UNIVERSITÀ POLITECNICA DELLE MARCHE

FACOLTÀ DI INGEGNERIA

Corso di Laurea Triennale in Ingegneria Informatica e dell'Automazione

Studio e simulazione di un sistema NGC per robot biomimetici pesciformi

Study and simulation of a NGC system for biomimetical robots

Relatore:

Prof. David Scaradozzi

Correlatore:

Dott. Daniele Costa

Dott. Nicolò Ciuccoli

Rapporto Finale di:

Tommaso Pieroni

Anno Accademico 2019/2020

Indice

Introduzione

Questo lavoro si pone l'obiettivo di trovare un possibile modello matematico del movimento sull'asse verticale tramite un motore brushless di un robot pesciforme e di svilupparne un controllore che permetta a tale veicolo subacqueo di raggiungere una profondità preimpostata come riferimento. Lo studio è situato dentro l'ambiente di ricerca del Dipartimento di Ingegneria dell'Informazione dell'Università Politecnica delle Marche, precisamente nell'area tematica di Robotica Sottomarina che lavora allo sviluppo di veicoli marini biomorfi e che viene supervisionata dal Prof. David Scaradozzi. La seguente tesi si sviluppa attorno al concetto di identificazione e progettazione di un modello studiandone delle coppie input-output.

La prima parte del lavora sarà dunque quella di ricavare tali valori attraverso la descrizione dell'ambiente fisico in cui il robot dovrà andare ad agire e quindi, attraverso una successiva simulazione, andare a valutare lo spostamento lungo l'asse dato dalla velocità dell'elica del motore centrale impostato come ingresso: tali informazioni costituiranno le coppie di dati utili all'identificazione. Per sviluppare il modello fisico è stato usato come ambiente di lavoro il software CAD Adams fornito da MSC Software. Per la seconda parte relativa all'identificazione si è utilizzato il software Matlab e per il collegamento tra i due ambienti Simulink. Si ha quindi una prima parte di spiegazione del lavoro svolto su Adams e di cosa sia Matlab in breve; una seconda parte di approfondimento di quello che è Simulink, cioè il software necessario alla condivisione dei dati tra Matlab e Adams.

Verranno poi presentate prima le formule fisiche che governano il pesce robot ed in generale la fisica dei veicoli subacquei, la sua struttura, in linea teorica il problema dell'identificazione e la sua risoluzione, adottando il metodo black box. Soltanto a questo punto si spiegheranno i passaggi che hanno portato alla definizione del modello che meglio descrive il comportamento del pesce robot portando al risultato armax(3 1 6 0). Occorre precisare che la valutazione del modello considera solamente le famiglie dei modelli arx e armax, partendo dall'ipotesi di equilibrio del pesce in cui la forza peso viene bilanciata dalla spinta di Archimede.

Una volta individuato il modello e sviluppate tutte le considerazioni del caso tra cui una verifica della bontà del sistema tramite valutazione dell'incertezza parametrica, si effettuerà un controllo sul modello importato da Adams: tale modello sarà variato rispetto al precedente supponendo di non essere in una condizione di equilibrio e quindi considerando diversi i moduli della forza peso e della spinta archimedea. In questa ultima sezione il controllo verrà effettuato tramite PID

(proporzionale, integrativo e derivativo) che agirà su un modello Simscape di Simulink di un motore brushless.

1. Il software Adams

Il primo strumento necessario allo sviluppo del modello è il software Adams: utilizzato per la dinamica multibody, supporta gli ingegneri nello studio della dinamica delle parti in movimento e nell'analisi della distribuzione di forze e carichi attraverso i sistemi meccanici. Tramite Adams sono stati costruiti il pesce e le forze che lo governano in modo tale da sviluppare poi una raccolta dati che prevede in ingresso la rotazione dell'elica di uno dei motori del robot - tale argomento verrà approfondito nel Cap. 4 - e in uscita la profondità raggiunta dal pesce. L'ambiente Adams si presenta come in *figura 1.1*. Dalla sezione in alto riportata in *figura 1.2* si riesce ad interagire con lo spazio sottostante: il modello del pesce e di tutte le sue componenti è stato importato come sviluppo di un lavoro precedente effettuato sempre nel dipartimento di Ingegneria dell'Informazione dell'Università Politecnica delle Marche. Quello che è stato fatto è andare ad applicare una serie di forze e di misure al robot e assegnare ad esso una precisa massa: chiaramente facendo ciò si è supposta una massa uniforme, ma tale approssimazione, come verrà poi spiegato in seguito, non è fuorviante ai fini del risultato. Una volta importato il file, si inizia a lavorare sul modello rappresentato in *figura 1.3*: tutte le componenti del pesce sono inizialmente divise e per trovare il centro di massa, su cui poi applicare le forze, si sono dovute fondere le singole componenti tramite il comando  (merge) evidenziato in *figura 1.2* nella sezione "Booleans" di "Bodies". In tal modo, una volta completato il merge di tutti i solidi presenti, è stato possibile individuare il centro di massa del veicolo che è stato usato come punto di applicazione delle forze agenti. Prima di andare ad inserire le forze nel progetto Adams, è stata inserita la massa del robot facendo un doppio click sul pesce e agendo sulla finestra di dialogo come riportato in *figura 1.4*. Una volta completato ciò, tramite la sezione "Forces" e il comando  sono state create le diverse forze che verranno definite nel Cap. 4: attraverso la finestra di dialogo è stato assegnato il rispettivo valore alle singole forze anche in forma di equazione. Inoltre è stato annullato il contributo, e quindi l'effetto, della forza peso poichè, come già accennato, si affronterà il problema dell'identificazione del robot inizialmente sotto ipotesi di equilibro del veicolo in cui la forza peso eguaglia la spinta di Archimede. Per la costruzione delle formule è stato necessario creare, tramite la sezione "Elements" (indicata anch'essa in *figura 1.2*) e il comando , un'incognita che definisse la variabile di controllo vale a dire la velocità di rotazione del motore, la quale verrà passata ad Adams nel modo che verrà visto in seguito. L'ultimo lavoro è stato quello relativo alla definizione di alcune misure necessarie alla valutazione dell'andamento del robot: per questo è stata messa in evidenza una misura della variabile creata, una della velocità e della

