ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Dipartimento di Ingegneria industriale

Corso di Laurea in
INGEGNERIA AEROSPAZIALE

Studio di un algoritmo di Path-Following per un drone subacqueo

Tesi di Laurea in
ING-INF/04: CONTROLLI AUTOMATICI

Relatore
Prof. Matteo Zanzi

Presentata da
Giorgio Aversente

Sessione Luglio 2026

Anno Accademico 2025/2026

Indice

Elenco delle figure

Acronimi

UUV	Unmanned Underwater Vehicle
Gdl	Gradi di Libertà

Capitolo 1

Introduzione

Con lo sviluppo ed il consequente utilizzo sempre più diffuso di droni in ambito marino, a portato al problema di avere sistemi di guida che siano affidabili in una varietà abbastanza ampia di situazioni, ma che siano anche robusti e di semplice concezione per evitare problemi dovuto a sitemi troppo complessi.
Avendo questa come motivazione principale, si è scelto come oggetto di studio lo sviluppo di un sistema di guida per un drone sottomarino autonomo o UUV.
Il sistema di guida ha come due componenti principali: il modello fisico-matematico del drone basato sul modello a 4 gradi di libertà che garantisce una buona rappresentazione delle caratteristiche fisiche del drone e il controllore del percorso che è ha come base una legge di guida.
Per le caratteristiche fisiche del drone si è usato come riferimento il drone Blucy del progetto StreamER, dell'Alma Matter Studiorum Università di Bologna.
E per la creazione del sistema di guida si è usato Matlab per la scrittura dei codici e Simulink per la creazione vera è propria del sistema tramite la progettazione Model-Based.
La prima parte delle tesi si concentra sul descrivere il drone il usato come modello di riferimento, i sistemi di coordinate, la cinematica, la dinamica del drone e come tutto converga nel modello a 4 Gdl che funge da base per il sistema di guida.
Nella seconda parte si prendono tutte le scelte fatte nella prima parte e le si traducono in un modello Simulink, che ha come base il modello rappresentativo del drone, per poi essere collegato al vero e proprio sistema di guida, basato sulla legge di guida Pure Pursuit, che fa seguire al drone un percorso prestabilito. Infine si ha un sistema di frenata che ferma la simulazione quando si è raggiunta la destinazione prestabilita.
Nell'ultima parte si sono eseguite diverse simulazioni del modello, con diversi tipi di traiettorie e variando il metodo di calcolo della traiettoria teorica del drone. Poi si è usato il parametro Cross-track error per verificare l'accuratezza del percorso effettivo del drone rispetto alla traiettoria teorica e per confrontare i diversi metodi di calcolo usati.
Il sistema di guida con le caratteristiche descritte, a dimostrato attraverso le simulazioni fatte di essere preciso in diversi tipi di traiettorie e di avere un calcolo computazionale leggero e robusto, anche seguendo traiettorie complesse.

Capitolo 2

Modello Fisico Matematico

Questo capitolo illustra, il drone usato come modello di riferimento, il sistema di coordinate, la cinematica e la dinamica usate nel modello a 4 Gdl e la spiegazione delle sue equazioni e assunzioni.

2.1 Drone Blucy

Il drone usato come modello di riferimento per il sistema di guida è il drone Blucy del progetto StreamER, dell'Alma Matter Studiorum Università di Bologna, progettato per il raccoglimento di campioni biologici nel mare.

Figura 2.1: Blucy

Blucy avendo una lunghezza orizzontale maggiore di quella laterale, ha i momenti di rollio e imbardata indipendenti tra loro. E' mosso da 6 eliche, 2 per il piano verticale, 2 per il piano orizzontale e 2 per piano laterale, che posso funzionare in maniera singola o accoppiante tra di loro. Ha come massa a secco 216.45 kg e come massa bagnata 216.45 kg, nel modello a 4 Gdl si è usato la massa a secco.
I coefficienti idrodinamici, stabiliti tramite simulazioni CFD, e la massa aggiunta, trovata tramite lo strumento di ricerca AMCOMP, usati nel modello a 4 Gdl sono riassunti nella seguente tabella:

Coefficienti Idrodinamici		Massa aggiunta
Lineari $[Kg/s]$	Quadratici$[Kg/m]$	
$X_u = -2.61$	$X_{u\|u\|} = -61.82$	$X_{\dot{u}} = -28.94kg$
$Y_u = -24.72$	$Y_{v\|v\|} = -597.62$	$Y_{\dot{v}} = -166.03kg$
$Z_w = -2.82$	$Z_{w\|w\|} = -255.86$	$Z\dot{w} = -94.13kg$
$N_r = -0.044$	$N_{r\|r\|} = -375.53$	$N\dot{r} = -33.58kgm2$

2.2 Sistema di Coordinate

Per un UUV la scelta del sistema di coordinate adeguato è di primaria importanza per rappresentare in maniera corretta,la posizione,l'orientamento e velocità del drone. Per gli UUV si usano principalmente due sistemi di riferimento il sistema NED e il sistema BODY.

2.2.1 Sistema NED

Sistema mobile definito in riferimento al piano orizzontale locale, cioè il piano tangente alla superficie terreste di riferimento in un qualsiasi punto. Perciò considera come ipotesi che la Terra sia piatta e ciò semplifica le leggi Newtoniane che si applicano senza la curvatura della Terra. Il NED ha come origine il punto P e assi $n = (x_n, y_n, z_n)$:

- L'asse x_n punta al Nord geografico;
- L'asse y_n punta verso Est;
- L'asse z_n punta verso il basso perpendicolare alla superficie della Terra;

Il sistema NED è adatto anche per per path-following e la navigazione tramite wayponit, dove le traiettorie del percorso sono definite rispetto alle coordinate geografiche fisse, semplificando cosi la pianificazione del percorso che le strategie di controllo.

2.2.2 Sistema BODY

Sistema di riferimento solidale al veicolo che si muove con esso ed ha come origine il suo CG ed assi $b = (x_b, y_b, z_b)$:

- L'asse x_n punta lungo l'asse longitudinale del velivolo;
- L'asse y_n punta verso la destra del veicolo;
- L'asse z_n punta vero il basso, perpendicolare al piano del veicolo;

Il sistema BODY è essenziale nella rappresentazione della velocità e delle forze, come anche nella progettazione del sistema di controllo, dove le forze e i comandi di controllo sono applicati al sistema BODY per poi essere convertiti nel sistema NED per la navigazione.

2.3 Cinematica e Dinamica

Nella creazione del sistema di guida si è usata una forma semplificata delle equazioni a 6 gradi di libertà. Queste equazioni si concentrano sul descrivere come la posizione e la attitudine del veicolo si evolvono in base alla velocità del veicolo in assi BODY. Queste relazioni sono espresse dai vettori:

$$\eta = [\eta_1 \quad \eta_2]^T \quad \text{dove} \quad \eta_1 = [x \quad y \quad z]^T \quad \text{e} \quad \eta_2 = [\phi \quad \theta \quad \psi]^T \tag{2.1}$$

$$\upsilon = [\upsilon_1 \quad \upsilon_2]^T \quad \text{dove} \quad \upsilon_1 = [u \quad v \quad w]^T \quad \text{e} \quad \upsilon_2 = [p \quad q \quad r]^T \tag{2.2}$$

dove il termine η rappresenta la posizione del sistema BODY rispetto al sistema NED in termini di spostamento lineare con il vettore η_1 e degli angoli di Eulero con η_2. Il termine υ rappresenta nel sistema BODY la velocità lineare con il vettore υ_1 e la velocità angolare con il vettore υ_2.
La forma semplificata delle equazioni per il modello a 4 Gdl disaccoppia l'angolo di imbardata ψ dagli angoli di beccheggio θ e rollio ϕ rendendo questi ultimi trascurabili in questo tipo di modello, e di conseguenza sono trascurabili anche le velocità angolari p e q. Cosi come nuovi vettori abbiamo:

$$\eta = [\eta_1 \quad \eta_2]^T \quad \text{dove} \quad \eta_1 = [x \quad y \quad z]^T \quad \text{e} \quad \eta_2 = [0 \quad 0 \quad \psi]^T \tag{2.3}$$

$$\upsilon = [\upsilon_1 \quad \upsilon_2]^T \quad \text{dove} \quad \upsilon_1 = [u \quad v \quad w]^T \quad \text{e} \quad \upsilon_2 = [0 \quad 0 \quad r]^T \tag{2.4}$$

Ed usando le equazioni della dinamica del corpo rigido formulate da Fossen, espresse come:

$$M_{RB}\dot{\upsilon} + C_{RB}(\upsilon)\upsilon = \tau_{RB} \tag{2.5}$$

dove M_{RB} è la matrice d'inerzia del corpo rigido e C_{RB} è la matrice del corpo rigido di Coriolis. Il termine τ_{RB} è espresso come somma e delle forze e momenti esterni espressi nel sistema BODY:

$$\tau_{RB} = \tau_{hs} + \tau_{hd} + \tau_{vento} + \tau_{onda} + \tau_p \tag{2.6}$$

dove τ_{hs} e τ_{hd} sono dovuti alle forze idrodinamiche e idrostatiche, τ_{vento} e τ_{onda} sono forze e momenti dovuti alla presenza di onde e vento, che sono trascurabili dato che il drone immergendosi ne è poco influenzato e τ_p rappresenta le forze e i momenti dovuti al sistema propulsivo del drone.
Per il modello a 4 Gdl e il sistema di guida su cui si basa, la componente di τ_{RB} che viene usata è τ_p con le sue componenti lungo x e y intono all'asse z.

