UNIVERSITÀ
POLITECNICA
DELLE MARCHE

Facoltà di Ingegneria
Corso di Laurea Magistrale in Ingegneria Meccanica

Regressione e controllo della dinamica di un drone

Candidato:
Lorenzo Lattanzi

Relatore:
Dott. Pierpaolo Belardinelli

Anno Accademico 2020-2021

UNIVERSITÀ
POLITECNICA
DELLE MARCHE

Facoltà di Ingegneria
Corso di Laurea Magistrale in Ingegneria Meccanica

Regressione e controllo della dinamica di un drone

Candidato:
Lorenzo Lattanzi

Relatore:
Dott. Pierpaolo Belardinelli

Anno Accademico 2020-2021

Università Politecnica delle Marche
Facoltà di Ingegneria
Corso di Laurea Magistrale in Ingegneria Meccanica
Via Brecce Bianche – 60131 Ancona (AN), Italy

Sommario

Nel lavoro si utilizzano le tecniche di machine-learning applicate al caso studio di un drone. All'inizio viene descritto il modello dinamico del drone, che è composto da 12 equazioni differenziali di primo grado, di cui sei di posizione e sei di velocità. Poi si passa allo studio e all'applicazione di tecniche di regressione per la traiettoria: i modelli che sono stati scelti sono quelli gaussiani e le reti neruali. L' obiettivo della regressione è andare a diminuire in maniera esponenziale il calcolo computazionale. Si pensi di simulare l'andamento di una sola variabile dell' oggetto preso come studio, per esempio l'altezza del drone, facendo variare un solo parametro,come la velocità del vento lungo una direzione. Anche in questa semplice situazione sarebbero necessarie simulazioni molto onerose, perché il parametro velocità del vento può avere una mole di dati molto elevata. La regressione appunto è stata scelta per ovviare a questo problema. Tenendo in considerazione che una variabile dipende da tutti i parametri che infuliscono nel sistema, si è estesa la trattazione del problema vedendo l'influenza della velocità lungo le direzioni x e y nei confronti dell'altezza del drone. Tutti i risultati ottenuti da due codici, uno che è stato scritto e l'altro che è stato fatto utilizzando libirerie già esistenti, sono stati confrontati sia nel caso monodimensione sia bidimensionale. L'ultima parte dell'elaborato tratta il controllo del drone. Per questo problema è stato scelto un MPC (model predictive control) a cui è stato associtato un algortimo di regressione sparsa (SINDYc). Si utilizza questo sistema affinché si riesca a ricavare con continuità il modello che l'MPC tenta di controllare.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

Lo sviluppo tecnologico ha un'andamento esponenziale, e i recenti progressi nel campo dell'areodinamica, dei micro componenti e di tutta la parte sensorisca, hanno contribuito a uno sviluppo capillare dei droni. Le piccole dimensioni, il basso costo e la manovrabilità di questi sistemi li hanno resi soluzioni potenziali in una vasta classe di applicazioni. Le loro caratteristiche li rendono anche una sfida, perché hanno difficoltà di controllo e di gestione significative. A livello mondiale si sta andando sempre di più verso le guide autonome, senza pilota sia a livello automobilistico sia a livello aeromobile. I droni stanno prendende sempre più piede nell'ambito della diagnostica e del soccorso, poiché favoriti dalla loro struttura compatta. Il funzionamento autonomo dei veicoli aerei si basa sulla stabilizzazionea bordo e sulle capacità di tracciamento della traiettoria. Questi problemi sono aggravati su piccola scala, poiché il veicolo è più suscettibile agli effetti ambientali(vento, temperatura, ecc.). La piccola scala rende anche più difficile per i sensoriMEMS essere isolati dalle vibrazioni che sono comuni in queste piattaforme di volo.Vista la difficoltà nella stabilizzazione e la sensibilità che devono avere gli strumentidi misura, per andare a semplificare a livello di calcolo e il processamento dati inquesto elaborato si è pensato ad uno strumento per velocizzarli. Inoltre nel lavoro si cerca di dare un nuovo metodo per il controllo non lineare applicato ai droni.

1.1 Storia del drone

Oggi parlare di droni e vedere volare degli oggetti senza pilota è una normalità, però è interessente vedere gli sviluppi dai primi droni fino al giorno d'oggi. Etienne Oehmichen è stato il primo scienziato a sperimentare progetti di aeromobili a rotore nel 1920 [1] (vedi Fig. 1.1). Tra i sei progetti che provò, il suo secondo multicottero aveva quattro rotori e otto eliche, tutti azionati da un unico motore. L'Oehmichen utilizzava un telaio in tubi d'acciaio, con rotori a due pale alle estremitá dei quattro bracci. L' angolo di queste pale poteva essere variato con la deformazione. Cinque delle eliche, girando sul piano orizzontale, stabilizzavano la macchina lateralmente. Un'altra elica era montata sul muso per il pilotaggio. La coppia di eliche rimanente era per la propulsione in avanti. L'aereo era molto stabile e anche controllabile, fece più di mille voli di prova durante la metà del 1920. Nel 1923 era in grado di rimanere

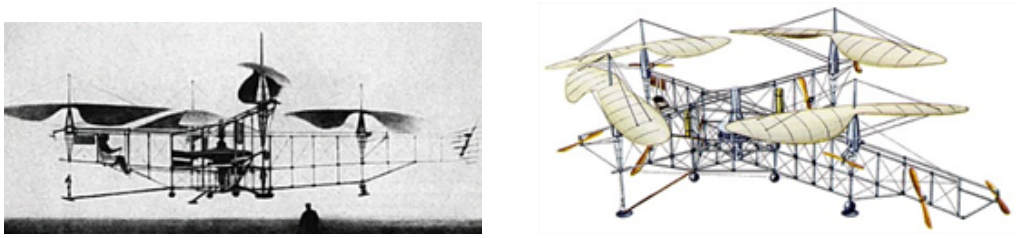

Figura 1.1: Aeromobili a rotore nel 1920. Fonte Google images

in volo per diversi minuti ed il 14 aprile 1924 stabilì il primo record di distanza per elicotteri della Fédération Aéronautique Internationale (FAI) [2] di 360 metri. Ha completato il primo volo a circuito chiuso di 1 chilometro da parte di un velivolo ad ala rotante.

Dopo Oehmichen, il Dr. George de Bothezat e Ivan Jerome svilupparono questo aereo [1], con sei rotori a pale all'estremità di una struttura a forma di X. Due piccole eliche a passo variabile sono state utilizzate per il controllo della spinta e dell'imbardata. Il veicolo utilizzava un controllo collettivo del passo. Fece il suo primo volo nell'ottobre 1922. Circa 100 voli sono stati fatti entro la fine del 1923. La massima altezza raggiunta fu di circa 5 m. Ad oggi, Amazon ha ricevuto il via libera per le consegne in America tramite un sistema di trasporto gestito dai droni. Amazon non è la prima realtà aziendale ad aver valutato l'idea delle consegne via drone, ma è l'unica che si è spinta a considerarlo un'alternativa davvero realizzabile. Oggi i droni sono sempre più intelligenti: nel 2016 Yuneec ha presentato Typhoon H, il primo APR semiprofessionale in versione Ready-To-Fly, equipaggiato con la tecnologia Intel RealSense per riconoscere gli ostacoli ed evitarli autonomamente.

1.1.1 Importanza del controllo

Numerosi metodi di controllo sono stati proposti per i quadrotori, sia per la regolazione che per l'inseguimento della traiettoria. L'obiettivo è trovare una strategia di controllo che permetta ad un quadrotor di convergere verso un insieme arbitrario di stati di riferimento variabili nel tempo stati di riferimento variabili nel tempo. Molti lavori precedenti [3, 4, 5, 6] hanno dimostrato che è possibile controllare il quadrotor usando tecniche di controllo lineare. Si va a linearizzare la dinamica in un punto di lavoro, di solito scelto per essere l'hover. Tuttavia, per avere più presione e prestazioni migliori si utilizzano tecniche di controllo non lineare che considerano una forma più generale della dinamica del veicolo in tutte le zone di volo. All'interno di questi metodi non lineari, backstepping [7, 8], sliding mode [9, 10] e linearizzazione di feedback [11] hanno dimostrato di essere efficaci per il controllo. Un lavoro recente [11] suggerisce una struttura di linearizzazione in retroazione che decostruisce la dinamica del quadrotor in un ciclo interno contenente l'atteggiamento e l'altezza del veicolo e un ciclo esterno che contiene la posizione. Un'altra linearizzazione a feedback permette di ottenere un modello lineare con la tecnica della Dynamic Feedback Linearization.

Tutte le tecniche di controllo suggerite sopra richiedono una conoscenza completa del modello del sistema e dei parametri del modello, ma gli errori nei valori identificati dei parametri possono portare a un significativo deterioramento delle prestazioni del controllore. Inoltre, variazioni non modellate nei parametri del sistema (come la massa o l'inerzia) durante il volo possono causare significativi errori di stabilizzazione. La necessità di un accurato modello non lineare della dinamica del drone può essere superato utilizzando metodi adattivi che possono reagire e correggere gli errori nelle stime dei parametri del modello del modello, modificare le stime dei parametri quando cambiano e anche adattarsi ai disturbi esterni. I metodi adattivi lineari come il Model Reference Adaptive Control (MRAC) [12]. Tuttavia, come per la maggior parte dei metodi lineari, la traiettoria ottenibile del drone è limitata a causa del presupposto della linearizzazione. Il lavoro di Huang [13] suggerisce un metodo backstepping adattivo, questo approccio è stato esteso per includere i parametri di inerzia nella legge di adattamento [14]. Alcuni lavori recenti hanno utilizzato i quadrotori usando anche dei metodi adattativi indiretti, come il metodo dei minimi quadrati (per la massa) proposto da Kumar et al. [15]. Tuttavia, tutti metodi indiretti correggono gli errori dei parametri in base alla differenza tra i risultati attesi e quelli effettivi dell'controllo, ma non correggono esplicitamente i parametri del modello (come fanno i metodi adattativi diretti). I metodi di adattamento diretto sono stati suggeriti per la prima volta da Craig [16] per i manipolatori meccanici. In questo lavoro si utilizzerà il Model Predictive Control (MPC) con associato un modello a dispersione SINDYc, per andare a ricreare la traiettoria del drone istante per istante, in modo che la triattoria non lineare su cui si basa il MPC sia sempre aggiornata e ricostruita rispetto all'evolversi delle condizioni in cui si trova il drone.

1.2 Modelli Gaussiani

I processi gaussiani sono una classe semplice e generale di modelli di funzioni. Per essere precisi, una GP è una qualsiasi distribuzione su funzioni tale che qualsiasi insieme finito di valori f(x1), f(x2), . . . f(xN) hanno una distribuzione gaussiana combintata. Un modello GP, prima del condizionamento sui dati, è completamente specificato dalla:

funzione media,

$$f(x) = \mu(x) \tag{1.1}$$

dalla funzione di covarianza,

$$Cov[f(x), f(x)'] = k(x, x') \tag{1.2}$$

È pratica comune assumere che la funzione media (1.1) sia semplicemente zero ovunque, poiché l'incertezza sulla funzione media può essere presa in considerazione aggiungendo un termine extra al kernel, ovvero la noise cioè l'incertezza. Dopo

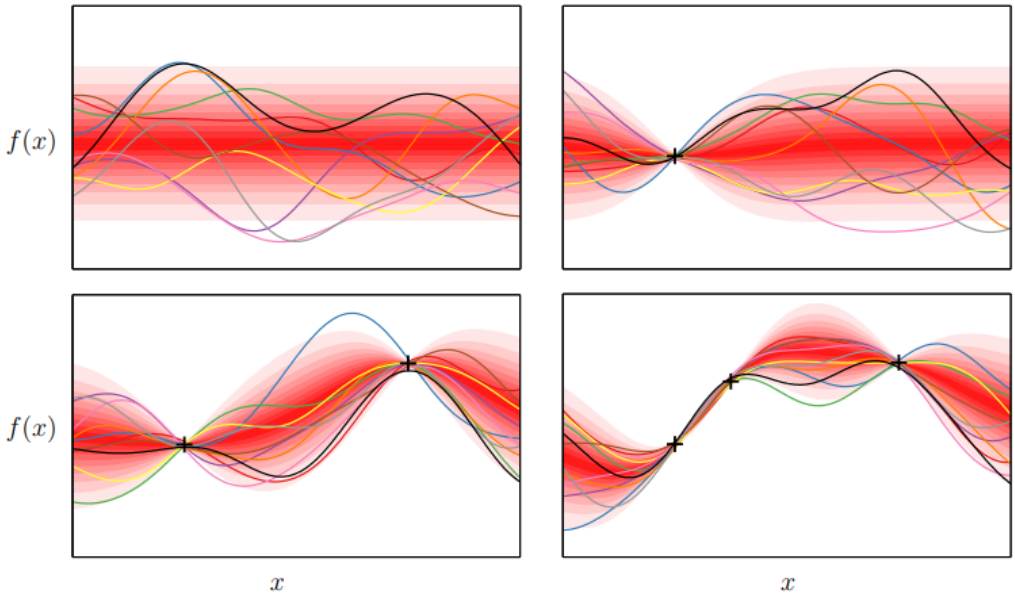

Figura 1.2: Rappresentazione visiva di un processo gaussiano che modella una funzione. Le linee colorate mostrano campioni del processo - esempi di alcune delle ipotesi incluse nel modello. In alto a sinistra: Un GP non ottimizzato su nessun datapoint.I restanti grafici: La regressione ottimmizata con diversi training points

aver tenuto conto della media, il tipo di struttura che può essere catturato da un modello GP è interamente determinato dal suo kernel (1.2). Il kernel determina come il modello si generalizza, o estrapola nuovi dati. Ci sono molte possibili scelte di funzione di covarianza, e possiamo specificare una vasta gamma di modelli semplicemente specificando il kernel di una GP. Per esempio, la regressione lineare, le spline e i filtri di Kalman sono tutti esempi di GP con kernel particolari. Tuttavia, questi sono solo alcuni esempi familiari di una vasta gamma di possibilità. Una delle principali difficoltà nell'uso delle GP è la costruzione di un kernel che rappresenti la particolare struttura presente nei dati da modellare. La Fig. 1.2 rappresenta una classica regressione variando il numero di training con un corretto kernel scelto.