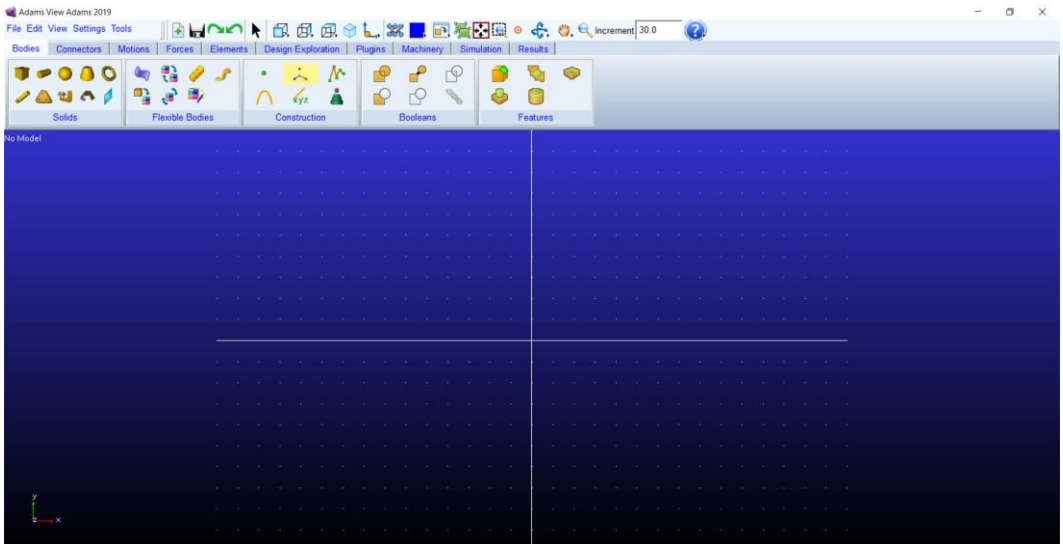

Figura 1.1: presentazione ambiente Adams

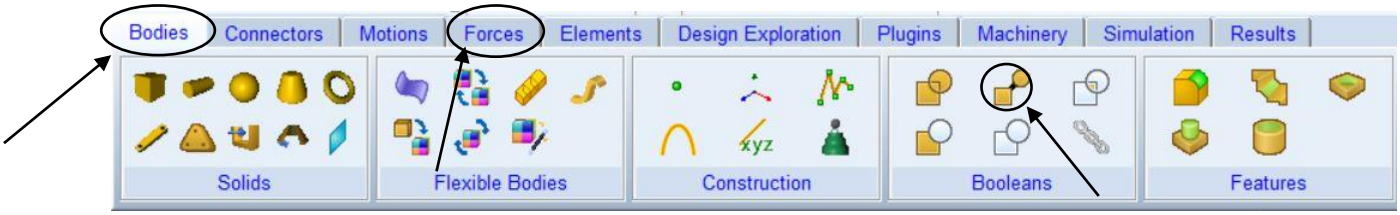

Figura 1.2: principali comandi Adams per il merge e la creazione di variabili

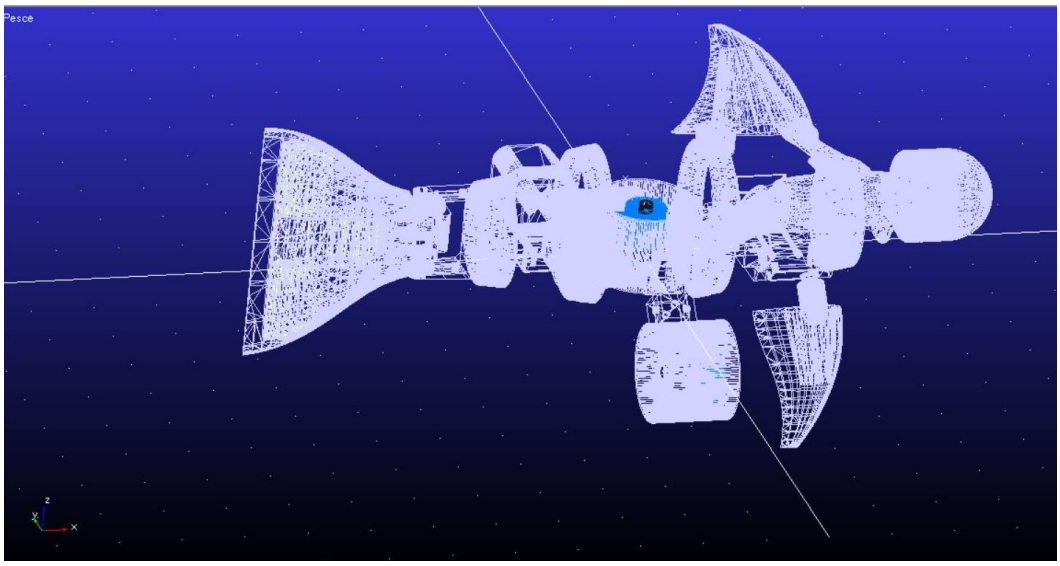

Figura 1.3: file importato in Adams

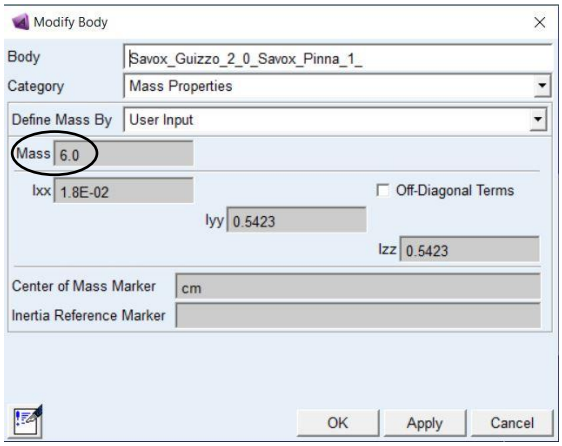

Figura 1.4: finestra per la definizione della massa del veicolo più dettaglio del relativo valore

accelerazione del centro di massa del pesce e una della profondità raggiunta dal centro di massa del robot. Volendo controllare tale profondità risulta intuitivo che questa misura dovrà coincidere con la variabile controllata che verrà retroazionata nel controllo e valutata come dato di uscita nell'identificazione. Per fare tutto questo attraverso Adams occorre andare sulla sezione "view" indicata nella figura 1.1 e selezionare le grandezze che si vogliono graficare.

2. Il software Matlab

In questa sezione verrà brevemente presentato il software Matlab e la modalità del suo utilizzo per lo sviluppo del lavoro effettuato. Matlab, abbreviazione di Matrix Laboratory, è un ambiente per il calcolo numerico e l'analisi statistica scritto in C e che comprende anche l'omonimo linguaggio di programmazione creato dalla MathWorks. Il software consente di manipolare matrici, visualizzare funzioni e dati, implementare algoritmi, creare interfacce utente e interagire con altri programmi e, in particolare, si vedrà il collegamento con Simulink. L'interfaccia principale di Matlab, come mostrato in *figura 2.1*, è suddivisa in quattro zone: la Command Window nella quale è possibile inserire righe di codice che verranno poi istantaneamente eseguite; il Workspace nel quale sono salvate tutte le variabili e gli oggetti creati dall'esecuzione di righe di codice; la Current Directory che risulta essere di notevole importanza nell'apertura dei file e degli elaborati con i quali interagire; infine, la zona della Command History permette di dare uno sguardo ai comandi eseguiti di recente. Il lavoro svolto è stato implementato come Live Script e non come semplice Script, in questo caso per ragioni di leggibilità: infatti, il Live Script permette di commentare e suddividere in sezioni il codice con la possibilità di andare ad eseguire una sola parte di codice per volta. I file scritti su Matlab sono stati due, il primo contiene i passi dell'identificazione mentre il secondo è utile per il controllo del pesce: entrambi i file con relativi pezzi di codice, scritti e spiegati, saranno illustrati nei capitoli successivi. Matlab risulta anche di notevole interesse per la capacità di offrire dei toolbox che permettono un ampliamento delle funzionalità del software per specifici settori di interesse: nel caso in esame verrà utilizzato il toolbox per l'identificazione e in particolare il comando chiamato "systemidentification" che apre una finestra di dialogo utile per lo sviluppo e la ricerca di un modello che meglio approssimi una serie di dati.

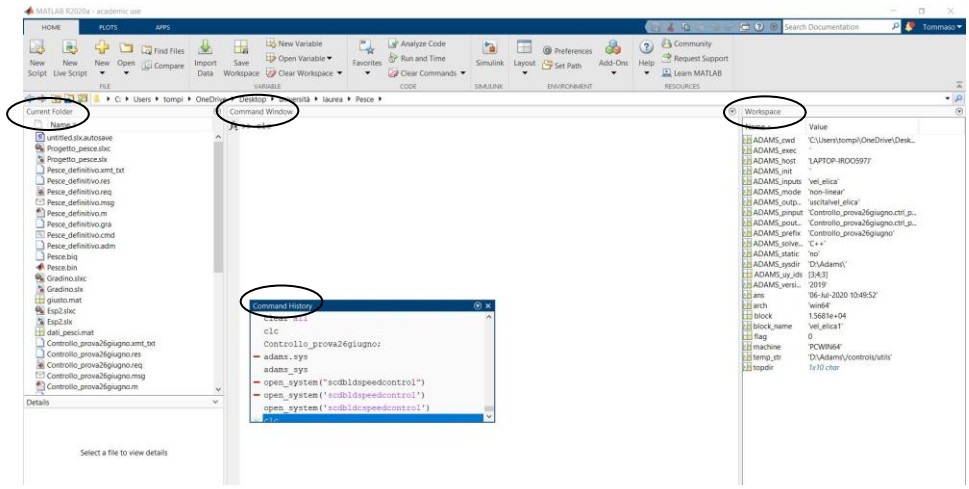

Figura 2.1: interfaccia iniziale di Matlab

3. Lo sviluppo tramite Simulink e Simscape

Simulink è un software per la modellazione, la simulazione e l'analisi di sistemi dinamici che sfrutta una logica a blocchi logici come in *figura 3.1*. Esso risulta necessario nel momento in cui si vuole collegare Matlab e Adams ed è possibile accedervi tramite la sezione apposita nella pagina iniziale di Matlab. La grande possibilità di Simulink è quella di poter simulare un comportamento collegando diversi programmi tramite blocchi, ciascuno dei quali prevede una specifica funzione e che possiede diversi ingressi e uscite.