2.4 Modello a 4 Gdl per UUV

Si è scelto questo modello perché pur essendo un modello semplificato, è molto valido per sistemi di controllo di path-following e trajectory-tracking. Le assunzioni per cui il modello sia valido sono:

- Che la distanza tra CG e il Centro di Galleggiamento del drone sia tale da generare una forza di ripristino che compensi le oscillazioni di rollio e beccheggio, rendendo i loro momenti e forze trascurabili;

- L'accoppiamento tra l'angolo di beccheggio e il movimento lungo l'asse verticale è trascurabile, questo è possibile se il drone si muove lungo l'asse verticale tramite dei propulsori verticali che non modificano il beccheggio;
- Che la velocità di oscillazione e perturbazioni variabili nel tempo siano compensate, rispettivamente dalla forza di smorzamento idrodinamico e dalla natura del drone che ne mitiga gli effetti;

Il modello a 4 Gdl è descritto dalle seguenti equazioni: per le velocità

$$\dot{x} = u\cos\psi - v\sin\psi \tag{2.7}$$

$$\dot{y} = u\sin\psi + v\cos\psi \tag{2.8}$$

$$\dot{z} = w \tag{2.9}$$

$$\dot{\psi} = r \tag{2.10}$$

e per le accelerazioni

$$\dot{u} = \frac{1}{m_{11}}(m_{22}vr - d_{11}u + \tau_u + d_u) \tag{2.11}$$

$$\dot{v} = \frac{1}{m_{22}}(-m_{11}ur - d_{22}v + d_v) \tag{2.12}$$

$$\dot{w} = \frac{1}{m_{33}}(-d_{33}w + \tau_w + d_w) \tag{2.13}$$

$$\dot{r} = \frac{1}{m_{66}}((m_{11} - m_{22})uv - d_{66}r + \tau_r + d_r) \tag{2.14}$$

Dove: $m_{11} = m - X_{\dot{u}}, m_{22} = m - Y_{\dot{v}}, m_{33} = m - Z_{\dot{w}}, m_{66} = I_Z - N_{\dot{r}}$,
$d_{11} = X_u + X_{u|u|}|u|, d_{22} = Y_v + Y_{v|v|}|v|, d_{33} = Z_w + Z_{w|w|}|w|, d_{66} = N_r + N_{r|r|}|r|$ con m che è la massa del drone, τ_u e τ_w sono forze lungo rispettivamente x_b e z_b e τ_r è la coppia intorno a z_b. I termini d_u, d_v, d_w e d_r rappresentano gli effetti di disturbo sul sistema. I coefficienti $X_{\dot{u}}, Y_{\dot{v}}, Z_{\dot{w}}, N_{\dot{r}}$ e X_u, Y_v, Z_w, N_r e $X_{u|u|,}, Y_{v|v|}, Z_{w|w|}, N_{r|r|}$ sono rispettivamente i coefficienti di massa aggiunta e idrodinamici lineari e quadratici, caratteristici del drone Blucy.

Capitolo 3

Modello Simulink

In questo capitolo introduce lo sviluppo del sistema di guida, in ambiente Simulink. Come prima cosa si sono trasposte le equazioni del modello a 4 Gdl in un blocco Simulink, per poi passare a collegarlo alla legge di guida Pure Pursuit e in fine a implementare un sottosistema per fermare la simulazione una volta raggiunto il Target deciso tramite la legge di guida.

3.1 Implementazione del modello a 4 Gdl

Per implementare le equazioni si è deciso di dividere le equazioni, che descrivono la dinamica del drone sull'asse z e la dinamica del drone sul piano xz, in due sottosistemi separati dato che per il modello usato i movimenti sull'asse z e sul piano xy sono indipendenti l'uno dall'altro.

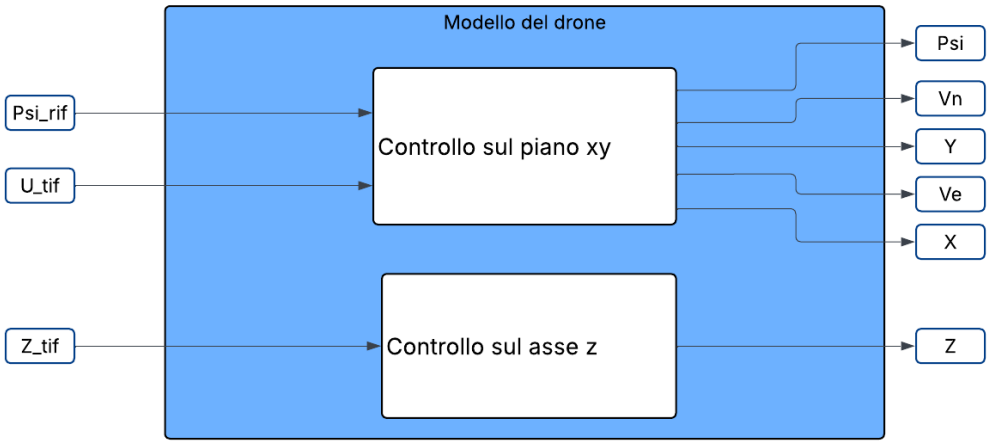


Figura 3.1: Schema del modello del drone.

Nel sottosistema inerente al controllo sul piano xy si hanno come input la velocità di riferimento u e l'angolo di sbandamento di riferimento ψ, implementando le equazioni (2.11),(2.12),(2.14) si trovano le accelerazioni, che poi vengono integrate per trovare le velocità ed usando le equazioni (2.7),(2.8),(2.10) le si converte nel

sistema NED, integrando si hanno come output l'angolo ψ del drone, le coordinate degli assi x e y del drone nel sistema NED e la velocità verso Nord V_n e la velocità verso Est V_e. Ed inserendo un controllo in retroazione si fa in modo che gli output non si discostino dai valori di riferimento.

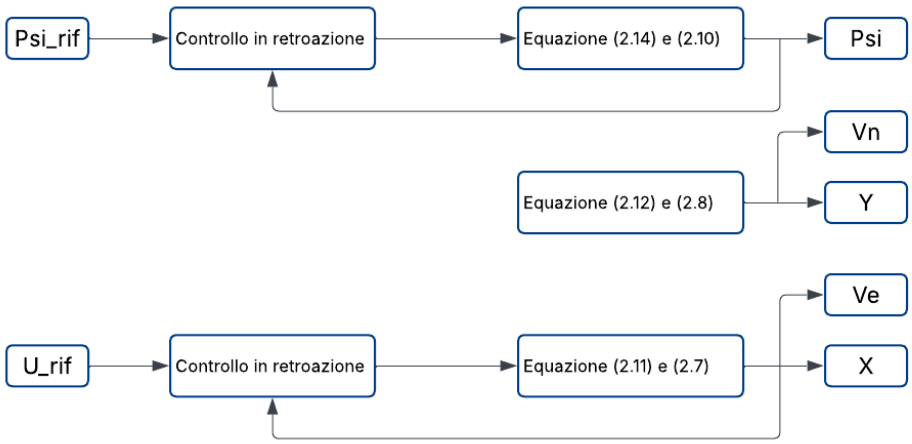


Figura 3.2: Schema del Controllo sul piano xy.

Nel sottosistema del controllo sul asse z si ha come input la z di riferimento, usando l'equazione (2.13) ed integrando si ha a velocità sull'asse verticale e con la (2.9) ed integrando si ha la coordinata dell'asse z del drone nel sistema NED. E si inserisce un controllo in retroazione si garantisce che i valore di output non si discosti dal valore di riferimento.

Figura 3.3: Schema del Controllo sul asse z.

3.2 Legge di guida

Con il termine legge di guida si intende l'algoritmo usato per la guida autonoma del drone. Quella usata per il sistema di guida, di questo lavoro, è la Pure Pursuit la cui tipologia dei problemi affrontati è di tipo Target Tracking.
Usando una legge di tipo Target Tracking si vuole che il drone raggiunga un obiettivo prefissato che sia fisso o mobile, con il drone inseguitore detto *Purser* e l'obiettivo detto *Target*. Nel caso in esame i *Target* sono punto fissi, detti *waypoint*, che il drone deve raggiungere e superare.
Per il movimento del *Puser* si suppone un modello del tipo Uniciclo 2D, cioè un

modello di un punto che si muove su un piano, a quota costante e per controllare la guida si interviene sulla direzione della velocità rispetto all'angolo di imbardata ψ, con assenza di vento di basa sul seguente sistema:

$$\begin{cases} \dot{N} = V_a \cos\psi \\ \dot{E} = V_a \sin\psi \\ \dot{\psi} = \Omega \end{cases} \tag{3.1}$$

dove Ω è il rateo di virata comandato per direzionare l'angolo d'imbardata.