1.3 Modelli a reti neurali

Le reti neurali, la cui implementazione non risultava pensabile nel breve termine, farebbero affermare a un Julius Verne redivivo che la fantascienza finalmente ha fatto il suo ingresso nella realtà. I circuiti neurali artificiali sono la base di sofisticate forme di intelligenza artificiale, sempre più evolute. Le reti neurali cercano di simulare i meccanismi del cervello umano, sono in grado di apprendere e migliorarsi un po' come l'intelligenza umana. Risultato: prestazioni impossibili per altri algoritmi. Le reti neurali artificiali riescono oggi a risolvere determinate categorie di problemi

avvicinandosi sempre più all'efficienza del nostro cervello, e trovando perfino soluzioni inaccessibili alla mente umana. Dalla nascita del concetto di neurone artificiale ad oggi è stata fatta molta strada. In moltissimi ed eterogenei settori scientifici, dalla biomedicina al data mining, le reti neurali hanno ormai un impiego quotidiano. Si tratta di un trend in crescita. I continui progressi permettono di ottenere circuiti sempre più sofisticati. Tutto lascia prevedere, insomma, che le reti neurali e il machine learning saranno parte notevole delle fondamenta del mondo ipertecnologico in cui ci stiamo addentrando.

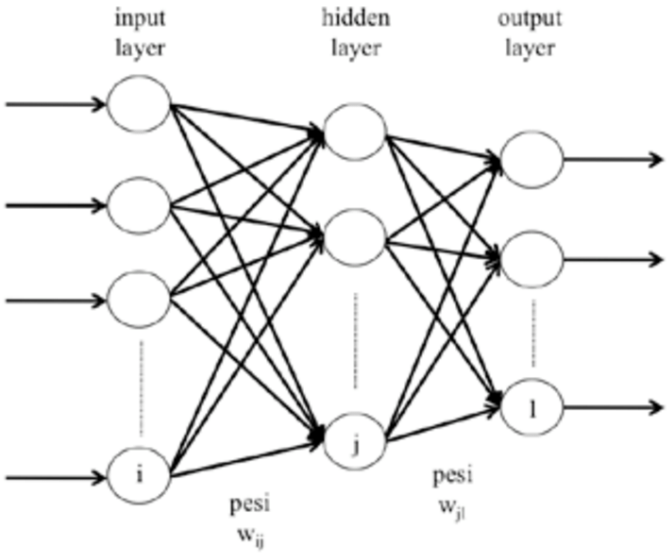

Figura 1.3: ANN: Artificial Neural Network

Il prototipo delle reti neurali artificiali (vedi Fig. 1.3) sono quelle biologiche. Le reti neurali del cervello umano sono la sede della nostra capacità di comprendere l'ambiente e i suoi mutamenti, e di fornire quindi risposte adattive calibrate sulle esigenze che si presentano. Sono costituite da insiemi di cellule nervose fittamente interconnesse fra loro. Al loro interno troviamo:

- i somi neuronali, ossia i corpi dei neuroni. Ricevono e processano le informazioni; in caso il potenziale di azione in ingresso superi un certo valore, generano a loro volta degli impulsi in grado di propagarsi nella rete;

- i neurotrasmettitori, composti biologici di diverse categorie (ammine, peptidi, aminoacidi), sintetizzati nei somi e responsabili della modulazione degli impulsi nervosi;

- gli assoni o neuriti: la via di comunicazione in uscita da un neurone. Ogni cellula nervosa ne possiede di norma soltanto uno;

- i dendriti: la principale via di comunicazione in ingresso; sono multipli per ogni neurone, formando il cosiddetto albero dendritico;

- le sinapsi, o giunzioni sinaptiche: i siti funzionali ad alta specializzazione nei quali avviene il passaggio delle informazioni fra neuroni. Ognuno di questi ne possiede migliaia. A seconda dell'azione esercitata dai neurotrasmettitori, le sinapsi hanno una funzione eccitatoria, facilitando la trasmissione dell'impulso nervoso, oppure inibitoria, tendente a smorzarlo. Le connessioni hanno luogo quando il neurotrasmettitore viene rilasciato nello spazio intersinaptico, raggiungendo così i recettori delle membrane post-sinaptiche (ossia del neurone successivo), e, alterandone la permeabilità, trasmettendo l'impulso nervoso.

Un singolo neurone può ricevere simultaneamente segnali da diverse sinapsi. Una sua capacità intrinseca è quella di misurare il potenziale elettrico di tali segnali in modo globale, stabilendo quindi se è stata raggiunta la soglia di attivazione per generare a sua volta un impulso nervoso. Tale proprietà è implementata anche nelle reti artificiali.

La configurazione sinaptica all'interno di ogni rete neurale biologica è dinamica. Si tratta di un fattore determinante per la loro efficienza. Il numero di sinapsi può incrementare o diminuire a seconda degli stimoli che riceve la rete. Più sono numerosi, maggiori connessioni sinaptiche vengono create, e viceversa. In questo modo, la risposta adattiva fornita dai circuiti neurali è più calibrata, e anche questa è una peculiarità implementata nelle reti neurali artificiali.

Capitolo 2

Modello matematico del drone

2.1 Sistemi di riferimento

Il quadrotor, un aereo composto da quattro motori, tiene la scheda elettronica al centro e i motori alle quattro estremità. Prima di descrivere il modello matematico di un quadrotor, è necessario introdurre le coordinate di riferimento in cui si descrive la struttura e la posizione. Per il quadrotore, è possibile utilizzare due sistemi di riferimento. Il primo è fisso e il secondo è mobile. Il sistema di coordinate fisso, detto anche inerziale, è un sistema in cui si considera valida la prima legge di Newton. Come sistema di coordinate fisso, usiamo i sistemi ONED, dove NED sta per Nord-Est-Giù. Come possiamo osservare dalla figura Fig. 2.1, i suoi vettori sono diretti a Nord, Est e al centro della Terra.

Il sistema di riferimento mobile che abbiamo menzionato in precedenza è unito al baricentro del quadrotor. Nella letteratura scientifica è chiamato OABC dove ABC sta per Aircraft Body Center. La figura Fig. 2.2 illustra il sistema di riferimento solidale al drone. L'atteggiamento e la posizione del quadrotore possono essere controllati ai valori desiderati cambiando le velocità dei quattro motori. Le seguenti forze e momenti possono essere eseguiti sul quadrotor: la spinta causata dalla rotazione dei rotori, il momento di beccheggio e il momento di rollio causato dalla differenza della spinta dei quattro rotori, la gravità, l'effetto giroscopico e il momento di imbardata.

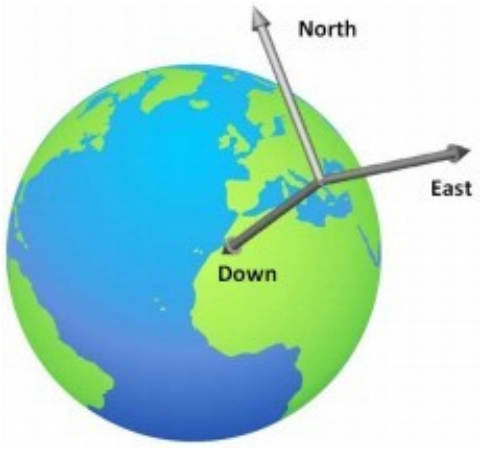

Figura 2.1: Sistema di riferimento fisso

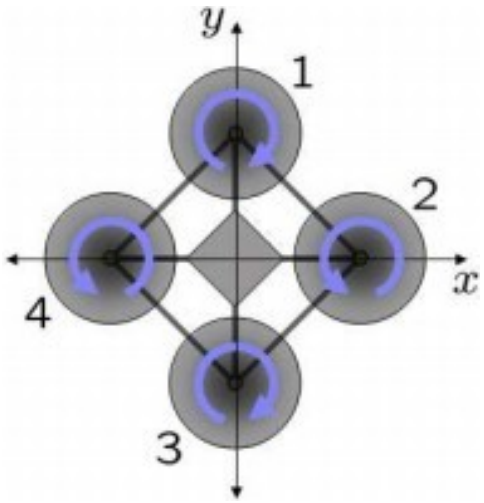

Figura 2.2: Sistema di riferimento solidale al drone

L'effetto giroscopico appare solo nel quadrotor di costruzione leggera. Il momento d'imbardata è causato dallo sbilanciamento delle velocità di rotazione dei quattro rotori. Il momento d'imbardata può essere annullato quando due rotori ruotano in direzione opposta. Quindi, le eliche sono divise in due gruppi. In ogni gruppo ci sono due motori diametralmente opposti che possiamo facilmente osservare grazie al loro senso di rotazione. Vale a dire, distinguiamo:

- eliche anteriori e posteriori (numeri 2 e 4 nella figura 2.2), che ruotano in senso antiorario;
- eliche destra e sinistra (numeri 1 e 3 nella figura 2.2), che ruotano in senso orario.

Il moto spaziale dell'aereo a corpo rigido può essere diviso in due parti: il movimento del baricentro e il movimento intorno al baricentro. Sei gradi di sono necessari per descrivere qualsiasi movimento nello spazio temporale. Sono tre movimenti del baricentro e tre movimenti angolari, cioè tre movimenti di traslazione e tre di rotazione lungo tre assi. Il controllo dei sei gradi di libertà può essere implementato regolando le velocità di rotazione dei diversi motori. I movimenti includono movimenti in avanti e indietro, movimento laterale, movimento verticale, movimento di rollio e movimenti di beccheggio e imbardata. Il movimento di imbardata del quadro può essere realizzato da una coppia reattiva prodotta dal rotore. La dimensione della coppia reattiva è relativa alla velocità del rotore. Quando le quattro velocità del rotore sono uguali, le coppie reattive si bilanciano a vicenda e il quadrotor non ruota, mentre se le quattro velocità del rotore non sono assolutamente uguali, le coppie reattive non si bilanciano e il quadrotor inizia a ruotare. Quando le quattro velocità del rotore aumentano e diminuiscono in modo sincrono è necessario anche nel movimento verticale. Il controllo come si diceva nel capitolo precedente è molto

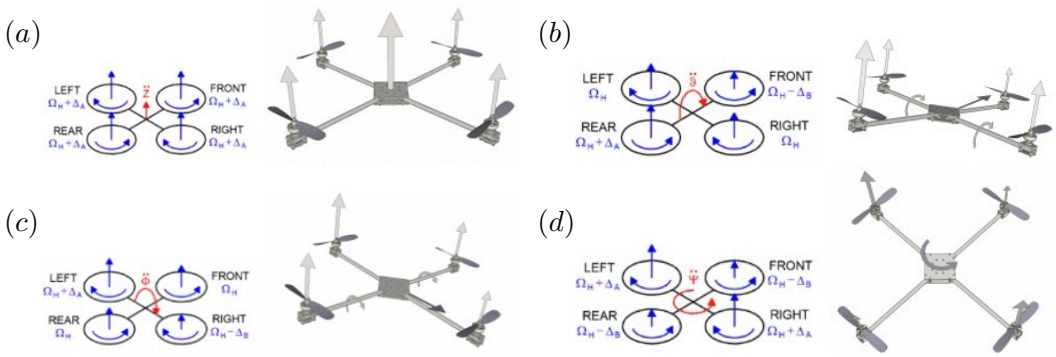

Figura 2.3: Movimenti del drone: a) Thrust, b) Pitch, c) Roll, d) Yaw.

difficolto, alcune ipotesi sono fatte nel processo di modellazione del quadrotor: il quadrotor è un corpo rigido, la struttura è simmetrica e l'effetto suolo è ignorato. A seconda della velocità di rotazione di ogni elica è possibile identificare i quattro movimenti di base del quadrotor, che sono mostrati nella figura 2.3.