Per ampliare maggiormente le funzionalità e i blocchi disponibili oltre a quelli base, vengono usate delle espansioni dentro Simscape il quale amplia Simulink con strumenti per la modellazione e la simulazione di sistemi fisici multidominio. Quindi, Simscape permette di creare e costruire modelli di componenti fisici basati su collegamenti fisici che si integrano direttamente con diagrammi a blocchi e altri paradigmi di modellazione: in particolare, in questo lavoro verrà ampliato Simulink mediante Simscape Electrical poiché, come verrà esposto sul capitolo relativo al controllo, sarà utile andare a importare un modello in Simulink di un motore Brushless. Come poi si vedrà in maniera più approfondita, l'obiettivo è quello di mandare un segnale in input al modello del pesce-robot in Adams e di ricevere in output un altro segnale che è la profondità del centro di massa: in sostanza, attraverso Simulink, si dovrà collegare il segnale generato in Matlab al modello costruito in Adams per poi ricevere da quest'ultimo un segnale che sarà poi da trasferire nuovamente a Matlab.

La situazione verrà affrontata in linea generale senza particolarizzare la questione al caso in esame poiché sarà approfondito nei successivi capitoli.

Per prima cosa è stato necessario creare il blocco Simulink relativo al modello in Adams del pesce. Per fare ciò occorre accedere a "Controls" dalla sezione "Plugins" ed andare su "Plant export": a questo punto, si apre una finestra di dialogo come quella riportata in *figura 3.2* in cui occorre inserire il nome del file di estensione .m (estensione dei file Matlab) e selezionare le variabili di ingresso e di uscita. Quindi si specifica, come mostrato in figura, il programma con il quale si vorrà leggere il file vale a dire Matlab, nel caso in esame. A questo punto si passa alla Command Window di Matlab e si apre il file appena creato: bisogna però trovarsi nella giusta cartella di lavoro cioè quella in cui è stato salvato il file export di Adams. Così, tramite il comando "adams_sys", viene aperto un progetto Simulink in cui si ha proprio il blocco Adams rappresentato in *figura 3.3*.

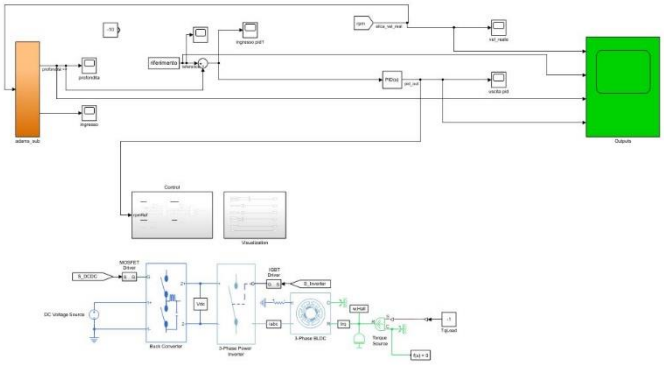

Figura 3.1: esempio di schema a blocchi su Simulink

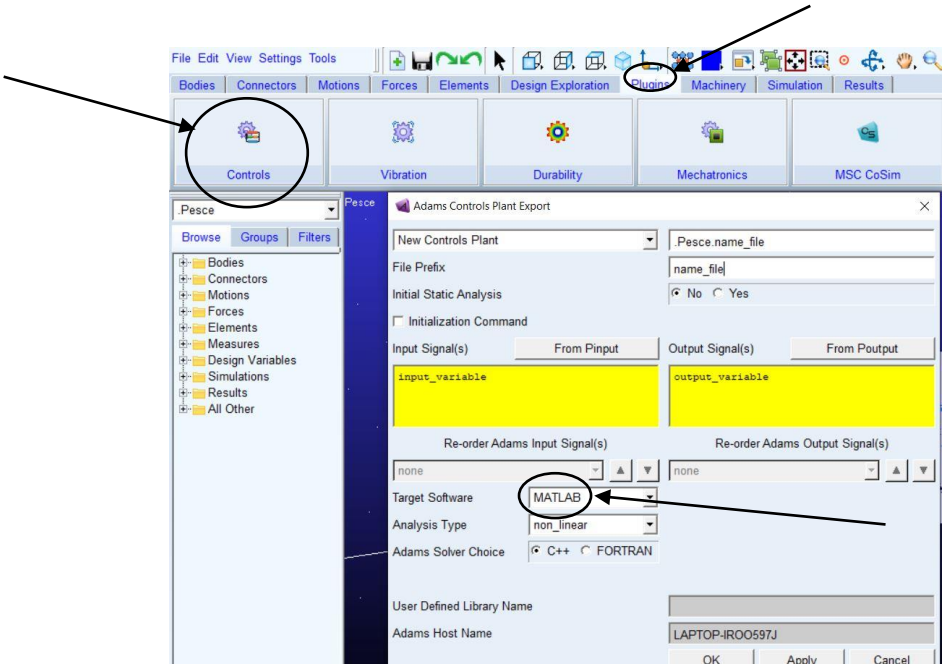

Figura 3.2: finestra per l'export del sistema da Adams con dettagli evidenziati

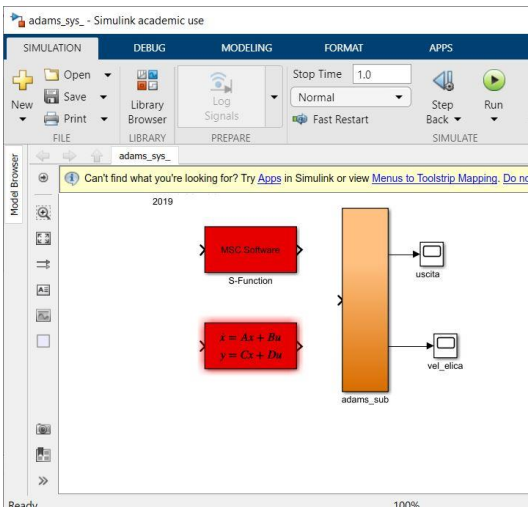

Figura 3.3: rappresentazione in Simulink del progetto esportato da Adams

Simulink permette l'esecuzione del sistema creato, e quindi la sua simulazione, in due modi: o a tempo continuo o a tempo discreto. Per il lavoro in esame viene usato la seconda modalità poiché si ha un sistema a tempo discreto e, in particolare, viene fatto il setting dello step con cui vengono passati i valori ad Adams e vengono letti i valori in uscita dal blocco. Questa cosa viene fatta per avere coerenza fra lo step di lavoro del blocco Adams, quello di lavoro di Simulink e quello con cui cambiano i dati. Infatti, il blocco Adams riceve in ingresso, non solo i valori, ma anche la temporizzazione con cui questi vanno presi: per questo motivo, in ingresso si dovrà avere una matrice composta da due colonne e non singoli valori.

Un'ultima considerazione viene fatta sul collegamento con Matlab: poiché Simulink è un ambiente integrato, vi è la possibilità tramite blocchi "From Workspace" e "To Workspace" di interagire con la Workspace di Matlab: a tal proposito, è importante avere tutte le variabili che verranno poi utilizzate nella Workspace e, quindi, risulta conveniente avere il progetto nella stessa cartella di lavoro del codice Matlab. Risulta essere di particolare interesse pratico che l'esporto di dati da Simulink è fatto sottoforma di struct e che quindi l'accesso a questi dati va effettuato come se si lavorasse con delle strutture.