3.2.1 Pure Pursuit

Descrivendo adesso la legge di guida Pure Pursuit, essa si basa sul puntare la direzione della velocità del *Pursuer* nella stessa direzione della linea di vista o Line Of Sight, LOS, che è il vettore tra il *Purser* e il *Target*. Per fare ciò impongo:

$$\dot{\psi} = \Omega_{PP} \tag{3.2}$$

con

$$\Omega_{PP} = k\alpha \tag{3.3}$$

dove

$$\alpha = \psi_{LOS} - \psi \tag{3.4}$$

che è l'angolo tra il vettore velocità del *Purser* e la LOS mentre k è un costante di proporzionalità.

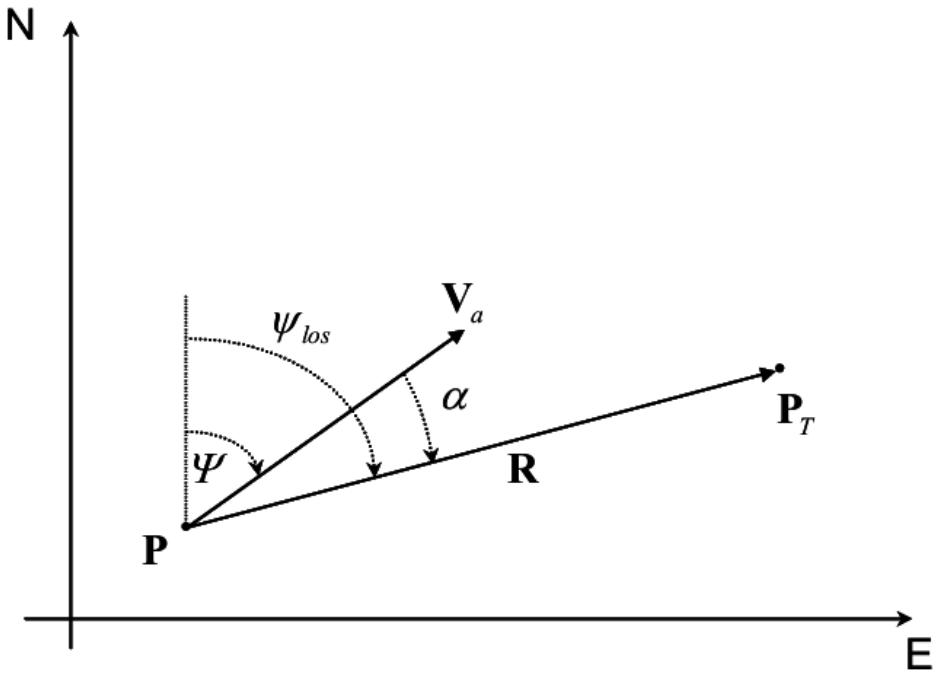


Figura 3.4: Cinematica della Pure Pursuit.

Definendo il vettore posizione relativa Pursuer-Target come il vettore differenza tra le due posizioni, $\mathrm{R} = \mathrm{P}_T - \mathrm{P}$, $R = \|\mathrm{R}\|$ e $V_a = \|\mathrm{V}_a\|$ si ha:

$$\sin\alpha = \frac{\mathbf{V}_a \times \mathbf{R} \cdot \hat{k}}{V_a R} \tag{3.5}$$

$$\cos\alpha = \frac{\mathbf{V}_a \cdot \mathbf{R}}{V_a R} \tag{3.6}$$

dove con $\hat{k}$ si indica il versore dell'asse Down. Pertanto si ha utilizzando l'arcotangente a quattro quadranti:

$$\alpha = \arctan_2(\mathbf{V}_a \times \mathbf{R} \cdot \hat{k}, \mathbf{V}_a \cdot \mathbf{R}) \tag{3.7}$$

da cui si ottiene la legge di guida Pure Pursuit, $\Omega_{PP} = k\alpha$.
Questa strategia è equivalente al modello preda-predatore ed è facile da implementare, però ha il suo principale svantaggio nel fatto che essa è influenzata negativamente dalla presenza di disturbi esterni o dal fatto che il target possa muoversi: infatti se c'è vento e/o il target si muove la Pure Pursuit non è in grado di compensare gli effetti misurando la velocità relativa tra Target e Pursuer, portando ad un inseguimento poco efficace.
Nel modello Simulink la legge di guida Pure Pursuit viene implementata inserendo come input nel calcolo di α, la V_n, la V_e, la x e la y presi dal modello del drone e i waypoint scelti. Dopo di che α viene moltiplicata per k e si ha la Ω_{pp}, che poi passa per un blocco integratore da cui si ricava la ψ_{rif}, che va nel Modello del drone come input.

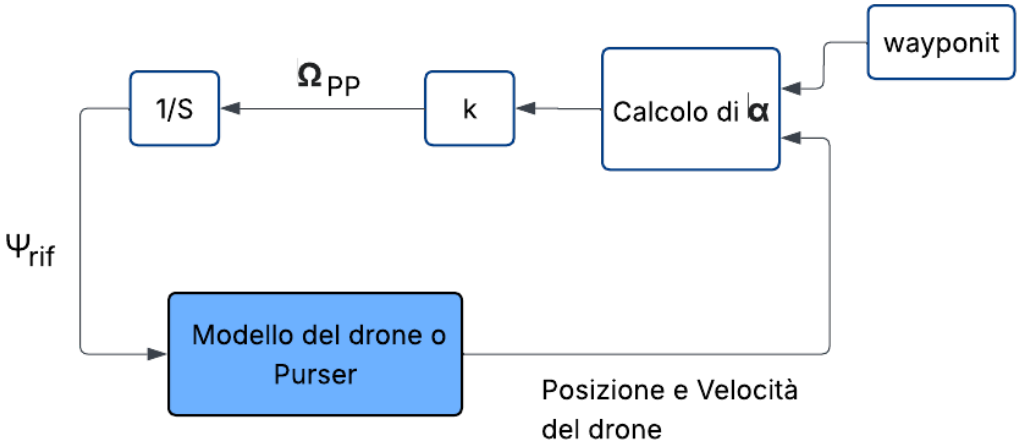


Figura 3.5: Schema a blocchi riassuntivo del sistema di guida basato sul Pure Pursuit

Il blocco per il calcolo della α è un blocco Matalb function il cui script è nel paragrafo A.1 dell'appendice. Nel codice si calcolano anche i cross-track errors lungo gli assi x e y, che sono utili per capire l'errore di calcolo durante la simulazione.

3.2.2 Cross-track Error

Per una linea che parte dal punto di riferimento (x^n_{ref}, y^n_{ref}) fino alla posizione del $Target$ (x^n_t, y^n_t) l'angolo tangente al percorso si calcola con:

$$\pi_p = \arctan_2(y^n_t - y^n_{ref}, x^n_t - x^n_{ref}) \tag{3.8}$$

Per un veicolo nella posizione (x^n, y^n), il cross-track error è calcolato come la distanza ortogonale tra il sistema di riferimento tangente al percorso con l'origine.

$$x_e = (x^n - x^n_{ref})cos(\pi_p) + (y^n - y^n_{ref})\sin(\pi_p) \tag{3.9}$$

$$y_e = -(x^n - x^n_{ref})\sin(\pi_p) + (y^n - y^n_{ref})\cos(\pi_p) \tag{3.10}$$

Cosi si ricava quant'è l'errore che il drone commette lungo il suo percorso rispetto alla traiettoria teorica, traccia a partire dai waypoint.

3.3 Sistema di controllo di stop della simulazione

Per fermare la simulazione una volta raggiunto l'ultimo wayponit detto, in questo caso, $Target$, si è inserito un sottosistema nel modello Simulink che verifica la distanza d tra l'ultimo wayponit (x^n_t, y^n_t) e la posizione del drone (x^n, y^n), per poi confrontarla con una costante scelta arbitrariamente.