Per andare a definire in modo corretto la dinamica del drone prima bisogna andare a definire gli angoli di riferimento. Gli angoli di Eulero sono tre angoli introdotti da Leonhard Euler per descrivere l'orientamento di un corpo rigido. Per descrivere tale orientamento nello spazio euclideo tridimensionale, sono necessari tre parametri. Possono essere dati in diversi modi; in questo elaborato si utilizzeranno gli angoli di Eulero ZYX [17]. Sono anche usati per descrivere l'orientamento di un quadro di riferimento rispetto ad un altro e trasformano le coordinate di un punto in un quadro di riferimento nelle coordinate dello stesso punto in un altro quadro di riferimento. Gli angoli di Eulero sono tipicamente indicati come $\phi \in [-\pi, \pi]$, $\theta \in [-\pi/2, \pi/2]$, $\psi \in [-\pi, \pi]$. Gli angoli di Eulero rappresentano una sequenza di tre rotazioni elementari, cioè rotazioni intorno agli assi di un sistema di coordinate, poiché qualsiasi orientamento può essere ottenuto componendo tre rotazioni elementari. La combinazione utilizzata è destritta dalle seguenti matrici [17]:

$$\mathbf{R_x}(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c(\phi) & -s(\phi) \\ 0 & s(\phi) & c(\phi) \end{bmatrix} \tag{2.1}$$

$$\mathbf{R_y}(\theta) = \begin{bmatrix} c(\theta) & 0 & s(\theta) \\ 0 & 1 & 0 \\ -s(\theta) & 0 & c(\theta) \end{bmatrix} \tag{2.2}$$

$$\mathbf{R_z}(\psi = \begin{bmatrix} c(\psi) & -s(\psi) & 0 \\ s(\psi) & c(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

dove $c(\phi) = cos(\phi), s(\phi) = sin(\phi), c(\theta) = cos(\theta), s(\theta) = sin(\theta), c(\psi) = cos(\psi), s(\psi) = sin(\psi)$. Quindi, le coordinate di posizione inerziale e le coordinate di riferimento

del corpo sono correlate dalla matrice di rotazione $\mathbf{R_{zyx}}(\phi, \theta, \psi) \in SO(3)$ Queste rotazioni partono da un orientamento standard noto come mostrato in Fig. (2.4).

$$\mathbf{R_{zyx}} = \mathbf{R_z}(\psi) \cdot \mathbf{R_y}(\theta) \cdot \mathbf{R_x}(\phi) \tag{2.4}$$

L'equazione (2.4) descrive il sistema di riferimento del drone rispetto al sistema di riferimento inerziale.

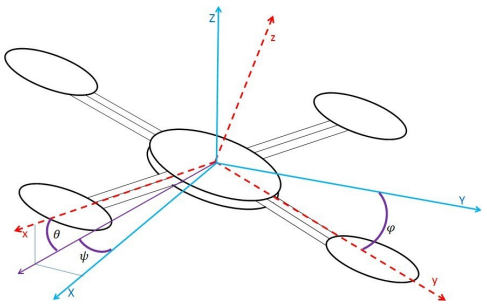

Figura 2.4: Angoli di eulero

2.2 Modellazione matematica

Forniamo qui un modello matematico del quadrotore, sfruttando le equazioni di Newton ed Eulero per il moto 3D di un corpo rigido. L'obiettivo di questa sezione è quello di ottenere una comprensione più profonda della dinamica del quadrotor e di fornire un modello che sia sufficientemente affidabile per simulare e controllare il suo comportamento. Chiamiamo $[x \quad y \quad z \quad \phi \quad \theta \quad \psi]^T$ il vettore contenente la posizione lineare e angolare posizione del quadrotor e $[u \quad v \quad w \quad p \quad q \quad r]^T$ il vettore contenente le velocità lineari e angolari nel telaio del corpo. Dalla dinamica del corpo 3D, segue che i due telai di riferimento sono collegati dalle seguenti relazioni:

$$v = R \cdot v_B \tag{2.5}$$

$$\omega = T \cdot \omega_B \tag{2.6}$$

dove $v = [\dot{x} \quad \dot{y} \quad \dot{z}]^T \in \mathcal{R}^3, \omega = [\dot{\phi} \quad \dot{\theta} \quad \dot{\psi}]^T \in \mathcal{R}^3, v_B = [u \quad v \quad w]^T \in \mathcal{R}^3, \omega = [p \quad q \quad r]^T \in \mathcal{R}^3$ and $\mathbf{T}$ è la matrice di trasformazione angolare (3.1)

$$\mathbf{T} = \begin{bmatrix} 1 & s(\phi)t(\theta) & c(\phi)t(\theta) \\ 0 & c(\phi) & -s(\phi) \\ 0 & \frac{s(\phi)}{c(\theta)} & \frac{c(\phi)}{c(\theta)} \end{bmatrix} \tag{2.7}$$

dove $t(\theta) = tan(\theta), s(\theta) = sen(\theta), c(\theta) = cos(\theta)$. Di seguito si riporta il vettore $\mathbf{x}$

il quale descrive lo stato del sistema. (2.8)

$$\mathbf{x} = [\phi \quad \theta \quad \psi \quad p \quad q \quad r \quad u \quad v \quad w \quad x \quad y \quad z]^T \in \mathcal{R}^{12} \tag{2.8}$$

in cui si riporta per esteso la descrizione delle diverse variabili di stato in tab. 2.1

$$\begin{cases} \dot{\phi} = p + r[c(\phi)t(\theta)] + q[s(\phi)t(\theta)] \\ \dot{\theta} = q[c(\phi)] - r[s(\phi)] \\ \dot{\psi} = r\dfrac{c(\phi)}{c(\theta)} + q\dfrac{s(\phi)}{c(\theta)} \\ \dot{p} = \dfrac{I_y - I_z}{I_x}rq + \dfrac{\tau_x - \tau_{wx}}{I_x} \\ \dot{q} = \dfrac{I_z - I_x}{I_y}pr + \dfrac{\tau_y - \tau_{wy}}{I_y} \\ \dot{r} = \dfrac{I_x - I_y}{I_z}pq + \dfrac{\tau_z - \tau_{wz}}{I_z} \\ \dot{u} = rv - qw - g[s(\theta)] + \dfrac{f_{wx}}{m} \\ \dot{v} = pw - ru + g[s(\phi)c(\theta)] + \dfrac{f_{wy}}{m} \\ \dot{w} = qu - pv + g[c(\theta)c(\phi)] + \dfrac{f_{wz} - f_t}{m} \\ \dot{x} = w[s(\phi)s(\psi) + c(\phi)c(\psi)s(\theta)] - v[c(\phi)s(\psi) - c(\psi)s(\phi)s(\theta)] + u[c(\psi)c(\theta)] \\ \dot{y} = v[c(\phi)c(\psi) + s(\phi)s(\psi)s(\theta)] - w[c(\psi)s(\phi) - c(\phi)s(\psi)s(\theta)] + u[c(\theta)s(\psi)] \\ \dot{z} = w[c(\phi)c(\theta)] - u[s(\theta)] + v[c(\theta)s(\phi)] \end{cases} \tag{2.9}$$

In cui con c, s e t descrivono rispettivamente le funzioni coseno, seno e tangente. Mentre le τ_{wx}, τ_{wy}, τ_{wz} e f_{wx}, f_{wy}, f_{wz} descrivono le forze e le coppie areodinamiche, agenti sulla struttura del drone, dei quali valori sono stati ricavati da letteratura per poter lavorari con valori verosimili delle condizioni di impiego del quadrotor.

Di seguito le formule descrittive delle τ_x, τ_y, τ_z e f_t le quali rappresentano rispettivamente le coppie agenti sulla struttura del disposito rispettivamente lungo gli assi x, y, z e la forza generata dai rotori.

$$\begin{cases} f_t = b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \\ \tau_x = bl(\Omega_3^2 - \Omega_1^2) \\ \tau_y = bl(\Omega_4^2 - \Omega_2^2) \\ \tau_z = bl(\Omega_2^2 + \Omega_4^2 - \Omega_1^2 - \Omega_3^2) \end{cases} \tag{2.10}$$

Mentre le forze d'inerzia agenti sul sistema sono:

$$\begin{cases} f_x = m(\dot{u} + qw - rv) \\ f_y = m(\dot{v} + pw - ru) \\ f_z = m(\dot{u} + pv - qu) \end{cases} \tag{2.11}$$

Variables	Symbol	Unit
roll	ϕ	rad
pitch	θ	rad
yaw	ψ	rad
velocità di roll	p	rad/s
velocità di pitch	q	rad/s
velocità di yaw	r	rad/s
velocità lungo x	u	m/s
velocità lungo y	v	m/s
velocità lungo z	w	m/s
posizione lungo x	x	m
posizione lungo y	y	m
posizione lungo z	z	m

Tabella 2.1: Parametri utilizzati in Eq. (2.9)

Capitolo 3

Regressione della dinamica del drone con una variabile di controllo

3.1 Processo di regressione Gaussiana del modello del drone

3.1.1 Selezione del modello

In questo capitolo affronteremo la predizione della traiettorie dell'oggetto in esame, ovvero il drone andando ad utilizzare tecniche di regressione gaussiana. Il modello che si va ad utilizzare è il modello classico gaussiano con una covariance function, che chiameremo kernel e una mean function. È pratica comune assumere che la funzione media sia semplicemente zero ovunque, poiché l'incertezza sulla funzione media può essere presa in considerazione aggiungendo un termine extra al kernel. Dopo aver tenuto conto della media, il tipo di struttura che può essere catturato da un modello di Processo Gaussiano (GP) è interamente determinato dal suo kernel. Il kernel determina come il modello si generalizza, o estrapola a nuovi dati. Ci sono molte scelte possibili di funzione di covarianza, e possiamo specificare una vasta gamma di modelli semplicemente specificando il kernel di un GP. Per esempio, la regressione lineare, le spline e i filtri di Kalman sono tutti esempi di GP con kernel particolari. Tuttavia, questi sono solo alcuni esempi familiari di una vasta gamma di possibilità. Una delle principali difficoltà nell'uso delle GP è la costruzione di un kernel che rappresenti la particolare struttura presente nei dati da modellare.

La proprietà cruciale delle GP che ci permette di costruire automaticamente i modelli è che possiamo calcolare l marginal likelihood di un set di dati dato un particolare modello, come evidenza (MacKay, 1992). Il marginal likelihood permette di confrontare modelli, bilanciando la capacità di un modello e il suo adattamento ai dati (MacKay, 2003; Rasmussen e Ghahramani, 2001). Il marginal likelihood di un sistema generico di N funzioni $[f(x_1), f(x_2), ...f(x_N)] := f(X)$ nelle posizioni X è dato da:

$$p(f(X)|X, \mu(.), k(.,.)) = \mathcal{N}(f(X)|\mu(X), k(X,X)) =$$

$$= (2\pi)^{\frac{N}{2}} \times \|k(X,X)\|^{-\frac{1}{2}} \times exp\{-\tfrac{1}{2}(f(X) - \mu(X))^T k(X,X)^{-1}(F(X) - \mu(X))\} \tag{3.1}$$

Questa densità gaussiana multivariata è chiamata probabilità marginale perché implicitamente integra su tutti i possibili valori della funzione f($\bar{X}$), dove $\bar{X}$ è l'insieme di tutti i luoghi in cui non abbiamo osservato la funzione.

Viste le considerazioni fatte precedentemente si può andare a chiedere al modello quali valori hanno probabilità di verificarsi e in qualsiasi luogo. Campionare una funzione da una GP è anche semplice: un campione da una GP in un insieme finito di posizioni è solo un singolo campione da una singola distribuzione gaussiana multivariata, data dall'equazione . Il nostro uso delle probabilità non significa che stiamo assumendo che la funzione appresa sia stocastica o casuale in alcun modo; è semplicemente un metodo coerente per tenere traccia dell'incertezza.