4. Il modello del pesce-robot

Il sistema reale preso in considerazione è quello di un pesce robotico costruito nel laboratorio di meccanica di Ingegneria dell'Università Politecnica delle Marche. A tal proposito tutte le considerazioni che verranno di seguito sviluppate, insieme alle formule riportate saranno ricavate dagli studi degli Ing. Daniele Costa e Ing. Nicolò Ciuccoli i quali hanno approssimato il robot del pesce ad un cilindro di raggio R e lunghezza L. Le considerazioni e le formule, con i relativi coefficienti, sviluppate da i due ingegneri, si rifanno in gran parte alla teoria dei veicoli subacquei sviluppata dal cibernetico norvegese Thor I. Fossen nel suo libro "*Handbook of Marine Craft Hydrodynamics and Motion Control*". Nei seguenti paragrafi verranno presentate quelle che sono le leggi su cui si basa il movimento del pesce robotico e lo sviluppo di tali formule nell'ambiente Adams.

4.1 Il pesce-robot

Il compito, in questa prima parte di lavoro, è quello di collegare in Adams il movimento del pesce lungo l'asse z, e quindi della profondità, alla variazione della velocità dell'elica del motore posto al centro del pesce stesso ed indicata dalla freccia nella *figura 4.1.1*. Il pesce robotico è formato da un corpo principale con una pinna snodabile e da tre motori, due laterali per il movimento lungo lo spazio xy e quello centrale con lo scopo inizialmente presentato, riportati in *figura 4.1.1* e figura *4.1.2*. L'equazione della dinamica che governa il movimento del pesce lungo l'asse z è riportata nell'eq. (1) seguente:

$$Z = -Z_{\dot{w}}\dot{w} + (Y_{\dot{v}}vp - Y_{\dot{u}}uq) - Z_{|w|w}|w|w + Z_{s1} + Z_{s2} + Z_{prop} \quad (1)$$

Prima di analizzare i singoli contributi della equazione riportata, occorre fare alcune precisazioni sul sistema di riferimento che è stato adottato: lo spazio di stato sarà descritto sia dalle terne (2) e (3), le quali rappresentano rispettivamente la posizione lungo gli assi e le componenti della velocità, sia dagli angoli di Eulero rispettivamente chiamati angoli di roll, pitch e yaw con le rispettive componenti delle velocità angolari riportate nelle terne (4) e (5).

$$\gamma_1 = [x, y, z]^T \quad (2)$$

$$v_1 = [u, v, w]^T \quad (3)$$

$$\gamma_2 = [\emptyset, \theta, \varphi]^T \quad (4)$$

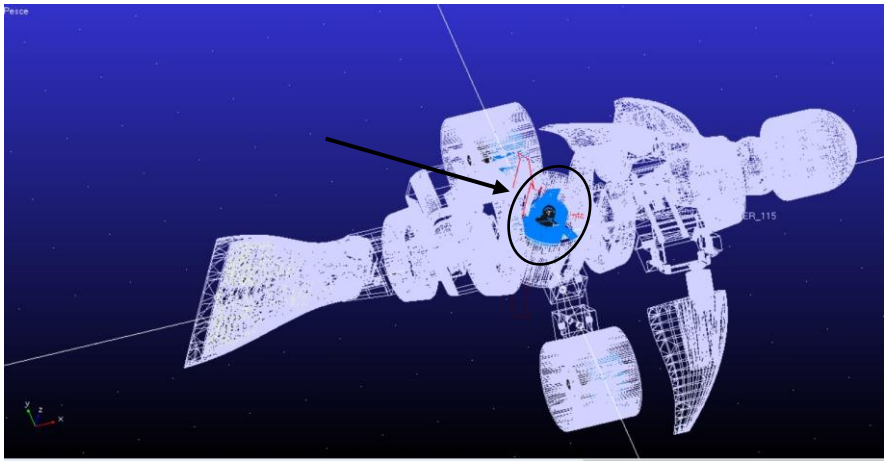

Figura 4.1.1: visione dall'alto del pesce con l'indicazione della posizione del motore centrale

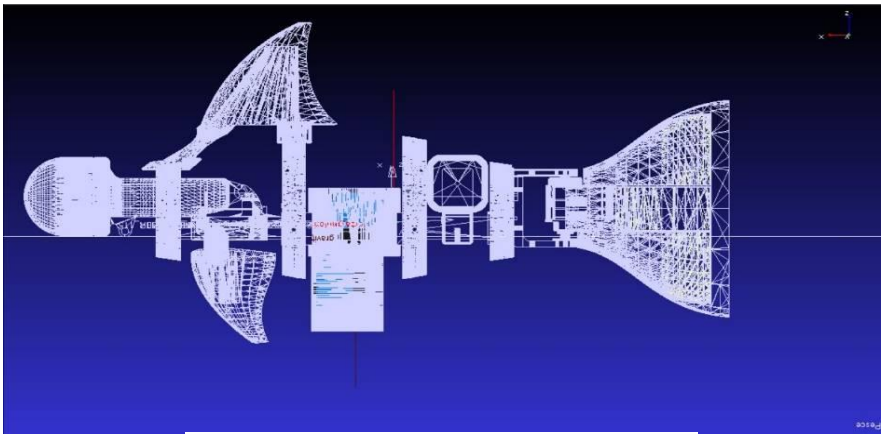

Figura 4.1.2: visione laterale del pesce

$$v_2 = [p, q, r]^T \tag{5}$$

In riferimento all'eq. (1) è necessario esplicare le componenti prese in considerazione, tra cui il primo contributo $-Z_{\dot{w}}\dot{w}$ il quale rappresenta la massa aggiunta prodotta dal movimento del pesce: esso, infatti, muovendosi sposta una massa pari al suo volume di acqua. Questo effetto incide negativamente nel suo spostamento poiché cerca di opporsi al movimento del robot e per questo è presente nella formula con segno negativo. Il valore $(Y_{\dot{v}}vp - Y_{\dot{u}}uq)$ non viene preso in considerazione poiché non contribuisce alle forze lungo l'asse z: difatti, si può notare la presenza delle velocità lungo gli assi x e y che sono nulle come già esposto nell'introduzione. Il terzo termine che si prende in considerazione è $-Z_{|w|w}|w|w$ il quale rappresenta un'approssimazione dell'attrito viscoso che subisce il pesce in movimento: per tale ragione compare anche in questo caso con

segno negativo poiché risulta essere una forza in opposizione al movimento del pesce, variando in maniera quadratica con la velocità.

Il pesce è composto da due pinne frontali inclinate di un angolo $\propto = 45°$ e una pinna caudale; queste, insieme, producono due forze F_{s1} e F_{s2} che possono essere decomposte in 4 componenti come mostrato in *figura 4.1.3* e che sono descritte dalle eq. (6-9):

$$F_{s1} = \frac{1}{2}\rho c_L S_{fin}(u^2\delta_{s1} + uw - x_{fin}uq) \quad (6)$$

$$F_{s2} = \frac{1}{2}\rho c_L S_{fin}(u^2\delta_{s2} + uw - x_{fin}uq) \quad (7)$$

$$Z_{s1} = -F_{s1}\cos \propto \quad (8)$$

$$Z_{s2} = -F_{s2}\cos \propto \quad (9)$$

Come già accennato in precedenza, verranno citate solamente le formule necessarie allo sviluppo di un modello del pesce lungo l'asse verticale. A tal proposito, si può già notare come il contributo delle forze citate nell'eq. (8) e nell'eq. (9) sia nullo poiché le velocità u e q hanno modulo nullo. Le formule riportano diverse costanti il cui valore è inserito nella *Tabella 4.1* mentre il loro significato viene di seguito riportato, in accordo con la teoria citata da Fossen: ρ è la densità dell'acqua, c_L è il coefficiente di sollevamento della pinna, mentre S_{fin} è la superficie della forma planare della pinna. Nelle parentesi si trovano x_{fin} cioè la posizione assiale lungo l'asse x della pinna, mentre δ_{s1} e δ_{s2} risultano essere gli angoli della pinna riferiti allo scafo del veicolo.