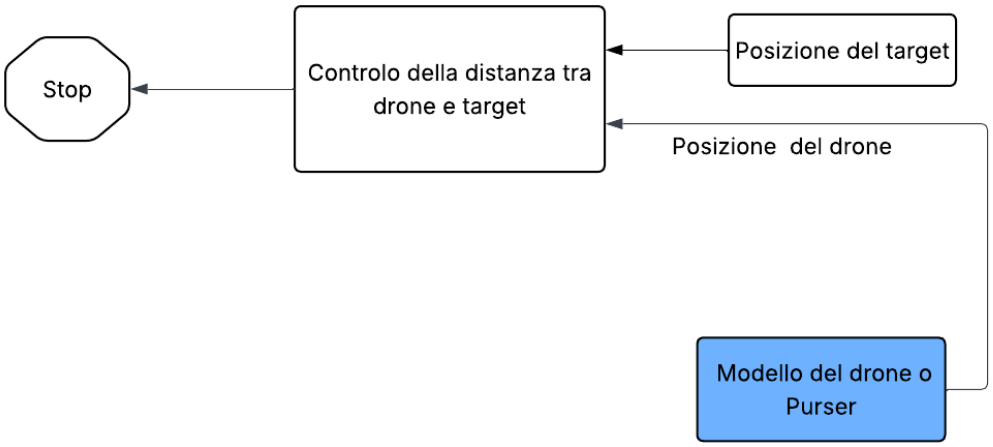


Figura 3.6: Schema a blocchi del sistema di frenata

Il blocco per controllo della distanza usa la seguente formula per calcolare d:

$$d = \sqrt{(x^n - x^n_t) \cdot (y^n - y^n_t)} \tag{3.11}$$

Se nel confrontare la costante con la d, quest'ultima risultasse più piccola si attiva il blocco Stop che fa fermare la simulazione. Così il drone si arresta in un intorno del $Target$, una circonferenza che ha come raggio la costante scelta.
Quindi più il valore della costante è minore, minore è la distanza tra il punto di frenata del drone e il $Target$ e maggiore è il grado di precisione della frenata.

Se la costante è troppo piccola il drone potrebbe non entrare nel perimetro del cerchio, non fermandosi e proseguendo fino alla fine della simulazione. Allora bisogna trovare il compromesso tra grado di precisione voluto e la possibilità che il drone si fermi. Per questo avere un traiettoria il più possibile aderente ai waypoint è molto importante.
Le costanti usate nelle simulazioni del prossimo capitolo sono state ricavate iterando la simulazione fino a trovare la più piccola che permetta al drone di fermasi.

Capitolo 4

Simulazione

In questo capitolo si illustrano i risultatati delle simulazioni condotte usando diversi tipi di traiettorie sul piano xy e diversi metodi di calcolo delle stesse e delle prove di movimento verticale lungo l'asse z. Per ogni caso preso in esame il punto di origine è 0,0,0 degli assi x, y, z e per le traiettorie sul piano xy si è immessa una velocità costante di 0.1 m/s, invece per il movimento verticale si è settata una distanza da raggiungere.
Le traiettorie simulate sono: parabolica, circolare e una composta da un sistema di generiche coordinate xy. I metodi di calcolo usati per le trattorie, paraboliche e circolari sono le loro espressioni numeriche e per il sistema di coordinate si è usato l'interpolazione lineare, in alternativa ai metodi sopracitati si è usato l'interpolazione Hermitespline. Nel paragrafo A.2 dell'appendice si trova il codice usato per calcolare le traiettorie, che vengono poi come wayponit nel blocco per il calcolo del α.
Per stabilire l'accuratezza del percorso effettivo del drone rispetto alla traiettoria calcolata e per comparare i diversi metodi di calcolo usati, si è usato il cross-track error per le traiettorie sul piano orizzontale. I grafici delle simulazioni di seguito mostrati sono stati fatti tramite il codice nel paragrafo dell'appendice A.3.

4.1 Sistema di Coordinate

Il sitema è composto dalle seguenti coordinate: sull'asse x [0;1;2;5;13;18;25;30;30] e sull'asse y [0;1.5;3;6;15;25;30;25;15] e unendo le coordinate con l'interpolazione lineare e con l'interpolazione Hermitespline si hanno delle traiettorie come in figura:

(a) Interpolazione lineare

(b) Interpolazione Hermitespline

Figura 4.1: Traiettorie del drone

Ed inserendo le due traiettorie nel modello Simulink si ha l'effettivo percorso del drone per ognuna di esse:

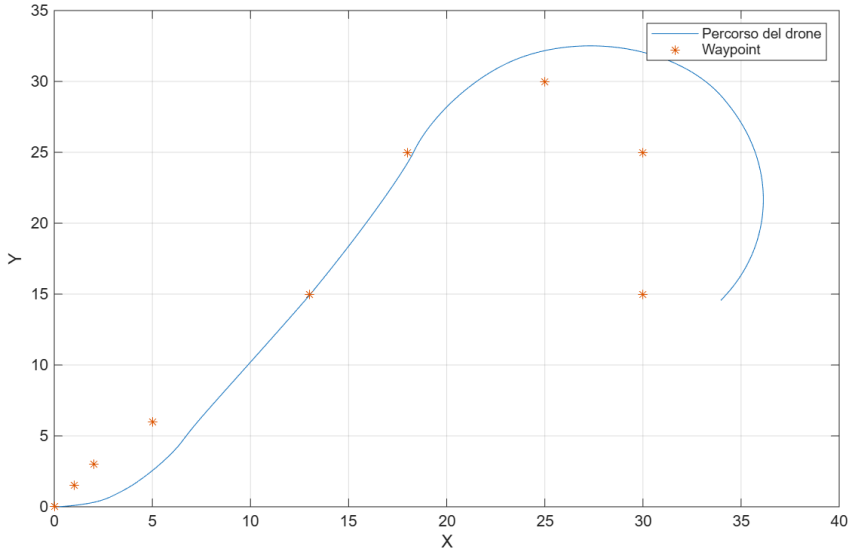


Figura 4.2: Percorso del drone con l'interpolazione lineare

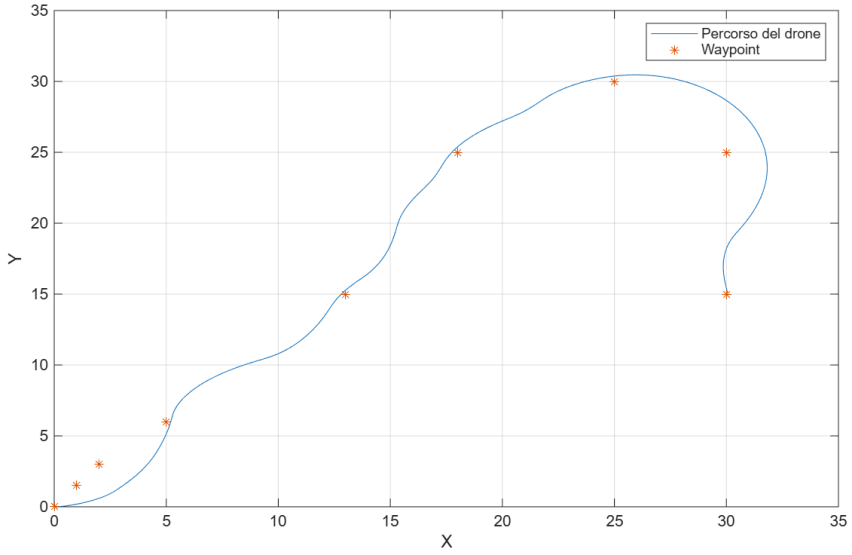


Figura 4.3: Percorso del drone con l'interpolazione Hermitespline

Dalla figura 4.1b si può notare come l'interpolazione Hermitespline riesca a rappresentare molto bene le parti più curve del percorso rispetto all'interpolazione lineare, che essendo del primo ordine approssima le curve come linee dritte tra due punti. Questo diverso grado di approssimazione del percorso del drone con l'interpolazione lineare nella figura 4.2 tende ad essere molto più impreciso e a reagire male ai cambi di direzione, con una costante di frenata uguale a 4.
Il percorso del drone con l'interpolazione Hermitespline in figura 4.3 tende a seguire

meglio i cambi di direzione, con una costante di frenata di 0.1.
Confrontando il Cross-track error dei due metodi di interpolazione