3.2 Strutture dei kernel

Un kernel (chiamato anche funzione di covarianza, funzione kernel o kernel di covarianza), è una funzione positivo-definita di due input x, x'. Viene descritta con questo linguaggio:

$$Cov[f(x), f(x)'] = k(x, x') \tag{3.2}$$

il kernel codifica il modo in cui i valori di training e di test siano correlati tra loro. Di funzioni di covarianza ce ne sono molteplici anzi uno si può creare la funzione kernel che preferisce. La scelta della matrice di covarianza è fondamentale per ottenere un buon adattamento del modello. Per iniziare a capire i tipi di strutture esprimibili dalle GP, cominceremo con brevemente esaminando i priori sulle funzioni codificate da alcuni kernel comunemente usati: il quadrato-esponenziale (SE), periodico (Per) e lineare (Lin). Questi kernel sono definiti in Fig. (3.1)

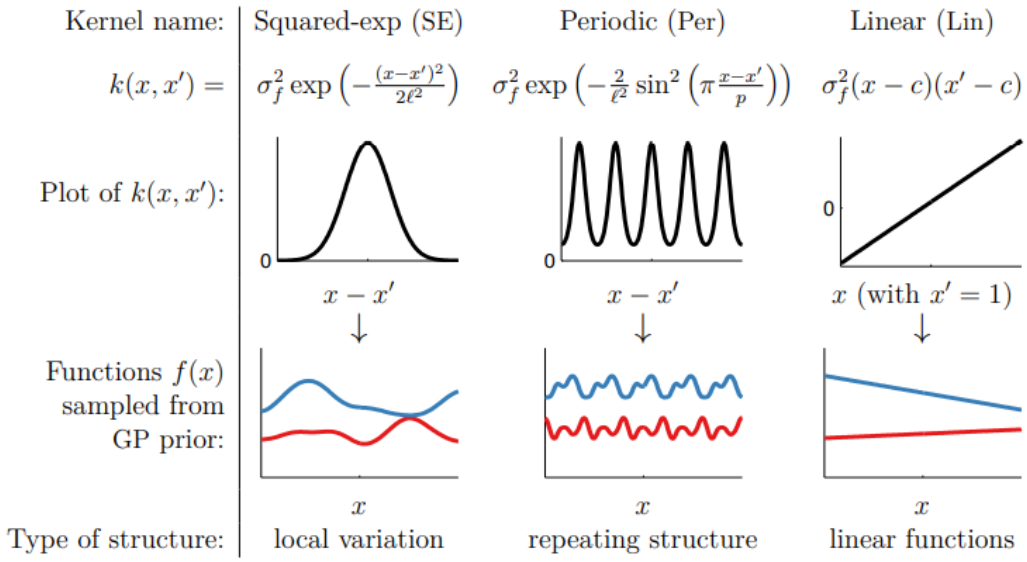

Figura 3.1: Esempi di strutture esprimibili da alcuni kernel di base

Ogni funzione di covarianza corrisponde a un diverso insieme di ipotesi fatte sulla la funzione che vogliamo modellare. Per esempio, l'uso di un kernel quadrato-exp (SE) implica che la funzione che stiamo modellando ha un numero infinito di derivate. Esistono molte varianti di kernel "locali" simili al kernel SE, ognuno dei quali codifica ipotesi leggermente diverse sulla morbidezza della funzione da modellare.

I kernel SE e (Per) sono stazionari, cioè che il loro valore dipende solo dalla differenza $x - x'$. Questo implica che la probabilità di osservare un particolare set di dati rimane la stessa anche se spostiamo tutti i valori x di la stessa quantità. Al contrario, il kernel lineare (Lin) non è stazionario, il che significa che il modello GP corrispondente produrrà previsioni diverse se i dati vengono spostati mentre i parametri del kernel vengono mantenuti fissi.

3.2.1 GPr con libreria standard

Andremo ad analizzare l'utilizzo delle regressione gaussiane (GPr) relativo al caso studio scelto, ovvero il drone. Prima di andare a fare una predizione utilizzando le regressione gaussiane o le reti neurali, bisogna fare acquisizione dati. Nel problema in questione non abbiamo fatto analisi di dati sperimentali, ma prendendo dati da simulazione analitiche. Ovvero andando ad integrare le dodici equazioni differenziali che caratterizzano la traiettoria del drone ((2.9)). Integrando queste equazioni in un determinato lasso di tempo si ottengono tutti i valori di posizione e velocità che ha il drone in base ad una determinata condizione iniziale. Le condizioni iniziale variano in base a come si vuole far partire il drone e anche in base a come sono scritte le equazioni e il sistema di riferimento di rotazione dei motori che viene preso. Per salvare i dati è stato fatto un modello di salvataggio iterativo che per ogni step del ciclo for aggiungeva una riga con tutti i valori delle equazioni del drone, ma anche per avere tutti i valori che caratterizzano quella particolare conidizione iniziale e i parametri che la governano, come per esempio la rotazione del motore, oppure l'attrito dovuto al vento in una direzione. Aver generato questa matrice di dati ci da la possibilità di andare a fare predizione utilizzando la tecniche prima citate. Quando si va a fare predizione ho dei valori di training e altri di test. Con la matrice di valori reali di posizione e velocità del drone vado a fare il training, ovvero prendo alcuni valori di quella matrice per esempio ne posso prendere 10 e vado a farci una regressione, ovviamente non sarò preciso come se facessi tutti test sperimentali o tutte le simulazione per ogni condizione di varazione dei parametri. Però a livello computazionale ho un risparmio enorme di dati. Quindi andremo ad utilizzare solo alcuni valori reali di traiettoria del drone che andremo ad interpolare utilizzando le regressioni gaussiane oppure le reti neurali. Il primo step che si è affrontato è andare un valore di riferimento scelto del drone, per esempio l'asse z e vedere come varia al variare di un parametro come la velocità del vento lungo la direzione x. Quindi la prima cosa che si è fatta e vedere come la variazione di un parametro varia la coordinata di riferimento scelta, ovviamente come già accennato prima noi prendiamo

solo dei punti reali gli altri sono tutti ricavati con le tecniche sopra citate. Questa parte del lavoro come si accennava all'inizio è stata svolta su un jupyter notebook, utilizzando come linguaggio di programmazione python, per andare a fare regressione, in particolare utilizzando il modello gaussiano, il modo più semplice è importare delle librerie già fatte, come per esempio scikit-learn o libreria GPy. All'inizio le prove di training si sono svolte in questo modo, ovvero andando ad utilizzare queste due librerie. La libreria scikit-learn è quella più tradizionale in cui non ho libera scelta di plottaggio o anche di andare a personalizzare i vari plot, mentre GPy è più flessibile.

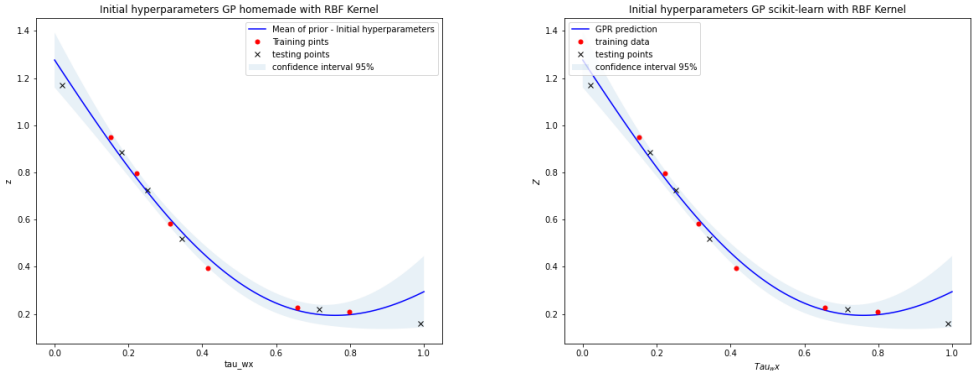

Figura 3.2: GPr non ottimizzata. Sinistra: GPr con codice homemade. Destra: GPr con codice scikit-learn

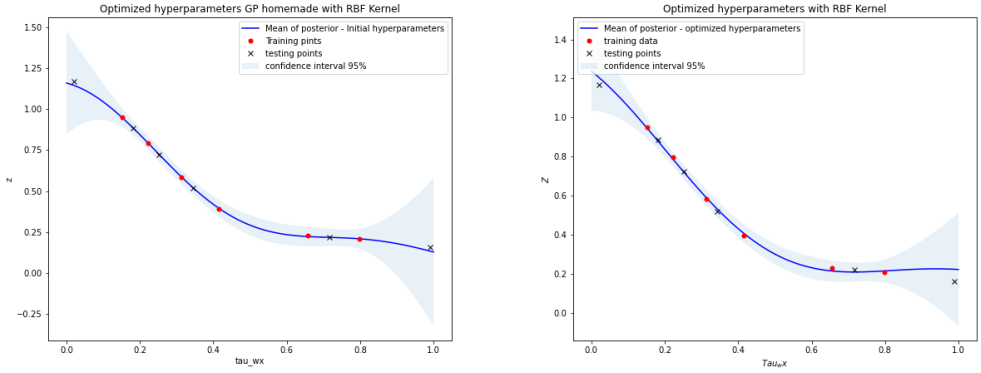

Figura 3.3: GPr ottimizzata. Sinistra: GPr con codice homemade. Destra: GPr con codice scikit-learn

In Fig. 3.2 si può vedere come la regressione non riesce a seguire totalemente i punti di training, perché i valori degli iperparametri non sono ottimizzati. Nel kernel utili in figura La scelta di andare a sostituire una libreria con un codice scritto da soli è stato scelto per avere più flessibilità oltre che per andare a migliorare le conoscenze nell'ambito della programmazione. Avere più flessibilità significa che il codice è

facilemente modificabile o variabile, per esempio il kernel. Nelle librerie ci sono kernel già predisposti e già definiti, infatti usare quelli standard è molto più facile, però nel caso si volessero andare ad utilizzare kernel non troppo standard la cosa diventa più difficile e dispendiosa a livello di tempo, perché bisogna andare a definire funzione esterne alla libreria ed importarle. Lo stesso vale per le ottimizzazioni, di default c'è il marginal likelihood, per andarene ad utilizzare altri bisogna importarli dall'esterno. Per ricapitalore è vero che il sistema è più dispendosio a livello di tempo, ma ti garantisce molta più flessibilità

3.2.2 MSE: confronto

Uno step fondamentale per andare a vedere se i valori calcolati nella fase di training abbiano senso è andare a confrontarli con dei valori di testing. I punti di riferimento presi come test sono sempre gli stessi e si va a fare il MSE di quei valori di test presi come riferimento e i punti di training che invece sono sempre diversi per ogni iterazione del ciclo. Così facendo in base al numero di campionamenti che scelgo di fare ho sempre lo stesso set di dati di riferimento. Utilizzare un set di dati di riferimento giusto per il problema che si va ad affronate è fondamentale perché ci fa capire se la regressione che abbiamo fatto abbia senso oppure no. Adesso andiamo un po' in dettaglio su cosa sia l'MSE ed il codice scelto. L'errore quadratico medio MSE mi va ad indicare la differenza quadratica media che c'è tra i valori di test e quelli di training.

$$MSE = \frac{1}{N}\sum_{i=1}^{n}(X_i - \hat{X}_i)^2 \tag{3.3}$$

l'equazione (3.1) ci fa capire quello che fa l'mse ovvero va a generare un valore per tutti i punti che io vado a considerare. Però per avere un confronto più statistico e che abbia più valore scientifico vado a fare più iterazioni. Quello che si ottiene è sostanzialmente questo:

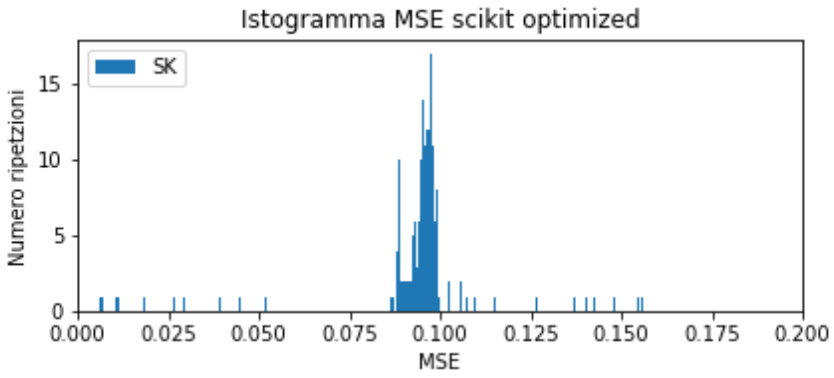

Figura 3.4: MSE calcolato attraverso i dati estrapolati con la libreria scikit

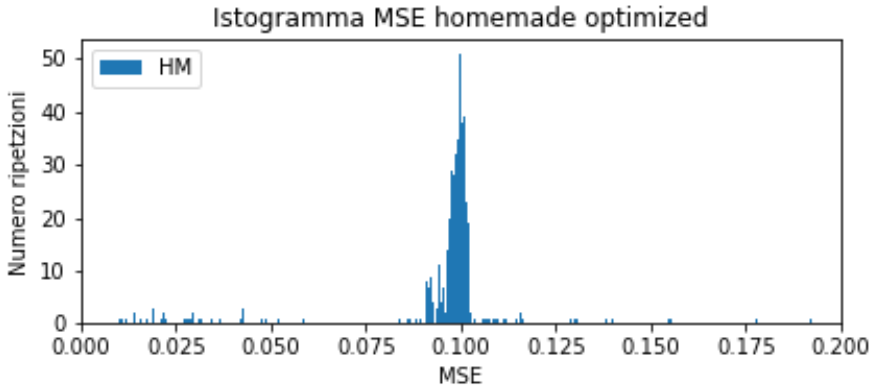

Figura 3.5: MSE calcolato attravero i dati estrapolati con il codice personalizzato

In Fig. 3.4 e in Fig. 3.5 i risultati ottenuti tra le due configurazioni sono molto simili tra loro, questo va a validare il codice scritto e quindi possibile da utilizzare anche per ampliare lo studio delle varibili da considare, infatti lo step successivo che si è fatto è andare a considerare non più un solo parametro di riferimento , ma andarne a considerare due.

