Politecnica delle Marche, 2021 - 2022.

[2] Daniele Gambini. Studio e sviluppo in simulink di tecniche di controllo per minidroni. Tesi triennale, UNIVPM - Università Politecnica delle Marche, 2019 - 2020.

[3] Francesco Paoli Leonardi. Studio e sviluppo di tecniche di controllo avanzate per droni. Tesi triennale, UNIVPM - Università Politecnica delle Marche, 2020 - 2021.

[4] Manuel Valerio Antoni. Controllo d'assetto e di quota per un drone a quattro rotori. Tesi triennale, UNIPD - Università degli studi di Padova, 2021 - 2022.

[5] Toppr. *Drag Force Formula.* URL: `https://www.toppr.com/guides/physics-formulas/drag-force-formula/#:~:text=Therefore%20a%20drag%20force%20is,the%20body%20and%20the%20fluid.`

[6] MathWorks. *Fly a Parrot Minidrone Using the Quadcopter Simulink Model.* URL: `https://it.mathworks.com/help/supportpkg/parrot/ug/fly-parrot-minidrone-quadcopter-simulink-model.html`.

[7] MathWorks. *Flight Simulation Simulink Template for Parrot Minidrone.* URL: `https://it.mathworks.com/help/supportpkg/parrot/ug/flight-simulation-simulink-template-for-parrot-minidrone.html`.

[8] MathWorks. *Quadcopter Project.* URL: `https://it.mathworks.com/help/aeroblks/quadcopter-project.html`.

[9] YouMath. *Sviluppo in serie di Taylor.* URL: `https://www.youmath.it/lezioni/analisi-matematica/derivate/537-come-sviluppare-una-funzione-in-serie-di-taylor.html`.

[10] L. Lanari. *Sintesi in Frequenza: sintesi per tentativi.* Università di Roma La Sapienza, Dicembre 2012. URL: `http://www.diag.uniroma1.it/lanari/FdA9/FdA9MatDid/SintesiTentativi.pdf`.

[11] Alessandro Pisano. *Controlli automatici - Regolatori PID - prima parte*. UNICA - Università degli studi di Cagliari. URL: `https://www.unica.it/unica/protected/247235/0/def/ref/MAT232108/`.

[12] Alberto Isidori. *Sistemi di controllo*, volume Second. Siderea, second edition, 1998.

[13] Nicola Schiavoni Paolo Bolzern, Riccardo Scattolini. *Fondamenti di controlli automatici*. McGraw-Hill, fourth edition, february 2015.