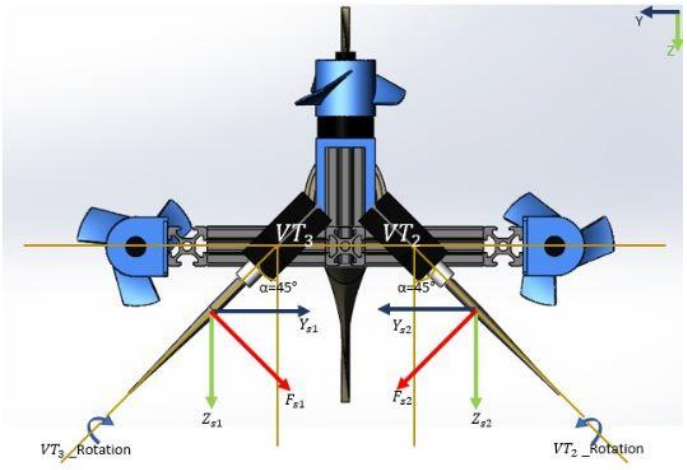

Figura 4.1.3: rappresentazione posteriore del motore e delle forze in gioco

L'ultimo contributo che viene analizzato è quello relativo alla spinta delle eliche e che può essere scritto come nell'eq. (10):

$$Z_{prop} = T_{|n_3|n_3}|n_3|n_3 - T_{|n_3|u_{a3}}|n_3|u_{a3} \tag{10}$$

dove n_3 è la velocità di rotazione dell'elica del motore centrale che verrà presa in ingresso da Simulink, mentre u_{a_3} è la velocità in ambiente acquoso ed espressa come nell'eq. (11) seguente:

$$u_{a_3} = (1 - \omega)u \tag{11}$$

dove ω è il wake fraction number compreso fra 0 e 1 (spesso fra 0.1 e 0.4), mentre con u viene indicato il modulo della velocità lineare e quindi, nel caso in esame, semplicemente w. Il modello di seguito presentato assume di avere forza gravitazionale e spinta archimedea bilanciate e, per tal motivo, i loro contributi non sono stati riportati nell'eq. (1).

Parametri	**Valori**
$massa\ del\ pesce$	6 kg
ω	0.2
ρ	1000 kg/m
c_L	2 n/a
S_{fin}	0.004 m^2
x_{fin}	0.09 m
$Z_{\|w\|w}$	$\rho A_t R^2 c_{d,t}/2$
A_t	$2\pi RL$
R	0.06 m
L	0.8 m
$c_{d,t}$	0.9
$T_{\|n_3\|n_3}$	1.2 N/rpm
$T_{\|n_3\|u_{a3}}$	20 N/rpm

Tabella 1: valore dei diversi coefficienti fisici

4.2 Rappresentazione del pesce-robot in Adams

Partendo dall'eq. (1) sono stati applicati sei contributi di forze al pesce nel suo centro di massa geometrico: la spiegazione del fatto che venga preso come valore di riferimento il centro di massa geometrico sta nel fatto che la massa del robot è quasi totalmente dovuta alla massa delle batterie dei tre motori, quindi non risulta essere completamente fuorviante considerare il nostro pesce di densità uniforme. A questo punto tramite comando "Create a Force" in Adams, contenuto nella sezione "Forces", sono stati aggiunti tutti i contributi come mostrato in *figura 4.2.1* in cui si nota come l'effetto della forza gravitazionale non sia stato preso in considerazione per i motivi citati in precedenza. Per il collegamento con Simulink è stato necessario dichiarare delle variabili tramite apposita sezione (cfr Cap. 1): come variabile di ingresso al sistema è stata considerata la velocità di rotazione dell'elica del motore centrale, mentre come variabili di uscita sono state prese la misura della profondità del centro di massa del robot (cfr Cap. 1) e l'ingresso al sistema, come valutazione della bontà dell'ingresso. Una volta generati i file per il collegamento a Simulink, come già riportato nel capitolo 3, ha inizio la procedura d'identificazione.

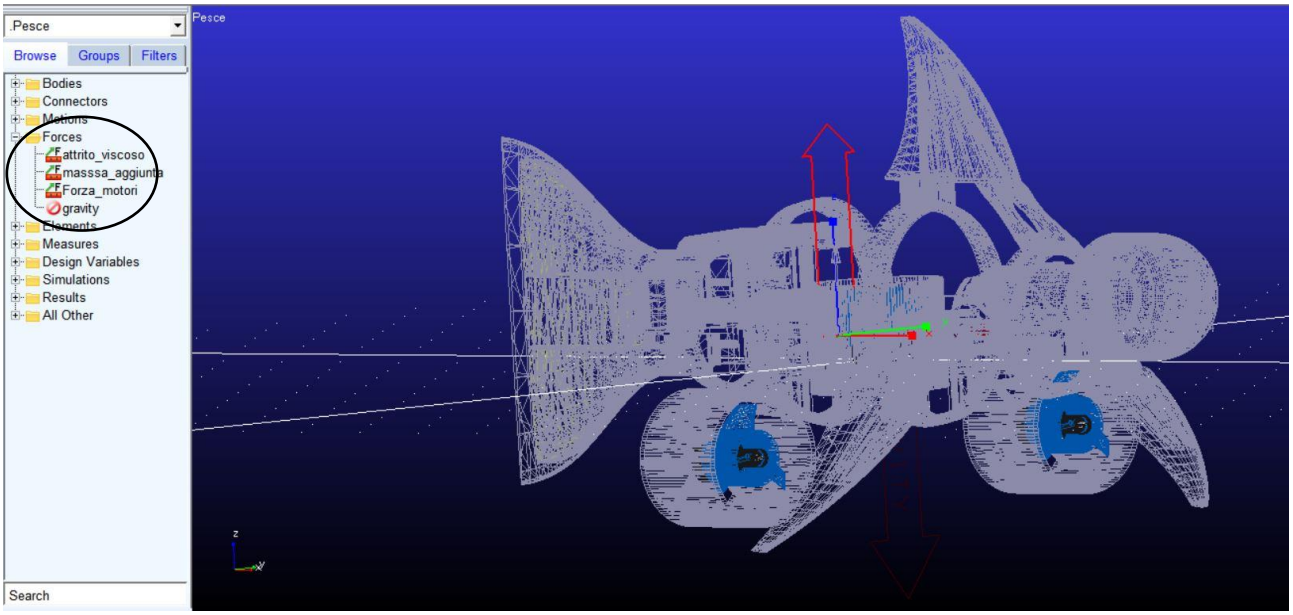

Figura 4.2.1: vista delle forze che governano il movimento del pesce; da notare il fatto che la forza di gravità sia disattivata.

5. L'identificazione del pesce-robot

In questo capitolo verranno affrontate quelle che sono le conoscenze necessarie allo sviluppo del modello per l'identificazione del pesce e poi i passi e i risultati conseguiti. Il problema dell'identificazione è quello di determinare un modello matematico, quindi un insieme di rigorose formule matematiche che spieghino adeguatamente l'andamento osservato dei dati. Questi dati risultano essere delle osservazioni nell'intervallo di tempo [0,T]e cioè misure riferite a valori assunti da alcune variabili negli istanti 0,1,…,T. In un primo momento il compito quindi è stato quello di descrivere il comportamento del sistema reale tramite un modello e poi in un secondo momento attraverso quest'ultimo avere la possibilità di prevedere il comportamento reale con la possibilità anche di effettuarne un controllo.