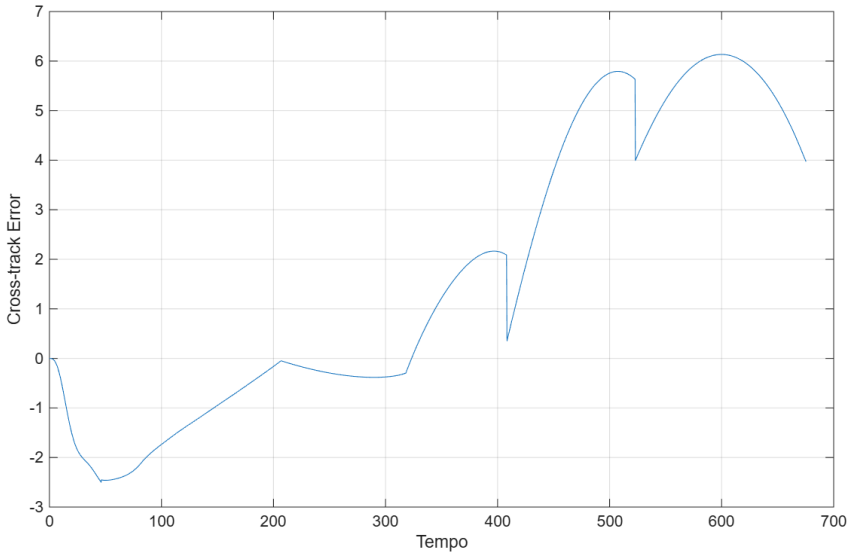


Figura 4.4: Cross-track error dell'interpolazione lineare

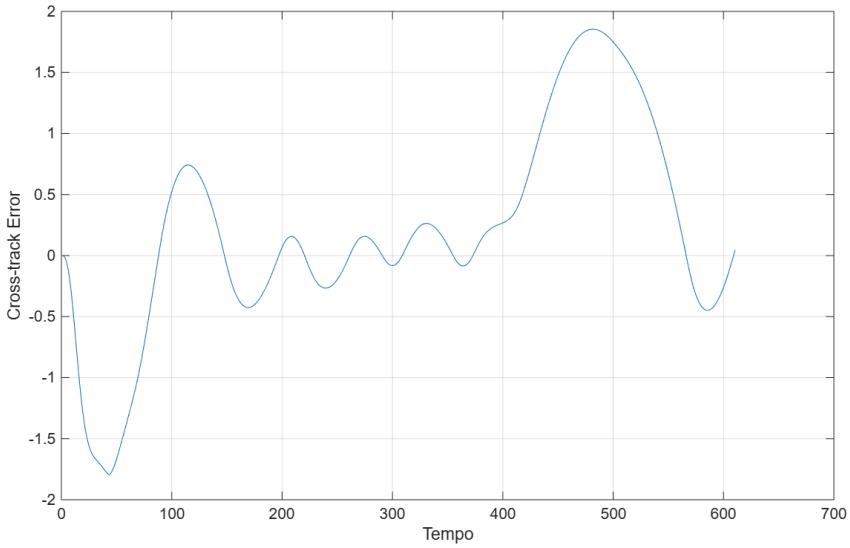


Figura 4.5: Cross-track error dell'interpolazione Hermitespline

si nota che il Cross-track error dell'interpolazione Hermitespline nella figura 4.5 ha un andamento molto più costante e con una tendenza a convergere molto meglio rispetto al Cross-track error dell'interpolazione lineare in figura 4.4, che ha un andamento poco costante che tende ad ad aumentare sopratutto in presenza di curve. Si ha anche che la media del Cross-track error dell'interpolazione lineare, 1.6481, è più alta di quello interpolato con Hermitespline che è di 0.2214, con picchi più alti e più prolungati.
Quindi si deduce che se la traiettoria e composta da una serie di di coordinate,

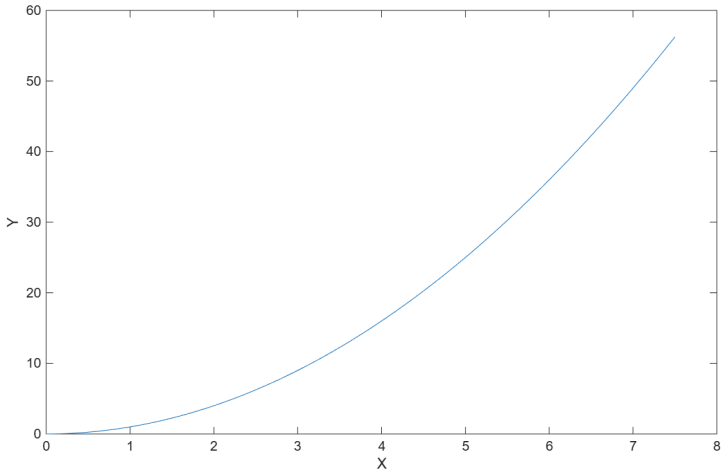
l'accuratezza del sistema di guida dipende fortemente da come queste coordinante sono state interpolate tra loro.

4.2 Traiettoria Parabolica

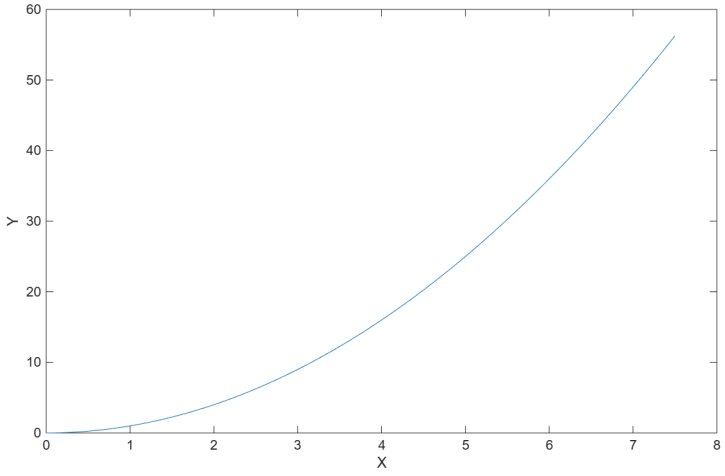
L'espressione matematica della parabola usata per la simulazione è:

$$\begin{cases} x = 0.5 * t \quad \text{con } t \in [0, 15] \\ y = x^2 \end{cases} \tag{4.1}$$

Usando l'espressione matematica e l'interpolazione Hermitespline si hanno le seguenti traiettorie:



(a) Funzione matematica



(b) Hermitespline

Figura 4.6: Traiettorie paraboliche del drone

Inserendole queste traiettorie nel modello Simulink si hanno i seguenti percorsi reali del drone:

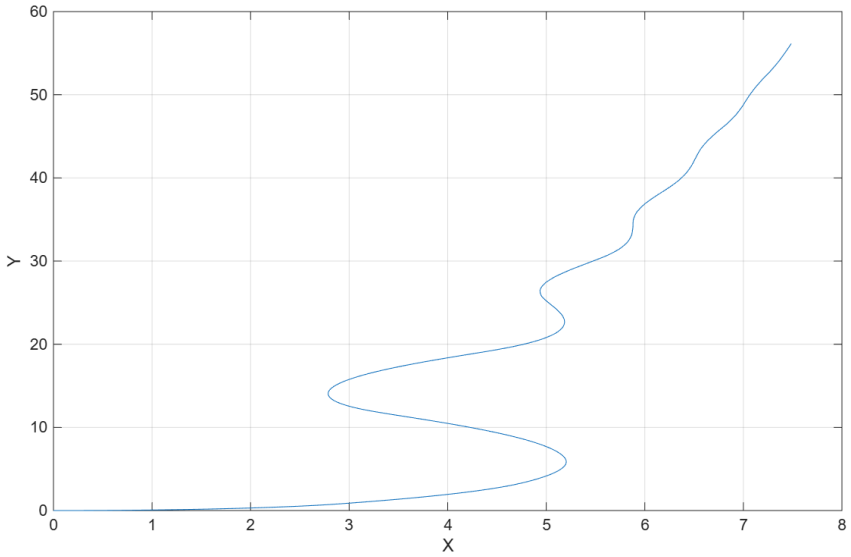


Figura 4.7: Percorso del drone usando l'espressione numerica della parabola

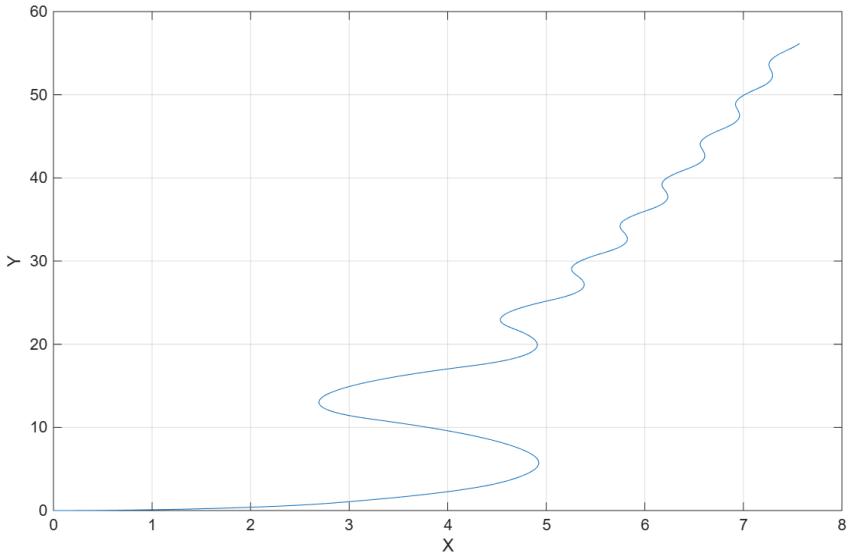


Figura 4.8: Percorso del drone con l'interpolazione Hermitespline della parabola

Già guardando i grafici in figura 4.6 si può notare la grande somiglianza tra le due traiettorie, che si riflette sui reciproci percorsi fatti dal drone. Infatti il due percorsi hanno la medesima costante di frenata, 0.1, con l'unica differenza che dopo le iniziali curve il percorso con l'interpolazione Hermitespline, ha un andamento oscillatorio, dovuta alla maggiore precisione nel interpolare le curve.
Questa differenza si nota anche nel confronto tra i Cross-track error dei due percorsi, dove l'andamento del del Cross-track error dell'interpolazione Hermitespline, in figura 4.10, oscilla in maniera costante dopo i picchi, a differenza del Cross-track error della funzione matematica, in figura 4.9, che dopo i picchi tende a convergere verso lo zero. Anche le medie dei Cross-track error sono molto simili, -0.3868 per la funzione numerica e -0.3318 per Hermitespline.