3.3 Rete neurale

Le reti neurali nella loro formulazione non bayesiana: l'algoritmo ML per eccellenza, basato su milioni di combinazioni di semplici funzioni di trasferimento che collegano neuroni (nodi) che dipendono dai pesi. L'allenamento consiste nel trovare questa grande quantità di pesi in modo che si adattino di conseguenza ai dati. Le reti neurali sono algoritmi molto semplici, quindi estremamente scalabili (possono gestire facilmente milioni di punti dati). Tuttavia, come si può immaginare, l'allenamento non è un processo banale a causa delle schiaccianti possibilità di trovare i pesi di ogni connessione. Anche i layer hanno un importanza nel raggiungimento dell'obiettivo finale, cioè nell'andare ad ottimizzare la traiettoria scelta.

Come si può notare dalla figura nell'asse X abbiamo la variabile del vento, mentre nelle Y la variabile osservata che in questo caso è l'altezza del drone, la rete neurale non riesci a comportarsi bene a livello di seguire i punti di training, che sono anche gli stessi per il testing, il che certifica l'importanza di avere più neurons nei layer.

In questa figura si può già vedere subito una differenza sostanziale rispetto a quella precedente. Considerando che i dati sono gli stessi (cioè i punti) e che nell'ascisse e ordinate sono le stesse, ovvero le condizioni al controno sono le medesime. Le

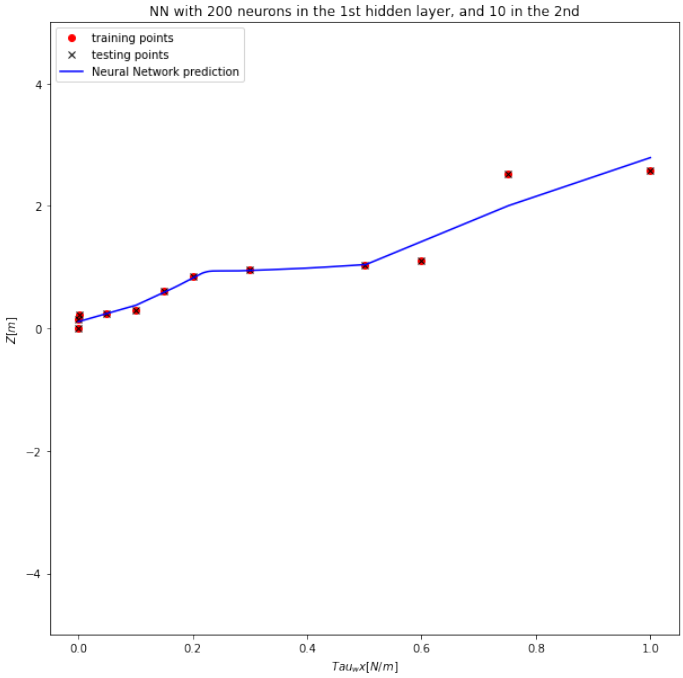

Figura 3.6: Neural Network con 10 neuroni

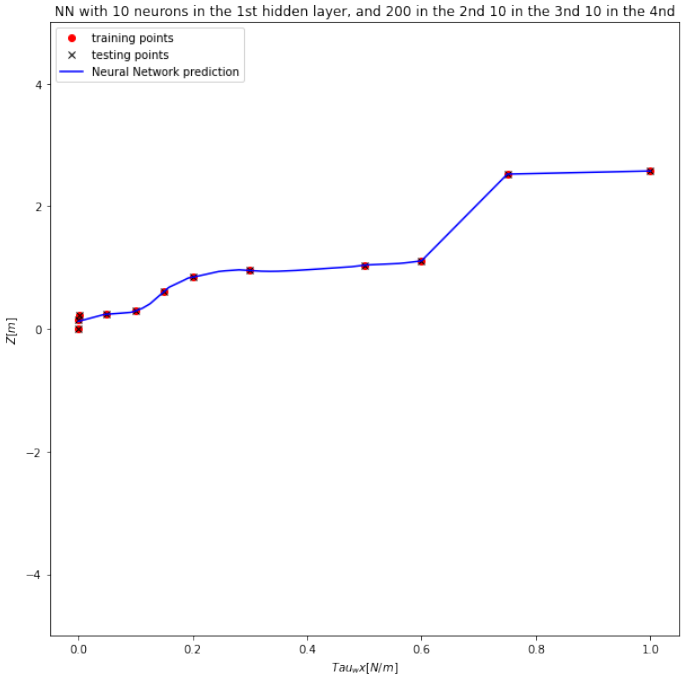

Figura 3.7: MSE calcolato attraverso i dati estrapolati con la libreria scikit

due regressioni però hanno esiti molto diversi, ovvero questa è molto più precisa e segue molto di più l'andamento dei dati utilizzati. La differenza sostanziale tra un grafico e l'altro sono il numero di layer e il numero di neurons, in questa particolare geometria abbiamo 4 layer, nel primo e nel terzo ci sono 10 neurons, mentre nel secondo 200, ovviamente aumentare i layer e i neurons a livello computanzionale è peggiorativo, quindi bisogna sempre cercare una via ottimale a livello di calcolo e di risultati ottenuti.

Capitolo 4

Regressione della dinamica del drone con due variabili di controllo

4.1 Processo Gaussiano con 2 variabili

Fino ad ora abbiamo considerato che una variabile caratteristica del drone, posizione o velocità, sia dipendente solo da una variabile esterna. Infatti si è considerato sempre la variabile di altezza del drone, ovvero la Z, dipendente dalla velocità del vento lungo una direzione, la τ_{wx}. Però in realtà la posizione del drone non è dipende solo da una variabile, ma da tutte quelle esterne che influenzano il drone, come per esempio la velocità del vento lungo tutte le direzioni, ma anche le forze che vanno ad agire sui motori sono forze che influenzano una sola variabile. Nella realtà la posizione Z del drone è infleunzata da ben 6 parametri, quindi andare a fare una regressione con un valore di input associato ad un valore di output è riduttivo, si dovrebbero associare 6 valori di input e uno di output. In questo capitolo si vanno a studiare l'influenza di due parametri. Prima di andare a vedere codice e risultati, bisogna andare a fare una precisazione sui kernel utilizzati. Per i sistemi a multi-input, la matrice di covarianza è diversa e ci sono varie possibilità per andare a generarla, soprattuto grazie alle proprietà che hanno i kernel. I kernel hanno la proprietà additiva

$$k_a + k_b = k_a(x, x^i) + k_b(x, x^j) \tag{4.1}$$

$$k_a + k_b = k_a(x, x^i) + k_b(x, x^j) \tag{4.2}$$

Tutti i kernel di base che abbiamo considerato nelle equazioni (4.1) e (4.2) sono unidimensionali, ma i kernel su input multidimensionali possono essere costruiti aggiungendo e moltiplicando tra kernel su dimensioni diverse. La dimensione su cui opera un kernel è denotata da un intero con pedice. Per esempio, RBF_2 rappresenta un kernel RBF sulla seconda dimensione del vettore x.

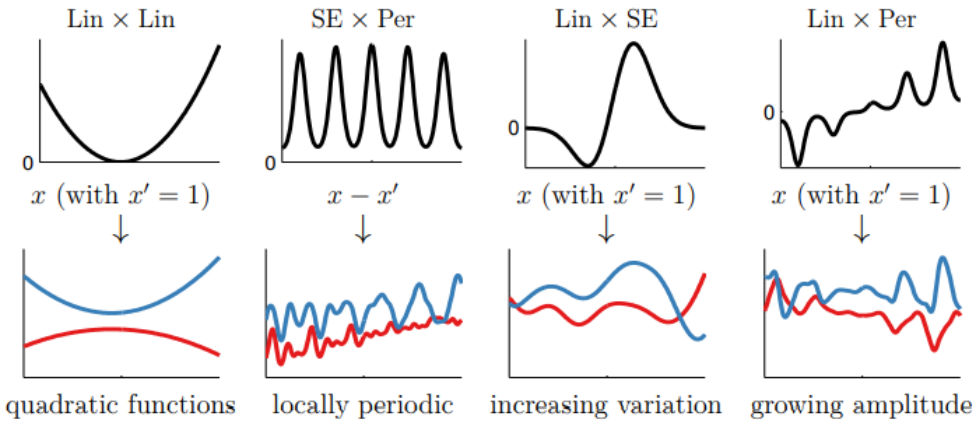

Figura 4.1: strutture di funzioni uni-dimensionali espresse moltiplicando i kernel [2]

In Fig. 4.1 viene fatta vedere la differenza di di trasferimento utilizzando un singolo kernel oppure dei kernel multipli. Ogni kernel ha un certo numero di parametri che specificano la forma precisa della funzione di covarianza. Questi sono a volte indicati come iper-parametri. Nel kernel SE per esempio la funzione caratterizzante è la lenghtscale che è definito con la ℓ, oppure la σ per il kernel RBF. Avendo questa proprietà posso andare a sommare gli effetti di due o più parametri per andare a vedere il comportamento di una sola variabile di output. Quando si modellano funzioni di dimensioni multiple, la somma dei kernel può dare origine a una struttura aggiuntiva attraverso diverse dimensioni.

$$f(x_1, x_2) \sim \mathcal{GP}(0, k_1(x_1, x_1') + k_2(x_2, x_2')) \tag{4.3}$$

l' equazione (4.3) è l'equazione generale che inidica la relazione tra mean function e funzione di covarianza con l'output di uscita.

in Fig 4.2 si può vedere come la sotto figura (c) è la somma di due kernel monodimensionali, mentre la figura (f) è l'effetto dei due kernel nelle due funzioni che genera il singolo output in uscita. In altre parole nel caso studio sono l'effetto per esempio delle due velocità del vento in una posizione del drone per esempio l'altezza.

$$\begin{aligned} f_1(x_1) &\sim \mathcal{GP}(0, k_1(x_1, x_1')) \\ f_2(x_2) &\sim \mathcal{GP}(0, k_2(x_2, x_2')) \\ f(x_1, x_2) &= f_1(x_1) + f_2(x_2). \end{aligned} \tag{4.4}$$

L'Eq. (4.4) mostra la decomposizione delle proprietà additive dei processi gaussiani

4.2 Regressione bidimensionale

Entriamo nel dettagio dell'applicazione svolta all'interno dell'elaborato. In questa sezione la challenge era estendere le regressioni gaussiane dallo studio mono-

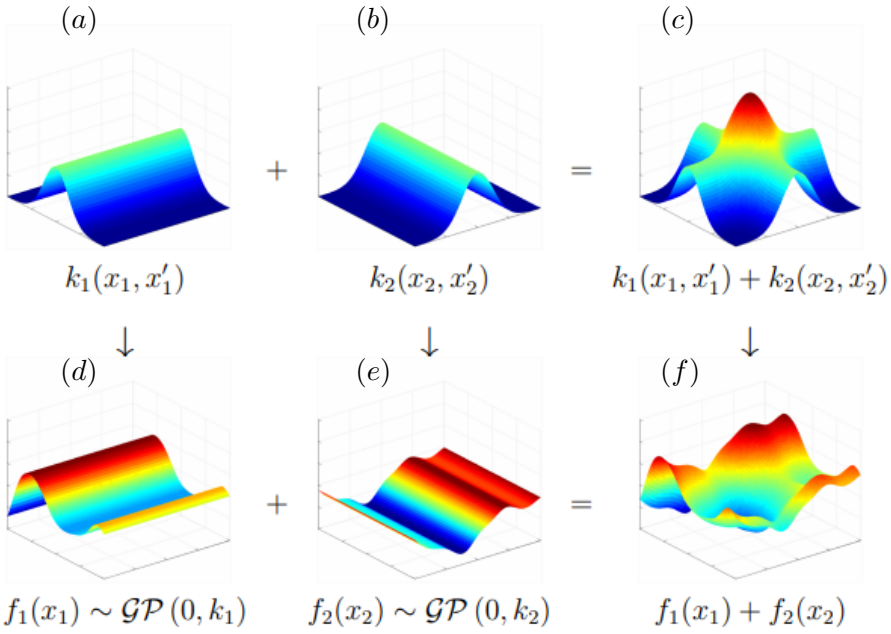

Figura 4.2: Somma di due kernel monodimensionali ortogonali

dimendionale, un valore di input e uno di output, a quello bi-dimensionale. Anche a livello bi-dimensionale si è andati avanti sempre con lo stesso apporccio. Si sono utilizzate parallelamente le librerie e si è sviluppata una nuova versione del codice,adatta a questo problema. Come primo step si è creata la matrice di punti di training utilizzanta per andare a fare la regressione gaussiana. Però il meccanismo per andare a generare l'output in uscita è diverso. La mole di simulazioni per generare la superfice che descrive la variabile di interessa al variare di più parametri, non scala linearmente con la dimensionalità del problema. Nel nostro caso si condisderano due parametri le quali sono la velocità del vento lungo la direzione x e lungo la direzione y, mentre la variabile di interesse è l'altezza del drone. Nel caso bi-dimensionale la difficoltà è regredire l'andamento di una variabile andando a considerare gli effetti di due parametri contemporanemante. Al fine di confrontare i due modelli e validarli sono state volte simulazioni con diverse quantità di training points.