5.1 La procedura dell'identificazione

L'approccio adoperato è quello a scatola nera (Black Box) cioè si presuppone di non conoscere nulla del sistema, quindi senza la possibilità di aprirlo, cioè astraendolo dal contesto fisico in cui è immerso e identificare la scatola tramite le coppie ingresso-uscita. L'identificazione può essere divisa in quattro passi: trovare delle buone pratiche per la raccolta dati, individuare una famiglia di modelli che possano descrivere il sistema da identificare, scegliere il miglior modello e infine verificarlo. Il primo passo è quindi la raccolta dei dati che dovrà partire da un ingresso persistentemente eccitante cioè che stimoli completamente tutto il sistema in esame e quindi tutti i suoi modi naturali: chiaramente si avrà la possibilità di controllare soltanto i modi eccitabili in ingresso ed osservabili in uscita cioè si riuscirà a risalire alla sola funzione di trasferimento. Il segnale che viene usato è un rumore bianco cioè un segnale del tutto casuale, non periodico, con componenti su tutto lo spettro di frequenze; il rumore bianco nella pratica non esiste a causa dell'impossibilità di generare uno spettro uniforme, ma è possibile averlo a banda finita o limitata come poi verrà visto. Tale segnale è totalmente impredicibile e ogni suo valore è indipendente dal precedente: si può considerare che i campioni siano una sequenza di variabili casuali non correlate, con media nulla e varianza unitaria. Molto interessante circa lo studio che verrà effettuato in seguito è che, dopo aver effettuato le previsioni di un modello, l'errore di previsione deve essere bianco cioè essere totalmente impredicibile: questo indica che se l'errore non fosse bianco ci potrebbero essere ancora alcune parti da modellare. Per completezza si ricorda che l'errore di previsione o errore di modello è la differenza tra l'uscita reale del sistema e l'uscita del modello costruito. Nel paragrafo di sviluppo del progetto verranno riportati quelli che sono i

passaggi per la formazione di un rumore bianco in Matlab, per intanto viene affrontato il quesito di come verificare che un segnale sia bianco. Vengono riportate qui di seguito le condizioni sottoforma di formule che identificano un rumore w come bianco:

$$\mu_w = E\{w\} = 0 \tag{12}$$

$$R_{ww} = E\{ww^T\} = \sigma^2 I \tag{13}$$

Occorre riprendere anche il concetto di funzione di correlazione; essa è definita tale quando fornisce la misura di quanto due segnali abbiano proprietà comuni. Per tal motivo si parlerà di funzione di autocorrelazione quando una funzione fornisce una misura di quanto un segnale si assomigli e abbia proprietà comuni con se stesso ritardate di un tempo tau. Proprio un confronto tra un intervallo di un segnale e un altro intervallo della medesima ampiezza è alla base del test numerico di bianchezza di Anderson. Di seguito verrà presentato in linea teorica il test perché sarà questo utilizzato come valutazione del rumore d'ingresso e dell'errore di previsione. Dato un valore di confidenza α che rappresenta l'area della gaussiana di validazione del test di bianchezza, da questo si ricava il valore β, quindi si va a valutare il numero dei valori della varianza campionaria normalizzata $\rho(\tau)$ che cadono all'esterno dell'intervallo $[-\beta\ \beta\ ]$ escludendo il valore in 0. Sia c il numero di tali valori che cadono fuori dall'intervallo allora si accetta l'ipotesi di bianchezza di un segnale se c/N < α. Viene considerata significativa da Matlab la correlazione tra un campione e quello seguente fino a 20° di ritardo quindi con N=20 e con τ che appartiene all'intervallo $[0\ N]$. L'implementazione del codice del test di Anderson in Matlab viene riportata nel paragrafo successivo a questo. Una volta completata la raccolta dei dati si passa ad una stima del modello $M(\theta)$ caratterizzato da un opportuno vettore dei parametri θ. La natura dei modelli da identificare può essere ricondotta o ad un sistema di controllo o ad una serie temporale: la differenza sta nel fatto che mentre nel primo caso si ha un ingresso e un disturbo agenti sul sistema, nel secondo caso non si hanno ingressi, ma soltanto disturbi che vanno ad agire sulla scatola(*figura 5.1.1* e *figura 5.1.2*): cambia quindi anche lo scopo che nel primo caso sarà effettuare un controllo mentre nel secondo sarà effettuare una previsione. Verranno presi in considerazione i modelli a errore di equazione descritti dalla *figura 5.1.1* ed in particolare verranno qui di seguito approfondite le famiglie ARX e ARMAX poiché saranno queste due famiglie

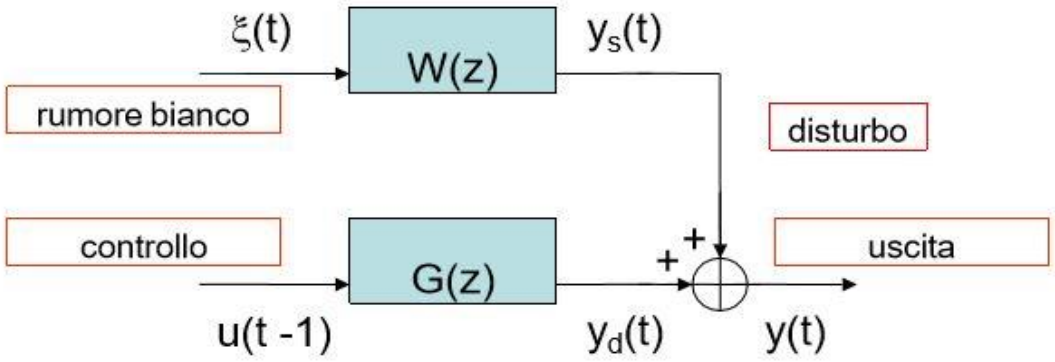

Figura 5.1.1: struttura generale di un sistema di controllo

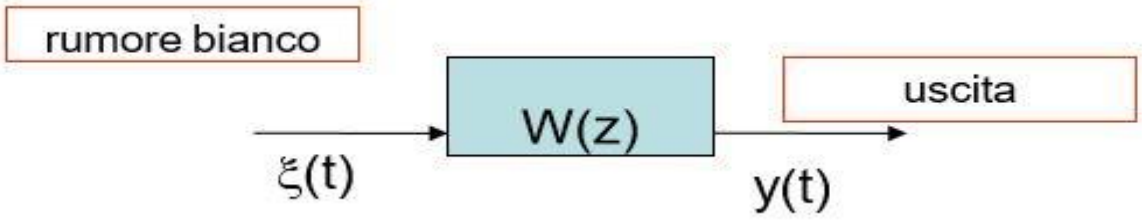

Figura 5.1.12: struttura generale di una serie temporale

ad essere utilizzate nell'identificazione. Per queste famiglie possiamo scrivere che l'uscita $y(t)$ è data dalle eq. (14) e eq. (15):

$$y(t) = G(z)u(t-1) + W(z)\,\xi(t) \tag{14}$$

$$\begin{aligned} y(t) = {} & a_1 y(t-1) + a_2 y(t-2) + \cdots + a_{na} y(t-na) + \\ & + b_1 u(t-1) + b_2 u(t-2) + \cdots + b_{nb} u(t-nb) + \\ & + w(t) \end{aligned} \tag{15}$$