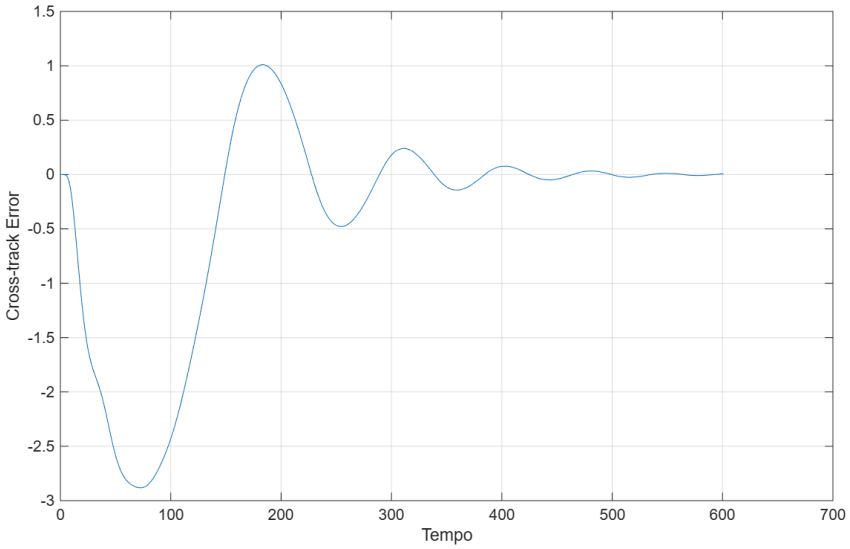


Figura 4.9: Cross-track error della funzione matematica della parabola

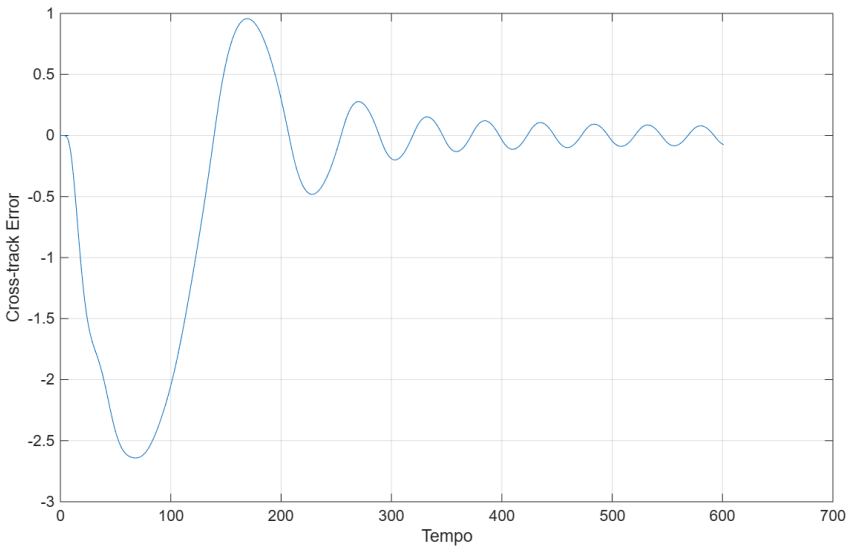


Figura 4.10: Cross-track error dell'interpolazione Hermitespline della parabola

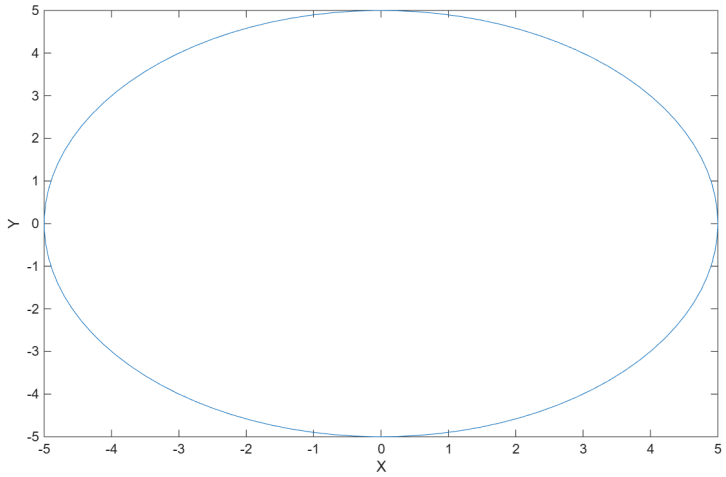
Avendo sia la traiettoria,il percorso effettivo del drone e una medi del Cross-track error molto simile, si può dire che l'accuratezza del sistema di guida, per traiettorie paraboliche, dipende molto poco da come viene interpolata la traiettoria.

4.3 Traiettoria Circolare

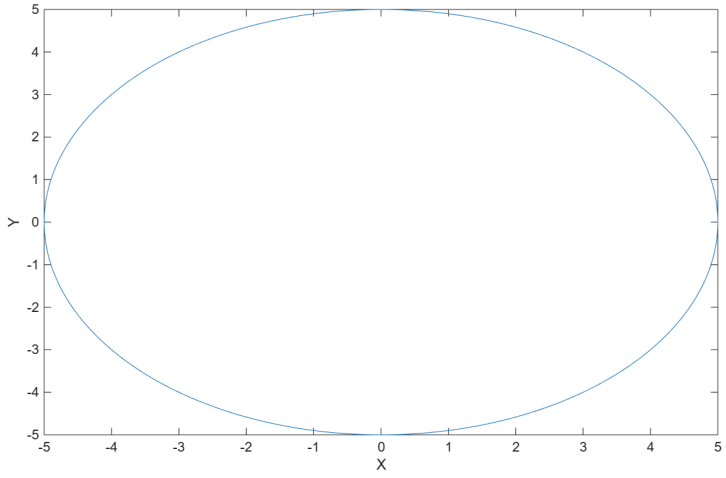
La circonferenza usata per la simulazione ha il raggio di cinque metri e come centro il punto $0, 0$, con la seguente espressione numerica:

$$\begin{cases} x = 5 * \cos\theta \\ y = 5 * \sin\theta \end{cases} \quad \text{con } \theta \in [0, 2 * \pi] \tag{4.2}$$

Le traiettorie che si ricavano dalla funzione numerica e l'interpolazione Hermite-spline sono le seguenti:



(a) Funzione matematica



(b) Hermitespline

Figura 4.11: Traiettorie circolari del drone

Immettendole nel modello Simulink si hanno percorsi reali del drone, riportati nelle figure 4.12 e 4.13. Confrontando le traiettorie ricavate dai due metodi di calcolo si denota la loro estrema somiglianza, che si riflette anche nel percorso effettivo del drone, con poca differenza tra i due percorsi. Questa somiglianza si ha anche dal fatto che la costante di frenata dei due casi è molto simile, per il percorso con la funzione numerica è di 0,16 e per il percorso con l'interpolazione Hermitespline è di 0.26. Confrontando anche i Cross-track error, rappresentati nei grafici nelle figure 4.14 e 4.15, si nota un andamento simile, supportato anche dalla bassa differenza delle loro medie che sono, -2,3669 per il Cross-track error della funzione numerica e -2,1519 per il Cross-track error dell'interpolazione hermitespline.