La Fig. 4.3 rappresenta l'andamento reale dell'altezza del drone andando a variare i due parametri considerati.

In Fig. 4.4 e Fig. 4.5 si può vedere l'andamento della traiettoria del drone ottenuto con l'utilizzo di 10 training points. L'immagine a sinistra è stata ottenutta andando ad utilizzare il codice che si è sviluppato, mentre quella a destra utilizzando quello della libreria scikit-learn. Entrambi i risultati sono già stati ottimizzati utilizzando lo stesso metodo del caso monodimiensionale ovvero il log marginal likelihood. Dalle figure si denota che il codice che si è scritto segue di meno la curva reale, ma riesce a generare una curva più smooth. Inoltre il codice che si è scritto riesce ad identificare l'andamento della variabile, questo indica che l'algoritmo sviluppato coglie la natura del modello affrontato.

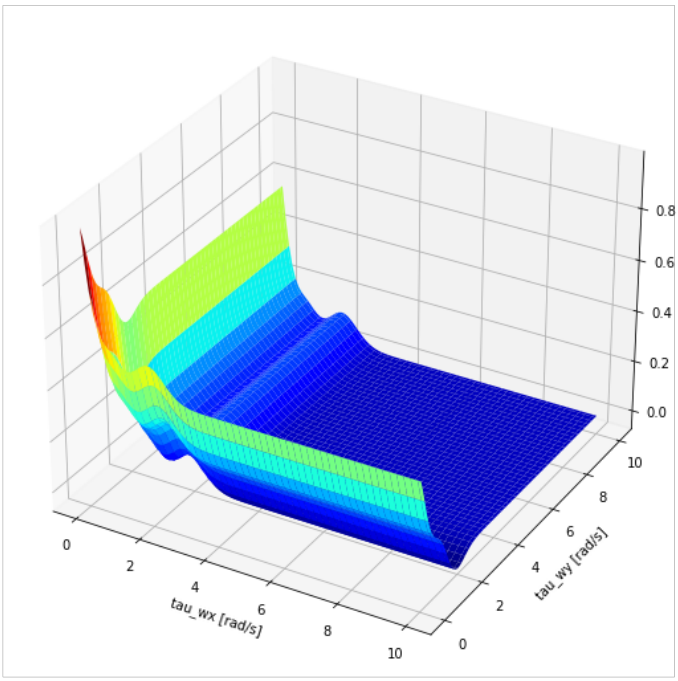

Figura 4.3: Andamento reale del drone

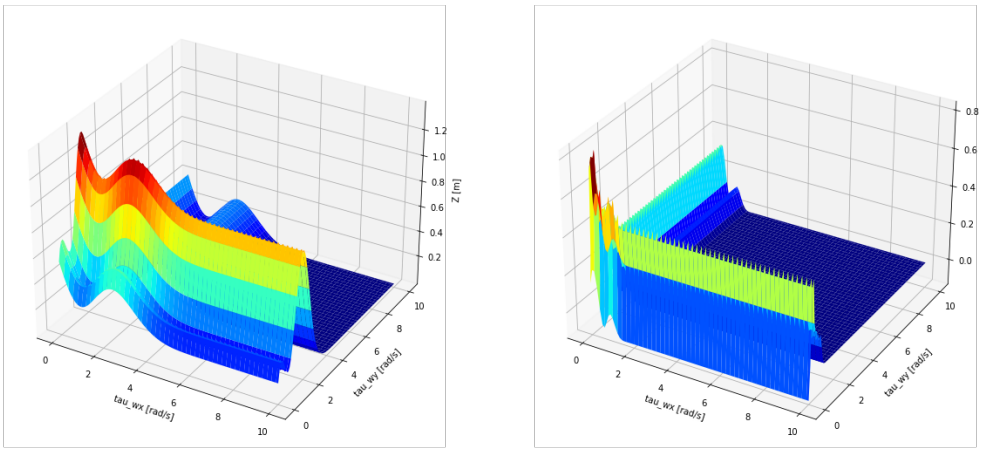

Figura 4.4: Ricostruzione della variabile con 10 training points. Sinistra: homemade; Destra: scikit-learn

Come si può vedere in Fig. 4.6 utilizzando un certo numero di training points si riesce a cogliere in modo quasi reale l'andamento della variabile. A sinistra la curva ottenuta con la regressione, mentre a destra quella reale. L'andamento dell'altezza del drone si ha lungo l'asse z, mentre le velocità del vento lungo due direzione sono

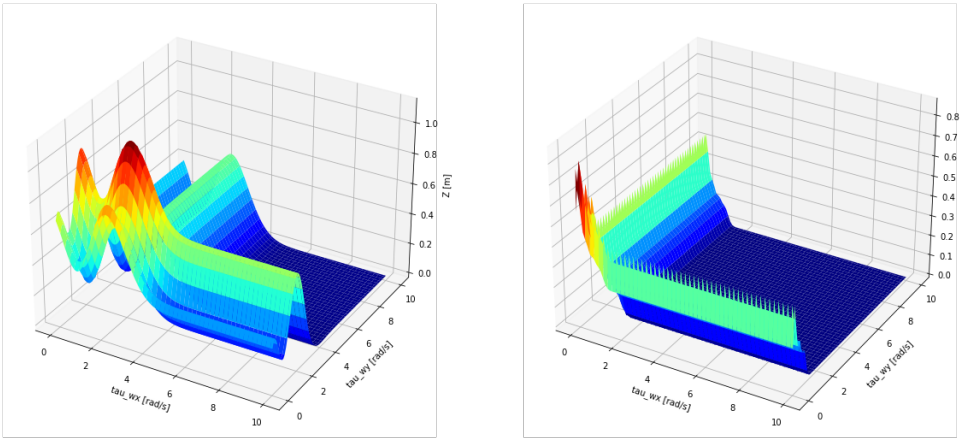

Figura 4.5: Ricostruzione della variabile con 50 training points. Sinistra: homemade; Destra: scikit-learn

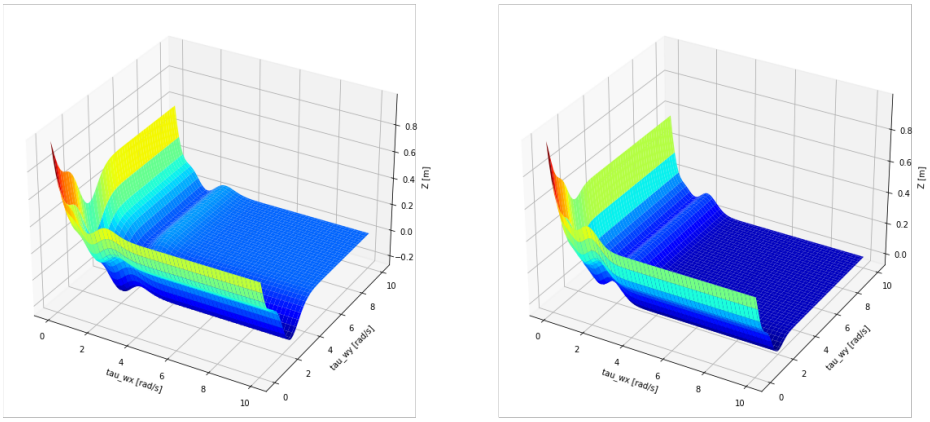

Figura 4.6: Ricostruzione della variabile con 100 training points. Sinistra: curva simulata con codice homemade; Destra: curva reale

rispettivamente nell'asse x e asse y.

4.2.1 MSE del GPr bidimensionale

L'MSE è il (mean squared error) ricordando la formula (3.3). Si utilizza come già spiegato precedenemente per andare a vedere l'errore generato dalla predizione rispetto a dei valori di test. I valori di testing points sono sempre i medesimi, mentre quelli di training sono randomici per ogni simulazione lanciata.

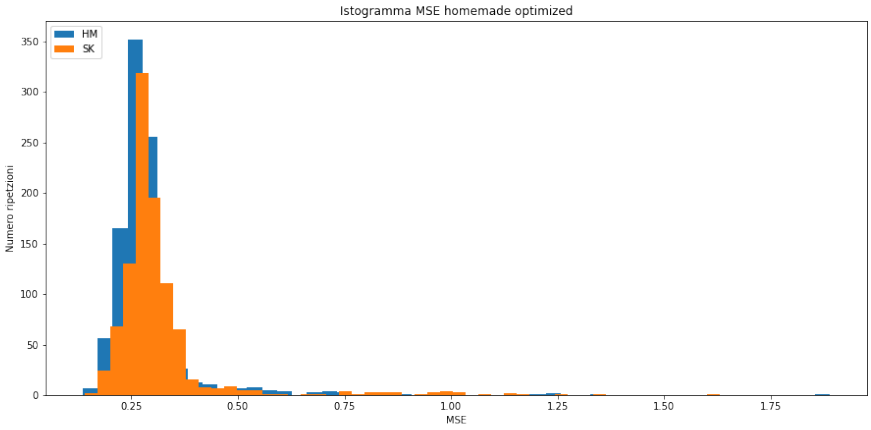

Figura 4.7: MSE del GPr bidimensionale

In Fig. 4.7 nelle ascisse abbiamo l'errore quadratico medio, mentre nelle ordinate ci sono il numero di ripetizioni, ovvero il numero di regressioni per cui è stato valutato il MSE. In altre parole per ogni predizione si vanno a sommare le differenze tra i valori di test e quelli di training e si ottiene un unico valore che è il mean squared error. Per ogni simulazione i valori di test sono sempre gli stessi, mentre quelle di training variano ogni volta.