Si può notare come l'uscita sia somma di tre contributi: una parte autoregressiva che fa dipendere l'uscita all'istante t da una storicità delle uscita come se il nostro sistema avesse una memoria fino all'uscita na-esima; la seconda parte fa dipendere l'uscita da una storicità di ingressi con una memoria di ordine nb; l'ultima parte viene detta residuo d'equazione. È interessante constatare che il nostro sistema evolva grazie all'ingresso u(t-1) precedente rispetto all'uscita y(t) questo perché vi è un tempo di elaborazione che va preso in considerazione e che viene introdotto come ritardo nell' eq. (14). I modelli ARX e ARMAX si distinguono per come le rispettive parti che sono formate nel seguente modo: nei modelli ARX il residuo di equazione w(t) risulta essere un rumore bianco quindi $w(t) = \xi(t)$. Ponendo

$$A(z) = 1 - a_1 z^{-1} - a_2 z^{-2} - \cdots - a_{na} z^{-na} \tag{16}$$

$$B(z) = b_1 + b_2 z^{-1} + \cdots + b_{nb} z^{-nb+1} \tag{17}$$

Si arriva a dimostrare che

$$\mathrm{y(t)} = \mathrm{B(z)/A(z)u(t-1)} + 1/\mathrm{A(z)}\,\xi\,\mathrm{(t)} \tag{18}$$

ove dalla (16) e (17) si denotano con na e nb rispettivamente gli ordini della parte autoregressiva e della parte esogena. Quindi un modello ARX risulta completamente descritto dal vettore dei parametri $\theta = [a_1, a_2, \dots, a_{na}, b_1, b_2, \dots, b_{nb}]^T$ e viene indicato con ARX(na,nb). Risulta d'interesse anche andare a definire il ritardo k del modello come il primo coefficiente b_k diverso da zero. Il modello ARMAX ha rispetto all'ARX un residuo d'equazione pari a

$$\mathrm{w(t)} = \xi\,\mathrm{(t)} + c_1\xi\,\mathrm{(t-1)} + c_2\xi\,\mathrm{(t-2)} + \cdots + c_{nc}\xi\,\mathrm{(t-nc)} \tag{19}$$

ove il residuo viene detto media mobile e risulta essere la combinazione lineare dei valori campionati di un rumore bianco sull'intervallo $[\mathrm{t-nc, t}]$. Chiaramente la (19) trasforma l'eq.15 nella eq. (15.1).

$$\begin{aligned} y(t) = a_1 y(t-1) + \ a_2 y(t-2) + \cdots + a_{na} y(t-na) + \\ + b_1 u(t-1) + b_2 u(t-2) + \cdots + b_{nb} u(t-nb) + \\ + \xi\,(t) + c_1\xi\,(t-1) + c_2\xi\,(t-2) + \cdots + c_{nc}\xi\,(t-nc) \end{aligned} \tag{15.1}$$

Sviluppando i calcoli come fatto in precedenza si ha che:

$$\mathrm{A(z)} = 1 - a_1 z^{-1} - a_2 z^{-2} - \cdots - a_{na} z^{-na} \tag{20}$$

$$\mathrm{B(z)} = b_1 + b_2 z^{-1} + \cdots + b_{nb} z^{-nb+1} \tag{21}$$

$$\mathrm{C(z)} = 1 + c_1 z^{-1} + c_2 z^{-2} + \cdots + c_{nc} z^{-nc} \tag{22}$$

Che portano al risultato dell'eq. (23)

$$y(t) = B(z)/A(z)u(t-1) + C(z)/A(z)\,\xi\,(t) \tag{23}$$

Quindi, considerano la (15.1), si hanno na,nb e nc che sono gli ordini rispettivamente della parte autoregressiva, della parte esogena e della parte a media mobile. Il vettore dei parametri diventerà dunque della forma $\theta = [a_1, a_2, \dots, a_{na}, b_1, b_2, \dots, b_{nb}, c_1, c_2, \dots, c_{nc}]^T$ e il modella verrà indicato come ARMAX(na,nb,nc). Anche in questo modello il ritardo k è il primo coefficiente non nullo della parte esogena. Il modello scelto farà parte di una delle famiglie presentate; occorre però costruire un predittore che si comporti come il modello del sistema reale attraverso dei criteri di scelta qui di seguito riportati. La definizione di predittore è un sistema $\widehat{M}(\theta)$ che elaborando i dati i/o del modello $M(\theta)$ sull'intervallo $[0, t-1]$ fornisce una predizione del valore di uscita $\hat{y}(t)$. Questa uscita essendo deterministica può essere paragonata all'uscita del sistema reale e dalla loro comparazione viene ricavato l'errore di predizione $\varepsilon(t)$: tale valore sarà indice dell'accuratezza con la quale si è predetto il comportamento del modello reale. Per trovare il migliore modello tra tutti quelli della famiglia M occorre trovare il punto di flesso della funzione "errore quadratico medio" espressa come $J_\theta = \frac{1}{N}\sum_1^N \varepsilon^2(t)$: si troverà un valore θ per cui J_θ è minima e il modello corrispondente sarà quello da scegliere. Essendo

$$y(t) = \left[1 - \frac{1}{W(z)}\right] y(t) + \frac{G(z)}{W(z)} u(t-1) + \xi\,(t) \tag{24}$$

e considerando la sua miglior predizione

$$\hat{y}(t) = \left[1 - \frac{1}{W(z)}\right] y(t) + \frac{G(z)}{W(z)} u(t-1) \tag{25}$$

risulta chiaro il motivo di quanto espresso in precedenza e cioè che l'errore che si dovrà attendere in uscita dovrà essere bianco poiché $\varepsilon(t) = y(t) - \hat{y}(t)$. Risulta interessante andare a vedere come data una matrice nxn contenente i valori degli ingressi e delle uscite essa avrà le colonne linearmente indipendenti fino ad un certo punto per poi avere colonne invece linearmente dipendenti: questo perché fino ad una certa colonna si avrà che questa aggiunge informazioni nuove rispetto alle precedenti mentre da un certo punto in poi non si avranno più informazioni

utili, o meglio si avranno ma non verranno più considerate perché minime e non significative. Questa dipendenza provoca la singolarità della matrice da cui ne deriva che almeno un autovalore abbia valore nullo: da questa caratteristica si intuisce che aumentando il numero delle colonne prese in considerazione il valore del più piccolo autovalore in valore assoluto tenda a zero. Quindi una misura di quest'ultimo fornisce una misura dell'aderenza ai dati: maggiore sarà l'ordine scelto maggiore sarà l'aderenza ai dati. Ciò provoca però due problemi: un primo derivante dal fatto che una adesione ai dati non è sempre buona considerando che dentro la modellazione entrano errori di misura e linearizzazioni che incidendo sull'uscita vanno ad incidere sul modello del sistema (si sta modellando non solo il sistema reale ma anche i disturbi presenti, situazione di sovraparametrizzazione); come secondo problema si ha che aumentando l'ordine del sistema si va ad aumentare la complessità del sistema: occorrerà dunque mediare tra una buona complessità ed una buona adesione ai dati. Altra cosa da prendere in considerazione è la valutazione del numero N dei valori raccolti presente nella formula dell'errore quadratico medio: come si nota tale valore è dipendente da N e varia con esso. Si può quindi concludere che al variare di N vari anche θ ed in particolare l'insieme dei valore che θ può assumere. È possibile dimostrare che per N che tende ad infinito θ tende al valore ottimo θ_{min} , ma ovviamente è impossibile avere una raccolta infinita di dati: come si vedrà in seguito oltre a trovare un valore per i parametri del modello, questi saranno soggetti a varianza che dovrà poi essere presa in considerazione per arrivare a discutere la bontà del modello. Si entra quindi nell'ultima parte dell'identificazione cioè la validazione: nel caso in esame che verrà a breve introdotto viene usato il metodo della cross validazione. Tale metodo utilizza una divisione dei dati in due sotto-raccolte in modo tale da avere una prima raccolta che sarà quella su cui verrà effettuata l'identificazione per poi andare a verificare la bontà del modello tramite un paragone con la seconda raccolta, in particolare valutando la bianchezza dell'errore di predizione. È interessante vedere come l'analisi della bianchezza fornisca un criterio di limitazione verso il basso dell'ordine del sistema, permettendo di capire quanto semplice sia il modello (in tal caso si avrebbe un rumore poco bianco); la tecnica della cross validazione invece ci consente di limitare in alto la scelta dell'ordine del nostro modello: una alta adesione ai dati risulta come detto in precedenza fuorviante per effetto della sovraparametrizzazione e questo è ben visibile nel paragone con altri dati che evidenzia sostanzialmente quanto si sia modellato il sistema piuttosto che il sistema con l'aggiunta dei rumori. La validazione verrà affrontata seguendo questi tre passi: una valutazione della bianchezza dell'errore di predizione, una valutazione della varianza dei parametri che ne esprime l'incertezza e infine la cross validazione per vedere l'adesione al

modello. Oltre alla cross validazione si hanno altri parametri detti indici di validazione che definiscono un criterio di bontà del modello: questi sono FPE(Final Prediction Error), AIC(Akaike Information Criterium) e MDL(Minimum Description Lenght). Vengono qui brevemente presentati perché utilizzati in seguito come criterio di scelta per un possibile modello ARX. Tutti i modelli penalizzano la complessità del modello a fronte dell'aderenza ai dati come viene mostrato nelle eq.