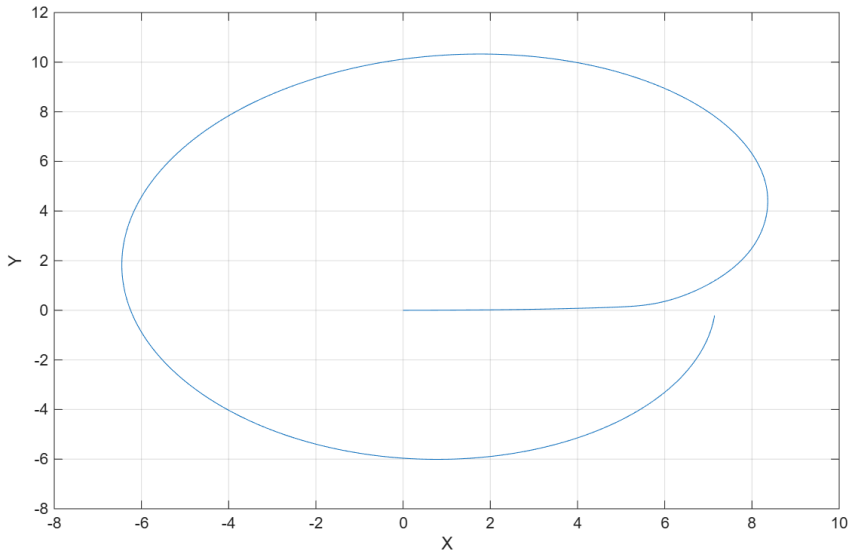


Figura 4.12: Percorso del drone usando l'espressione numerica del cerchio

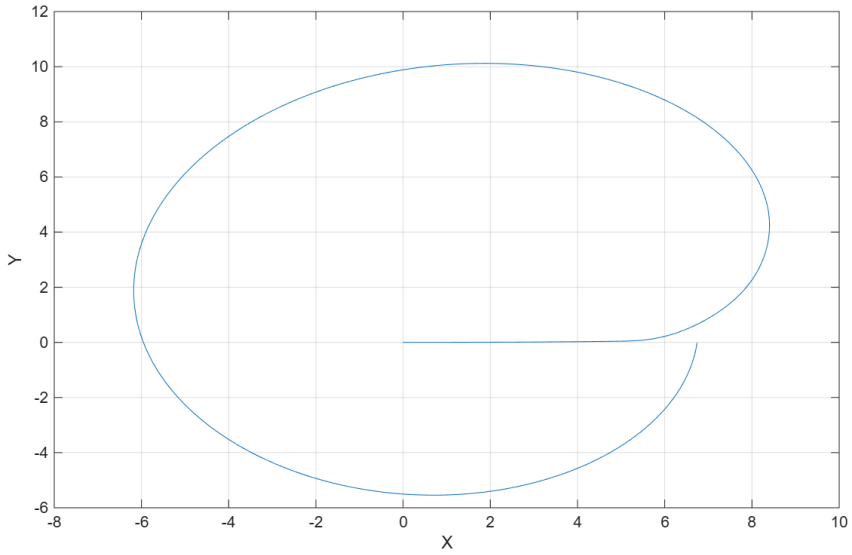


Figura 4.13: Percorso del drone con l'interpolazione Hermitespline del cerchio

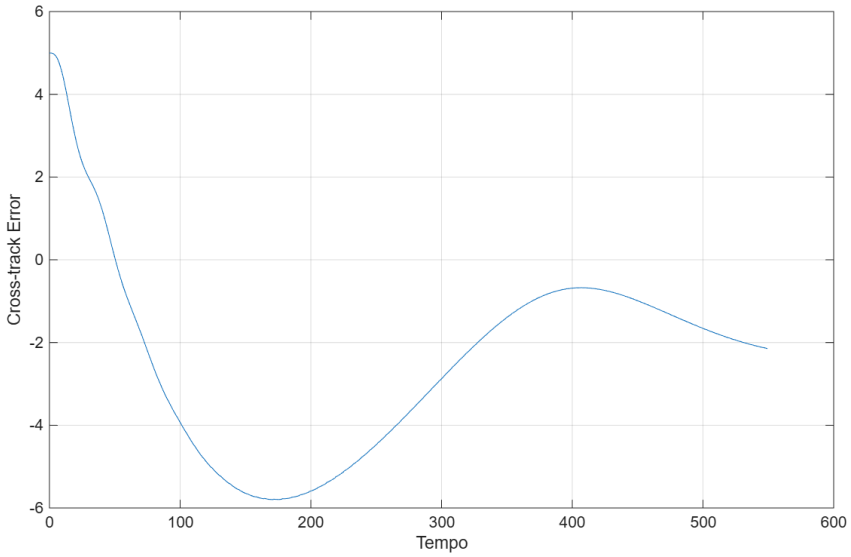


Figura 4.14: Cross-track error della funzione matematica del cerchio

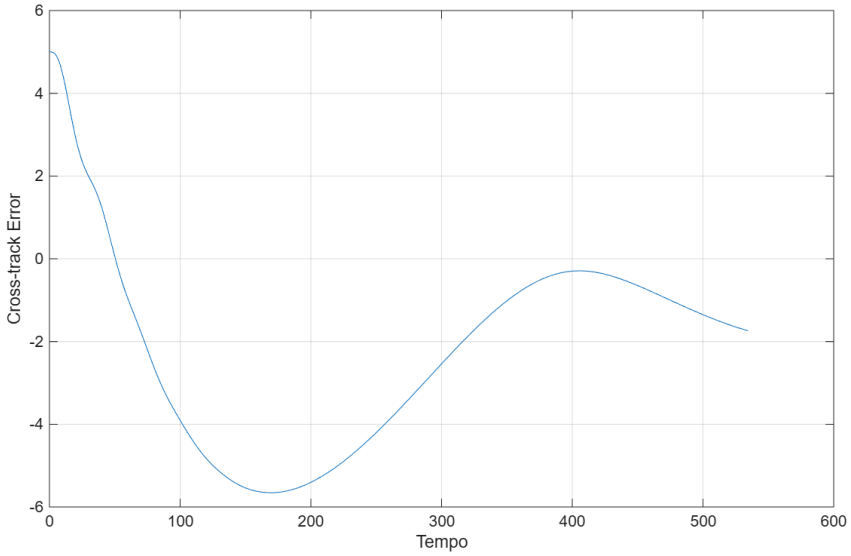


Figura 4.15: Cross-track error dell'interpolazione Hermitespline del cerchio

Quindi si può dedurre, in maniera simile alla traiettoria parabolica, che avendo traiettorie, percorsi reali molto simili con costanti di frenata anche esse simili tra loro, con anche poca differenza tra l'andamento dei Cross-track error e le loro medie, che per traiettorie circolari il sistema di guida dipende molto poco dal modo in cui viene interpolata la traiettoria.

4.4 Movimento sull'asse verticale

Per rispettare le ipotesi del modello a 4 Gdl il movimento lungo l'asse z avviene tramite propulsori verticali e con le velocità sul piano xy nulle. Per la simulazione il

drone deve raggiungere una distanza di 20 m, cioè arrivare nel punto di coordinate $0, 0, 20$. Il percorso del drone è rappresentato dalla seguente figura:

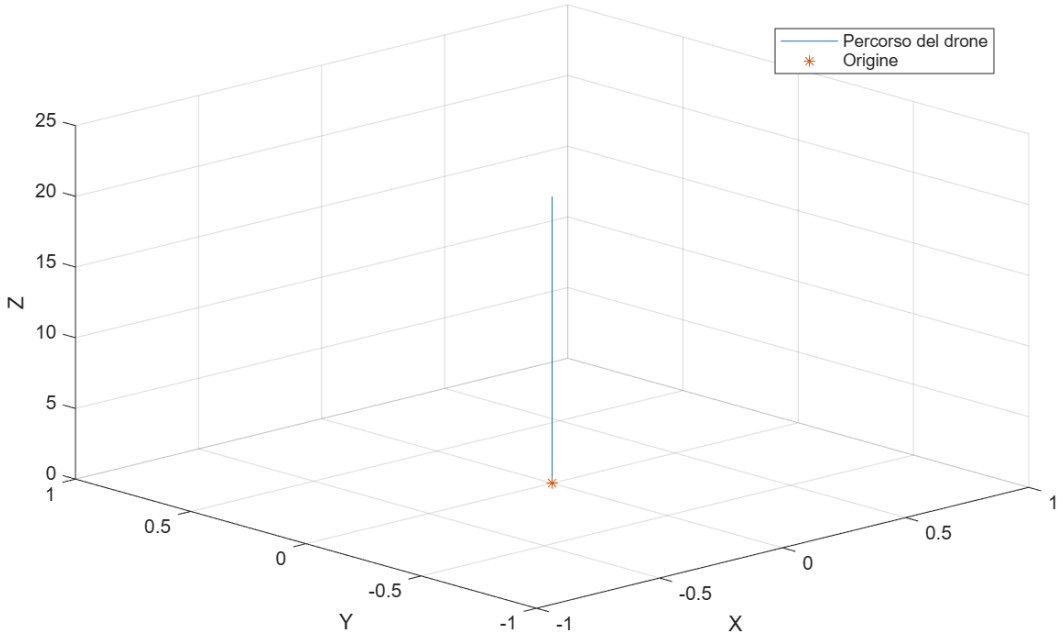


Figura 4.16: Percorso verticale del drone

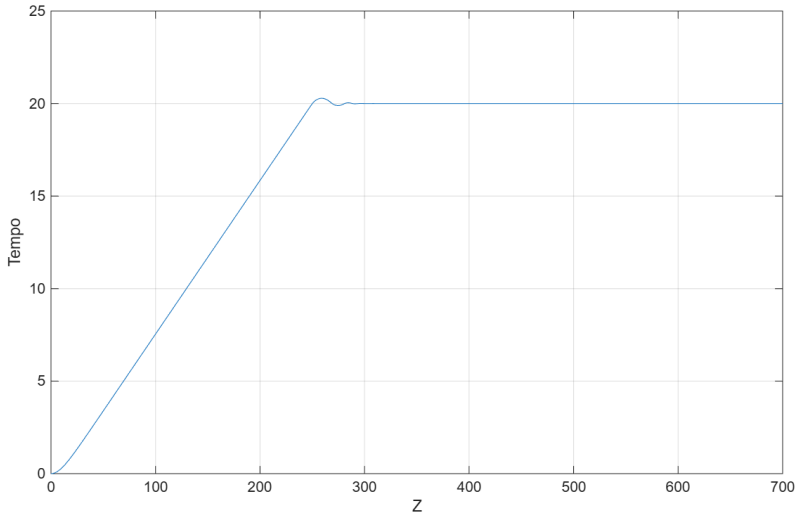


Figura 4.17: Percorso lungo l'asse z rispetto al tempo

In figura 4.17 si può notare come il movimento lungo l'asse z sia lineare fino a 250 s, per poi iniziare ad oscillare per poi convergere verso il 300 s. Con questa robustezza la componente del movimento verticale del sistema di guida si può definire affidabile data anche la sua semplicità.