Capitolo 5

Controllo del drone

5.1 Introduzione al problema di controllo

Nell'ottica di controllare la dinamica del drone e di tracciare dunque traiettorie specifiche durante il volo, riulta necessaria l'integrazione di un sistema di controllo. Naturalmente visto il forte carattere non lineare del sistema in esame, un semplice controllo di tipo PID non è in grado di gestire (non stabile) la complessa dinamica del drone. Tecniche di controllo più sofisticate sono richieste per poter gestire tale sistema. Fra tutte, per motivi legati alla robustezza e accuratezza del metodo si è scelto di utilizzare un controller di tipo MPC (Model Predictive Control). Il MPC è un tipo di controllo basato sul modello. La necessità di doversi appoggiare su di un modello rappresenta una problematica di non semplice risoluzione. Infatti sebbene esistano già modelli teorici accurati, descrittivi della dinamica del drone (vedi modello matematico esposto nel Capitolo 2), questi ultimi contengono un elevato numero di parametri vista la complessità del sistema che si cerca di controllare (dinamica di un oggeto avente 6 g.d.l. nello spazio). Vi è dunque un oggettiva difficoltà nel costruire modelli affidabili, basandoci sulla formulazione dei principi della fisica. E' da rimarcare infatti che il sistema dinamico dato dal drone è influenzato da un enorme quantità di disturbi esterni come turbolenze, variazioni di temperatura, o la presenza o meno di precipitazioni; Disturbi che tra l'altro posso anche non essere esclusivamente di natura ambientale, si pensi nell'ipotesi di trasporto (o supporto) di un ferito, alle sollecitazioni che possono essere applicate al drone, etc. La variabilità e l'imprevedibilità di queste fenomeni esterni alla meccanica del drone si traduce in variabilità del modello descrittivo della dinamica del sistema e in continue variazioni dei suoi stessi parametri (e.i. il modello di volo con o senza precipitazioni, oppure con o senza un carico da dover trasportare è diverso). Per poter risolvere i problemi appena esposti si applica un approccio basato sui dati che ci permetta al fronte di misurazioni acquisite ad intervalli di tempo prestabiliti (si considera anche la possibilità di acquisire in continuo misurazioni del sistema in real time), di regredire volta per volta (utilizzando lo storico delle misurazioni acquisite e facendo affidamento soprattutto a quelle acquisite più di recente) il modello della dinamica del sistema. A questo punto la tematica del controllo introduce due problemi, si deve poter regredire il modello in esame (i) in modo veloce (controllo in real time) e (ii) sfruttando una

ridotta quantità di dati. Gli algoritmi di regressione che si prestano bene per poter rispondere a questi due punti, sono gli algoritmi di regressione sparsa; nello specifico nel presente lavoro si utilizza SINDy (Sparse Identification of Non-linear Dynamics) [17]

5.2 SINDy-Control

Di seguito viene esposto il principio di funzionamento dell'algoritmo. SINDy così come tutti gli altri algoritmi di regressione sparsa risolve un problema inverso sottodeterminato, tramite la promozione della sparsità (la sparsità rappresenta lo strumento con cui si applica la regolarizzazione del problema). Si consideri un sistema di equazioni, in grado di fittare perfettamente i dati ottenuti dalle misurazioni del nostro sistema, come quello riportato di seguito

$$\dot{x}_1 = f_1(x_1) + f_2(x_1) + f_3(x_1, x_2) + f_4(x_1, x_2) + f_5(x_2) + ...$$

$$\dot{x}_2 = f_6(x_1) + f_7(x_1) + f_8(x_1, x_2) + f_9(x_1, x_2) + f_{10}(x_2) + ...$$

$$\dot{x}_3 = f_{11}(x_1) + f_{12}(x_1) + f_{13}(x_1, x_2) + f_{14}(x_1, x_2) + f_{15}(x_2) + ...$$

L'applicazione di SINDy permette di ottenere un modello parsimonioso (pochi termini che lo caratterizzano) e interpretabile dal punto di vista fisico. Tale risultato viene raggiunto risolvendo un problema di ottimizzazione, il cui scopo non è soltando quello di fare best fitting, ma di raggiungere un compromesso fra fitting e il numero di termini descrittivi della dinamica del drone, per cui soluzione portera alla cancellazione dei termini meno significati del sistema di Eq. sopra riportato. SINDy, si regge sull'ipotesi di lavorare con sistemi sparsi nello spazio delle possibili funzioni che li possono descrivere. Il concetto di sparsità (pochi temrmini attivi nella dinamica) su cui si regge SINDy permette l'individuazione del modello anche in presenza di dati limitati e rumorosi. Oltretutto la sparsità previene l'overfitting, e permette di generalizzare il modello al di là dei dati utilizzati per regredire il modello stesso (training data). L'integrazione di SINDy con l'MPC, fa si che si risalga ad un modello computazionalmente trattabile e accurato che può essere regredito con una piccola qunatità di training data.

Andando più nel dettaglio, di seguito viene presentato lo schema di funzionamento del codice

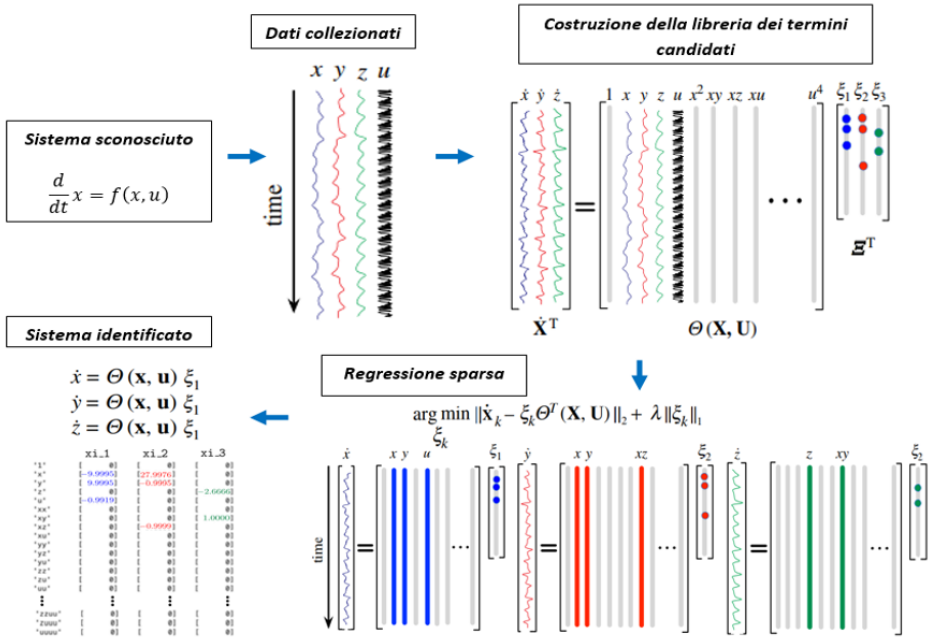

Figura 5.1: Schema funzionale dell'algoritmo di regessione sparsa SINDy [17].

Da quanto si può osservare da Fig. 5.1 SINDy è stato pensato per poter lavorare con un generico sistema dinamico non lineare (il cui vettore x rappresenta il vettore dello stato del sistema), permettendo di poter includere all'interno di esso anche dei parametri e.i. u che possono essere usati per il controllo, o meglio che rappresentano gli input control (vedi Eq. (5.1))

$$\frac{d}{dt}\mathbf{x} = f(\mathbf{x},\mathbf{u}), \quad \mathbf{x}(0) = \mathbf{x_0} \tag{5.1}$$

La procedura Data-driven ha inizio con l'acquisizione dei dati tramite misurazioni del sistema. Si misurano m snapshot dello stato del sistema x e del segnale di input u e li si riorganizza in due matrici:

$$X = [\mathbf{x_1}\mathbf{x_2}...\mathbf{x_m}], \quad U = [\mathbf{u_1}\mathbf{u_2}...\mathbf{u_m}] \tag{5.2}$$

Utilizzando queste due matrici è possibile costruire la libreria di termini candidati lineari e non $\Theta(X,U)$.

$$\Theta(X,U) = [1^T \quad X^T \quad U^T \quad (X\otimes X)^T \quad (X\otimes U)^T \quad ... \quad \sin(X)^T \quad \sin(U)^T \quad ... \quad] \tag{5.3}$$

Una volta ricostruita la libreria il sistema (5.1) può essere riscritto come

$$\dot{X} = \Xi\Theta^T(X,U) \tag{5.4}$$

il quale rappresenta il problema inverso da dover risolvere. I coefficienti di Ξ sono sparsi per molti sistemi dinamici compreso quello in esame. Dunque, si applica la

regressione sparsa per ricavare un Ξ sparso:

$$\xi_k = argmin_{\xi_k} \frac{1}{2} \|\dot{X}_k - \hat{\xi}_k \theta^T(X, U)\|_2^2 + \lambda \|\hat{\xi}_k\|_1 \tag{5.5}$$

Dove $\dot{X}_k$ rappresenta la k-esima riga di $\dot{X}$ e ξ_k la k-esima riga di Ξ. Si osserva che nell'Eq. (5.5) vi è il primo termine che promuove il fitting del modello con i dati, mentre il secondo termine che invece promuove la sparsità; La soluaione rappresenta un compromesso fra le due parti. Questa ottimizzazione viene risolta tramite un sequentially thresholded least squares (procedura che promuove la sparsità, vedi Fig. 5.2).

Algorithm 1. Sequentially thresholded least squares to learn the active library components.

Input: Time derivative $\dot{\mathbf{X}}$, library of candidate functions $\boldsymbol{\theta}^{\mathrm{T}}(\mathbf{X}, \mathbf{U})$, thresholding parameter ε
Output: Matrix of sparse coefficient vectors $\boldsymbol{\Xi}$

function STLS_REGRESSION($\dot{\mathbf{X}}, \boldsymbol{\theta}^{\mathrm{T}}(\mathbf{X}, \mathbf{U}), \varepsilon, N$)
  $\hat{\boldsymbol{\Xi}}^0 \leftarrow (\boldsymbol{\theta}^{\mathrm{T}})^{\dagger} \dot{\mathbf{X}}$ ▷ Initial least squares guess.
  while not converged **do**
    $k \leftarrow k + 1$
    $\mathbf{I}_{\text{small}} \leftarrow (\mathtt{abs}(\hat{\boldsymbol{\Xi}}) < \varepsilon)$ ▷ Find small entries ...
    $\hat{\boldsymbol{\Xi}}^k(\mathbf{I}_{\text{small}}) \leftarrow 0$ ▷ ... and threshold.
    for all variables **do**
      $\mathbf{I}_{\text{big}} \leftarrow \sim \mathbf{I}_{\text{small}}(:, ii)$ ▷ Find big entries ...
      $\hat{\boldsymbol{\Xi}}^k(\mathbf{I}_{\text{big}}, ii) \leftarrow (\boldsymbol{\theta}^{\mathrm{T}}(:, \mathbf{I}_{\text{big}}))^{\dagger} \dot{\mathbf{X}}(:, ii)$ ▷ ... and regress onto those terms.
    end for
  end while
end function

Figura 5.2: Sequential threshold least square. Figura presa dal lavoro di Brunton et.al [17]

5.3 MPC

Come già esposto l'MPC basa il suo funzionamento sulla conoscenza del modello del sistema che si vuole controllare. Iterativamente con un certo step temporale fissato, SINDy genera un nuovo modello, il quale supporterà il sistema di controllo. L'MPC risolve un problema di ottimizzazione sul controllo rispetto un orizzonte temporale che avanza, soggetto a vincoli sul sistema, per determinare la successiva azione di controllo. Questa ottimizzazione si ripete ad ogni time step, e la legge di controllo viene aggiornata come mostrato in Fig. 5.3 (b). Con l'MPC si determina la sequenza ottimale di control inputs $u(\cdot|x_j) := \{u_{j+1}, \quad ..., \quad u_{j+k}, \quad ..., \quad u_{j+m_c}\}$ lungo l'orizzonte di controllo $T_c = m_c \Delta t$ dato lo stato del sistema all'istante corrente x_j che minimizza la funzione di costo J lungo l'orizzonte di predizione; Δt è il time step del modello, il quale può essere diverso dal tempo di campionamento delle misurazioni. Il primo input control u_{j+1} viene applicato, e l'ottimizzazione viene re-inizializzata e ripetuta ad ogni successivo time step, per progettare la sequenza di

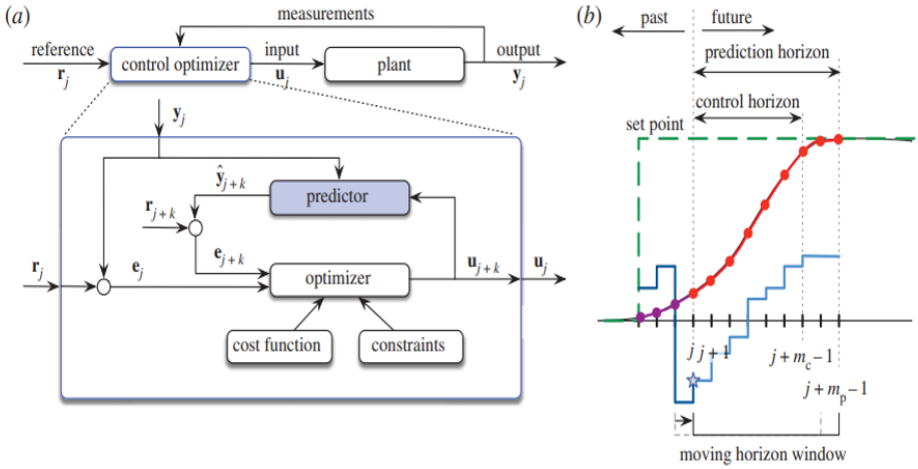

Figura 5.3: MPC. figura presa da [18].

attuazione $u(\cdot|x_j)$. Ciò si traduce in un'implicita legge del feedback control

$$K(x_j) = u(j+1|x_j) = u_{j+1} \tag{5.6}$$

dove u_{j+1} è il primo input nella sequenza di attuazione ottimizzata partendo dalla condizione iniziale x_j.

Come ultimo step dell'analisi del MPC ci si sufferma brevemente sulla funzione di costo che viene utilizzata nel problema di ottimizzazione. La suddetta funzione di costo che viene ottimizzata ad ogni time step è data da:

$$\begin{aligned} min_{u(\cdot|x_j)}\, J(x_j) = min_{u(\cdot|x_j)}[&\|\hat{x}_{j+m_p} - x^*_{m_p}\|^2_{Q_{m_p}} + \textstyle\sum_{k=0}^{m_p-1} \|\hat{x}_{j+k} - x^*_k\|^2_Q \\ &+ \textstyle\sum_{k=1}^{m_c-1}(\|\hat{u}_{j+k}\|^2_{R_u} + \|\Delta\hat{u}_{j+k}\|^2_{R_{\Delta u}})] \end{aligned} \tag{5.7}$$

Da quanto esposto sopra, si osserva che la funzione di costo (5.7) è influenzata da:

- La dinamica (discretizzata nel tempo) del sistema: $\hat{x}_{k+1} = \hat{F}(\hat{x}_k, u_k)$
- Vincoli sul Δ input:$\Delta u_{min} \leq \Delta u_k \leq \Delta u_{max}$ (si sottolinea che in generale i vincoli possono essere di vario tipo)
- Vincoli sul Δ input:$u_{min} \leq u_k \leq u_{max}$ (si sottolinea che in generale i vincoli possono essere di vario tipo)

La funzione J penalizza le deviozioni dello stato predetto $\hat{x}_k$ dallo stato x^*_k e include nel costo anche $\hat{x}_{m_p}$. Anche le ampiezze u_k e le variazioni di ampiezza Δu_k sono penalizzate, in modo da minizzare l'azione del controllo sulla traiettoria. Ogni termine è calcolato come norma pesata di un vettore, e.i. $\|x\|^2_Q := x^T Q x$. Le matrici dei pesi $Q \geq 0$, $Q_{m_p} \geq 0$, $R_u \geq 0$ e $R_{\Delta u} \geq 0$, sono rispettivamente definite semi-definite positive e positive. Si noti che con tale approccio si ammette che il

modello predetto possa essere diverso dalle misurazioni acquisite. Mentre il modello e la legge di controllo, possono essere appresi simultaneamente, nel presente lavoro si sceglie di adottare un approccio a due stadi, dove prima viene ricavato il modello e poi utilizzato per ottimizzare il controllo con l'MPC.

5.4 Risultati MPC-SINDyC

Nella presente sezione vengono riportati i risultati relativi all'applicazione del MPC supportato da SINDy in cui si lavora con uno e con quattro parametri di controllo . Nel problema in esame si cerca di controllare la traiettoria del sistema agendo sulla velocità di rotazione dei rotori. Nelle simulazioni svolte i dati relativi alle coppie areodinamiche agenti sul sistema sono stati presi da letteratura in modo da poter lavorare in condizioni verosimili.

5.4.1 MPC con SINDyC con un parametro di controllo

Come caso esempio per verificare le prestazioni del framework SINDyC-MPC si è andato ad agire sulle condizioni al contorno sulle velocità dei rotori, imponendo che queste assumano fra loro sempre lo stesso valore per ogni istante di tempo. A seguito delle condizioni al contorno imposte è possibile schematizzare il sistema come controllato da un singolo parametro. Una volta definite le condizioni di funzionamento del drone, si è proceduto simulando la risposta del sistema a seguito di una variazione improvvisa del valore dell'angolo di roll.

In Fig. 5.4 si plotta l'andamento del segnale dell' attuatore in risposta ad una variazione dell'angolo di roll. In Fig. 5.5 viene riportato l'andamento delle tre variabili angolari e della quarta variabile la quale si sta cercando di controllare. L'angolo di roll infatti deve seguire quello che è l'andamento del reference value, riportato anch'esso in Fig. 5.5 (linea nera continua, r).

Il framework SINDyC-MPC è capace di controllare sistemi fortemente non lineari, esclusivamente tramite acquisizione di dati ottenuti dalle misurazioni, e l'identificazione del modello è veloce abbastanza da permetterne l'applicazione in tempo reale, anche in risposta a cambiamenti improvvisi e bruschi della dinamica del modello.

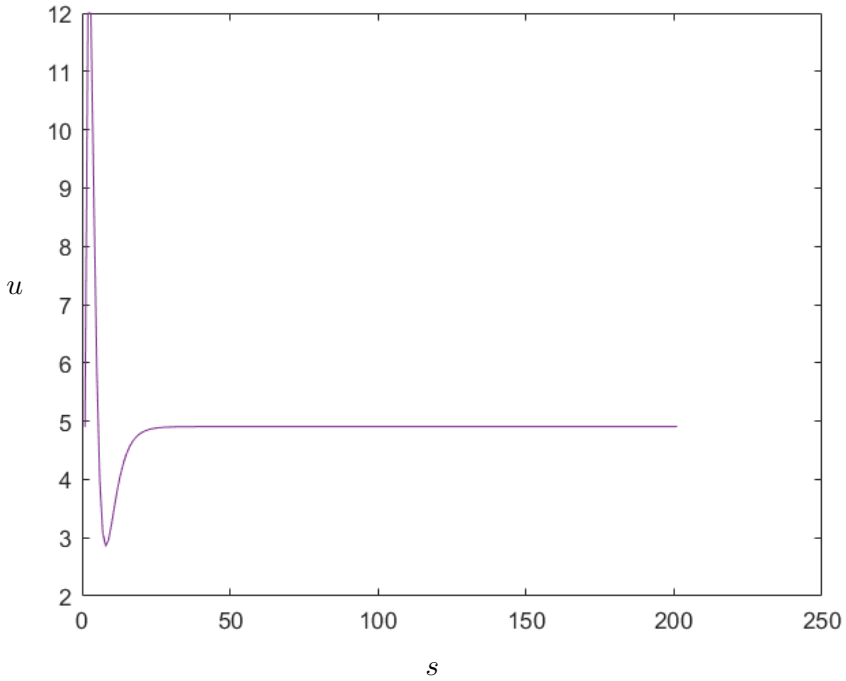

Figura 5.4: Andamento del segnale dell'attuatore che permette di seguire l'andamento del reference value dell'angolo di roll

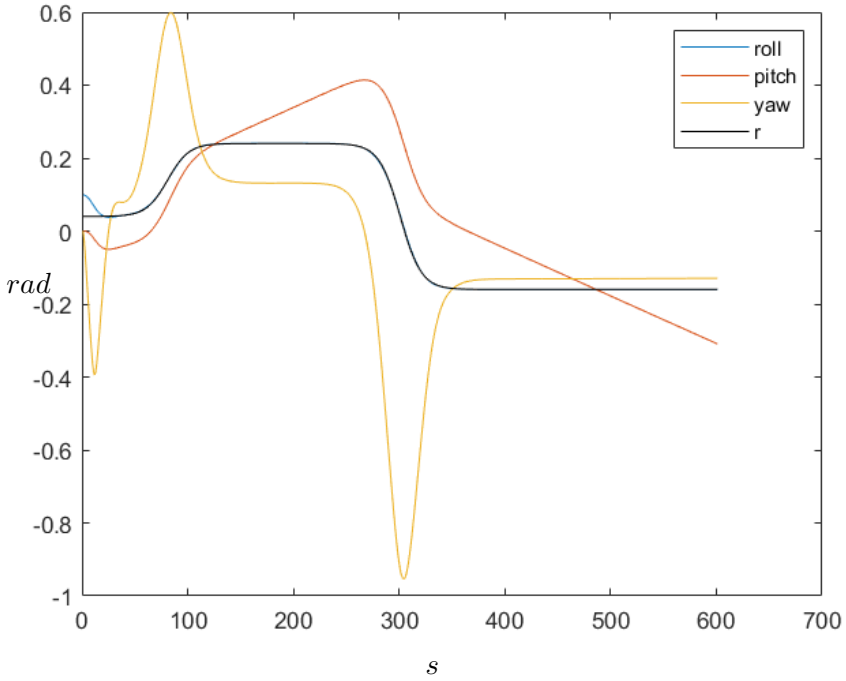

Figura 5.5: Controllo rispetto all'angolo di Roll. r=reference.

Capitolo 6

Conclusioni

In questo lavoro si e' analizzata la dinamica di un drone. Si e' partiti dallo studio delle equazioni del moto e si e' studiata la complessita' del sistema a seguito della numerosita' delle variabili e dei parametri in gioco.

Simulando alcune traiettorie a partire da condizioni iniziali diverse si e' analizzata l'influenza di parametri essenziali come le coppie applicate ai motori. La grandezza del set di parametri rende impossibile generare curve di risposta del sistema per qualsiasi combinazione. E' per questo motivo che si e' deciso di predirre la dinamica attraverso techiche di machine learning. Si e' cercato di capire come regredire la dinamica con approccio Gaussiano e verificare l'affidabilita' della soluzione proposta tramite validazioni numeriche.

La regressione dei modelli e' stata studiata al variare dei kernel e analizzandone la sensibilita' rispetto al set di training. Tali informazioni di input possono essere piu' o meno sufficienti per di catturare a pieno la dinamica ed estrapolare scenari fattibili.

La regressione e' stata estesa anche al campo bidimensionale e confrontata con la predizione basata con reti neurali.

Un secondo studio e' affrontato inserendo il controllo del drone. Il framework SINDyC-MPC è capace di controllare sistemi fortemente non lineari, esclusivamente tramite acquisizione di dati ottenuti dalle misurazioni, e l'identificazione del modello è veloce abbastanza da permetterne l'applicazione in tempo reale.

Dei futuri sviluppi possono essere l'utilizzo di ottimizzazione per il controllo globali, quando si hanno più parametri da ottimizzare ovviamente le ottimizzazione locali non sono attendibile e potrebbero non dare la soluzione corretta. D'altro canto le ottimizzazione globali hanno un costo computazione molto più elevato

Capitolo 7

Bibliografia

[1] http://krossblade.com/history-of-quadcopters-and-multirotors/

[2] http://www.fai.org/

[3] R.V. Jategaonkar. Flight vehicle system identification: a time domain methodology. American Institute of Aeronautics and Astronautics, 2006.

[4] I.D. Cowling, O.A. Yakimenko, J.F. Whidborne, and A.K. Cooke. A prototype of an autonomous controller for a quadrotor uav. In European Control Conference, pages 1–8, 2007.

[5] S. Bouabdallah, A. Noth, and R. Siegwart. PID vs LQ control techniques applied to an indoor micro quadrotor. In International Conference on Intelligent Robots and Systems 2004, volume 3, pages 2451 – 2456 vol.3, 2004

[6] P. Pounds, R. Mahony, and P. Corke. Modelling and control of a quadrotor robot. In Australasian conference on robotics and automation 2006, Auckland, NZ, 2006.

[7] T.Madani and A. Benallegue. Backstepping Control for a Quadrotor Helicopter. In Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on, pages 3255 –3260, 2006.

[8] S.Buabdallah and R. Siegwart. Backstepping and sliding-mode techniques applied to an indoor micro quadrotor. In International Conference on Robotics and Automation 2005, pages 2247 – 2252, april 2005.

[9] R. Xu and U. Ozguner. Sliding mode control of a quadrotor helicopter. In Decision and Control, 2006 45th IEEE Conference on, pages 4957 –4962, dec. 2006.

[10] D. Lee, H. Jin Kim, and S. Sastry. Feedback linearization vs. adaptive sliding mode control for a quadrotor helicopter. International Journal of Control, Automation and Systems, 7(3):419–428, 2009.

[11] A. Das, K. Subbarao, and F. Lewis. Dynamic inversion with zero-dynamics stabilisation for quadrotor control. Control Theory & Applications, IET, 3(3):303–314, 2009.

[12] B. Whitehead and S. Bieniawski. Model Reference Adaptive Control of a Quadrotor UAV. In Guidance Navigation and Control Conference 2010, Toronto, Ontario, Canada, 2010. AIAA.

[13] M.Huang, B.Xian, C.Diao, K.Yang, and Y.Feng. Adaptive tracking control of underactuated quadrotor unmanned aerial vehicles via backstepping. In American Control Conference (ACC), 2010, pages 2076 –2081, 30 2010-july 2 2010.

[14] W. Zeng, B. Xian, C. Diao, Q. Yin, H. Li, and Y. Yang. Nonlinear adaptive regulation control of a quadrotor unmanned aerial vehicle. In Control Applications (CCA), 2011 IEEE International Conference on, pages 133 –138, 2011.

[15] D. Mellinger, Q. Lindsey, M. Shomin, and V. Kumar. Design, modeling, estimation and control for aerial grasping and manipulation. In Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on, pages 2668 –2673, sept. 2011.

[16] J.J. Craig, P. Hsu, and S.S. Sastry. Adaptive control of mechanical manipulators. The International Journal of Robotics Research, 6(2):16, 1987.

[17] Brunton, S. L., Proctor, J. L., and Kutz, J. N., 2016. "Dis-covering governing equations from data by sparse identifi-cation of nonlinear dynamical systems".Proceedings of theNational Academy of Sciences,113(15), pp. 3932–3937

[18] Kaiser E., Kutz J. N. and Brunton S. L. 2018Sparse identification of nonlinear dynamics for model predictive control in the low-data limitProc. R. Soc. A.474: 20180335. 20180335