Conclusioni

Dai risultati delle simulazioni fatte si può affermare che il sistema di guida, presentato e descritto in questa tesi, sia affidabile in diversi tipi si percorsi. Ciò è possibile usando un modello fisico-matematico relativamente semplice e con una legge di guida anche essa relativamente semplice nel concetto e nelle basi matematiche.
Un importate a aggiunta è stato la adozione dell'interpolazione Hermitespline, che ha permesso di avere un percorso del drone molto più aderente alla traiettoria teorica, sopratutto per un sistema di coordinate generico di coordinate, che può essere facilmente un tipo di traiettoria molto comune per questo tipo di droni.
Tuttavia questo sistema ha dei limiti che posso portare un notevole margine di miglioramento. Uno di questi limiti è che il sistema per fermasi dipenda da una costante che viene trovata iterando varie simulazioni, un importante miglioramento sarebbe trovare a priori la più piccola costante di frenata senza ripete la simulazione più volte. Un ulteriore miglioramento potrebbe essere l'uso di ulteriori algoritmi di interpolazione, per compiti specifici e per aumentare il campo di utilizzo del drone si potrebbero implementare l'uso i sensori per la guida anche in un ambiente che presenta ostacoli da aggirare.
Nonostante i limiti e i possibili miglioramenti sopraccitati, rispetta pienamente le caratteristiche ricercate in questo tipo di applicazione.

Appendice A

Codice Matlab

A.1 Calcolo della α

```
1  function [alfa,x_e,y_e]   =  alfa(Vn,Ve,x,y,waypoint)
2
3  k1=[0 0 1];
4  V=[Ve Vn 0];
5  P=[x y 0];
6  wpt.pos.x= waypoint(:,1);
7  wpt.pos.y= waypoint(:,2);
8  R_switch=0.001;
9  persistent k;         % active waypoint index, initialized
       by 'clear ILOSpsi'
10 persistent xk yk;     % active waypoint (xk, yk)
      corresponding to integer k
11
12
13 %% Initialization of (xk, yk) and (xk_next, yk_next), and
       integral state
14 if isempty(k)
15
16     % Check if R_switch is smaller than the minimum
          distance between the waypoints
17     if R_switch > min( sqrt( diff(wpt.pos.x).^2 + diff(
          wpt.pos.y).^2 ) )
18         error("The distances between the waypoints must
              be larger than R_switch");
19     end
20
21     % Check input parameters
22     if (R_switch < 0); error("R_switch must be larger
          than zero"); end
23
```

```
24      k = 1;                    % set first waypoint as the
            active waypoint
25      xk = wpt.pos.x(k);
26      yk = wpt.pos.y(k);
27      fprintf('Active waypoints:\n')
28      fprintf('  (x%1.0f, y%1.0f) = (%.2f, %.2f) \n',k,k,xk
            ,yk);
29
30  end
31
32  %% Read next waypoint (xk_next, yk_next) from wpt.pos
33  n = length(wpt.pos.x);
34  if k < n                          % if there are more
        waypoints, read next one
35      xk_next = wpt.pos.x(k+1);
36      yk_next = wpt.pos.y(k+1);
37  else                              % else, continue with
        last bearing
38      bearing = atan2((wpt.pos.y(n)- wpt.pos.y(n-1)), (wpt.
            pos.x(n)-wpt.pos.x(n-1)));
39      R = 1e10;
40      xk_next = wpt.pos.x(n) + R * cos(bearing);
41      yk_next = wpt.pos.y(n) + R * sin(bearing);
42  end
43
44  %% Compute the path-tangnetial angle w.r.t. North
45  pi_h = atan2( (yk_next-yk), (xk_next-xk) );
46
47  % Along-track and cross-track errors (x_e, y_e)
48  x_e =  (x-xk) * cos(pi_h) + (y-yk) * sin(pi_h);
49  y_e = -(x-xk) * sin(pi_h) + (y-yk) * cos(pi_h);
50
51  % If the next waypoint satisfy the switching criterion, k
        = k + 1
52  d = sqrt( (xk_next-xk)^2 + (yk_next-yk)^2 );
53  if ( (d - x_e < R_switch) && (k < n) )
54      k = k + 1;
55      xk = xk_next;          % update active waypoint
56      yk = yk_next;
57      fprintf('  (x%1.0f, y%1.0f) = (%.2f, %.2f) \n',k,k,xk
            ,yk);
58  end
59  Pt=[xk_next yk_next 0];
60  R=Pt-P;
61  alfa=atan2d(dot(cross(V,R),k1),dot(V,R));
```

A.2 Traiettoria del drone

```
Traiettoria=input('scegli la traiettoria del drone:');
switch  Traiettoria
    case 1
        disp('Sistema di coordinate')
        waypoint = [0   0 0;
            1 1.5 0;
            2 3 0;
            5   6 0;
            13   15 0;
            18   25 0;
            25   30 0;
            30 25 0];
        wpt.pos.x = waypoint(:,1).';
        wpt.pos.y = waypoint(:,2).';
        xRef = waypoint(:,1);
        yRef = waypoint(:,2);
        posizionetarget=[xRef(end) yRef(end) 0];
    case 2
        disp('Parabola')
        t=linspace(0,15,100);
        x=0.5.*t;
        y=x.^2;
        wpt.pos.x = x.';
        wpt.pos.y = y.';
        waypoint=[x.' y.'];
        xRef = x;
        yRef = y;
        posizionetarget=[xRef(end) yRef(end) 0];
    case 3
        disp('Cicolare')
        theta=linspace(0,2*pi,100);
        x = 5 * cos(theta);
        y = 5 * sin(theta);
        wpt.pos.x = x.';
        wpt.pos.y = y.';
        waypoint=[x.' y.'];
        xRef = x;
        yRef = y;
        posizionetarget=[xRef(end-2) yRef(end-2) 0];
end
[w_path, x_path, y_path, dx_path, dy_path, pi_h, pp_x,
   pp_y, N_forward,kappa] = Hermitespline(wpt, 0.1, 1);
pathHermitespline=[x_path.' y_path.' zeros(length(x_path)
   ) zeros(length(x_path))];
```

A.3 Grafici delle simulazioni

```
figure(1)
plot3(out.x,out.y,out.z,"-");grid;hold on;plot(0,0,'*');
   hold off
xlabel('X')
ylabel('Y')
zlabel('Z')
legend('Percorso del drone','Origine')
figure(2)
plot(out.tout,out.z,'-');grid
ylabel('Tempo')
xlabel('Z')
figure(3)
plot(out.x,out.y);grid
ylabel('Y')
xlabel('X')
legend('Percorso del drone','Waypoint')
figure(4)
plot(out.tout,out.Cross_track_error,'-');grid
xlabel('Tempo')
ylabel('Cross-track Error')
figure(5)
plot(x_path,y_path,'-');
ylabel('Y')
xlabel('X')
figure(6)
plot(waypoint(:,1).',waypoint(:,2).','-');
ylabel('Y')
xlabel('X')
meadia_Cross_track_error=mean(out.Cross_track_error);
```

Bibliografia

[1] Menghini, Massimiliano (2025) Neuro adaptive guidance and control system of a multipurpose autonomous underwater vehicle for underwater environment monitoring, [Dissertation thesis], Alma Mater Studiorum Università di Bologna. Dottorato di ricerca in Ingegneria e tecnologia dell'informazione per il monitoraggio strutturale e ambientale e la gestione dei rischi - eit4semm, 37 Ciclo. DOI 10.48676/unibo/amsdottorato/11908.

[2] M. Menghini, S.K. Mallipeddi, L. De Marchi, and P. Castaldi. "Modelling and parameter identification of UUV Blucy for control design: A benchmark model". In: Ocean Engineering 313 (2024), p. 119617. issn: 0029-8018. doi: https://doi. org/10.1016/j.oceaneng.2024.119617. url: https://www.sciencedirect. com/science/article/pii/S002980182402955X

[3] Zanzi Matteo, *Leggi di Guida* ,Alma Mater Studiorum Università di Bologna,2026.

[4] Thor I. Fossen. *Handbook of Marine Craft Hydrodynamics and Motion Controle.* WILEY,2ª edizione, 2021.

[5] Thor I. Fossen. *MSS.* GitHub. 29/06/2026. `https://github.com/cybergalactic/MSS`.Consultato il: 20/04/2026.

[6] MathWorks. *Follow Waypoints in Simulink Using Pure Pursuit Block.* MathWorks.com. 17/04/2026. `https://it.mathworks.com/help/nav/ug/follow-waypoints-using-pure-pursuit-block.html`. Consultato il: 18/04/2026.