$$FPE = \frac{N+n}{N-n} J_N(\theta) \tag{26}$$

$$AIC = \frac{2}{N} n + \log J_N(\theta) \tag{27}$$

$$MDL = \frac{\log N}{N} n + \log J_N(\theta) \tag{28}$$

Come già affrontato occorre puntualizzare che si preferisce un modello semplice ad uno complesso anche se quest'ultimo risultasse più aderente ai dati; chiaramente la modellazione non viene fatta solo del sistema ma anche di tutti quegli strumenti utilizzati per la raccolta dati e per le misure o il lavoro svolto; infine come si avrà modo di vedere occorre privilegiare un modello che aderisca bene in una ristretta banda di frequenza ed in caso utilizzare più modelli per ampie bande.

5.2 Applicazione della procedura d'identificazione del pesce-robot

A partire da quello espresso nei precedenti capitoli si cercherà ora di analizzare il lavoro svolto. Per prima cosa si è proceduto alla raccolta dati stimolando il sistema pesce tramite Simulink: in ingresso al blocco Adams del robot in Simulink è stato posto come da teoria un rumore bianco (*figura 5.2.1*). Tale rumore, come mostrato dalla *figura 5.2.2*, è costruito in Matlab generando casualmente i parametri e andando a effettuare un controllo sulla bianchezza tramite algoritmo di Anderson racchiuso dentro il ciclo "for": il ciclo "while" esterno viene utilizzato come controllo tramite una variabile di "flag" indicata con s. Se il valore di tale variabile risulterà pari a 1 allora il rumore generato sarà di tipo bianco, mentre in caso contrario si dovrà andare a rigenerare un vettore casuale di valori. Al rumore bianco generato si va inoltre a sottrarre la media in modo tale che il segnale abbia come da definizione di rumore bianco media nulla.

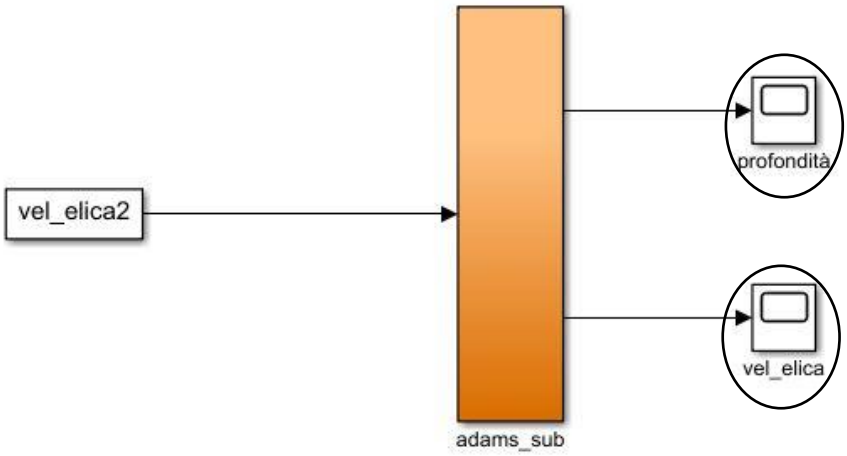

Figura 5.2.1: rappresentazione modello pesce in Simulink

```
s=0;
c=0;
while(s~=1)
    rumore_casuale=rand(1,n);
    rumore_casuale=rumore_casuale-mean(rumore_casuale);
gamma0=1/N*rumore_casuale(1:N)*rumore_casuale(1:N)';
for tau=1:N
    gamma(tau)=1/N*rumore_casuale(1:N-tau)*rumore_casuale(1+tau:N)';
    ro(tau)=gamma(tau)/gamma0;
    if abs(ro(tau))>beta
    c=c+1;
    end
end

if c/N<alfa
    s=1;
else
    s=0;
end

end
```

Figura 5.2.2: codice generazione rumore bianco

Si trova poi l'implementazione del test di Anderson: viene definita una variabile "gamma0" e un valore "gamma(tau)" dal cui rapporto, come mostrato, viene ricavato "ro(tau)". A questo punto secondo la condizione dentro "if" si decide se aumentare la variabile "c" che è il contatore dei valori uscenti dall'intervallo $[-\beta\ \beta]$ già discusso in precedenza. Allora, nell'ultimo "if", viene valutata la condizione di bianchezza che permette o meno di continuare l'esecuzione. I valori generati dalla funzione "rand()" in Matlab sono valori compresi tra $(0,1)$ e chiaramente non sono sufficienti per controllare la velocità dell'elica del motore perché da una parte non si

```
uscita=out.ADAMS_yout1;
uscita2=uscita([1:end-1],1);
ingresso2=uscita([1:end-1],2);
```

Figura 5.2.3: codice per l'importazione dei dati salvati da Simulink nel Workspace

apprezzerebbe una variazione nell'uscita e dall'altra si deve stimolare tutto il sistema(si vada a notare che avendo eliminato la media del segnale i valori varieranno nell'intervallo $(0.5,0.5)$. Per una ragione di accuratezza e di verifica a posteriori, sono stati introdotti due segnali bianchi diversi: uno che stimolasse il motore a metà della potenza effettiva quindi con una rotazione pari a $200\ rad/s$(quindi moltiplicando il rumore per 400), ed un segnale che facesse girare al massimo le eliche, moltiplicando quindi il rumore per 800(la velocità massima dell'elica è di $400\ rad/s$). Nella trattazione presente verrà considerato solo il rumore in ingresso moltiplicato per 400 e poi solo nella fase dell'identificazione verrà ripreso anche l'altro ingresso. Come introdotto nei capitoli relativi a Matlab e Simulink, quando andiamo a eseguire un modello in Simulink il blocco "scope" presente(si vedono cerchiati nella *figura 5.2.1*) esporta i dati in uscita nel Workspace di Matlab. Si dovrà quindi andare ad importare i dati che sono stati salvati e questo è fatto tramite un accesso ad una struttura dati che è "out" come si vede dalla *figura 5.2.3*. A questo punto dopo aver importato i dati di uscita e avendo quelli di ingresso possiamo andare a iniziare l'identificazione avendo creato le coppie ingresso uscita relative al sistema in esame: avremo come già spiegato in ingresso un rumore bianco che sarà indice della velocità di rotazione dell'elica centrale e in uscita viene valutata la profondità raggiunta dal pesce a partire da una situazione di equilibrio. Prima però occorre presentare il metodo di temporizzazione: si è deciso infatti di controllare il motore con un segnale che varia ogni 0.01s poiché tutto il sistema lavora in tempo discreto. Quindi anche i valori delle uscite riportate che sono state valutate in un intervallo di 100s sono valori presi ad istanti di tempo intervallati di 0.01s. Non viene qui di seguito riportato ma occorre precisare che bisogna valutare accuratamente l'intervallo di temporizzazione per stimolare bene il sistema e questo viene fatto tramite un ingresso a scalino e calcolando il tempo di salita del nostro sistema: da questa misura e applicando il teorema di Shannon si arriva ad una valutazione del tempo di campionamento. Utilizzando il toolbox "systemidentification" da prompt dei comandi di Matlab e importando le coppie ingresso e uscita si vuole ora visualizzare quello che è l'andamento grafico dei due segnali: nella *figura 5.2.4* si notano due segnali di colori diversi che ne formano uno; come già affrontato ed esposto si andrà a lavorare con metà dei valori(500 coppie) e si lascerà l'altra

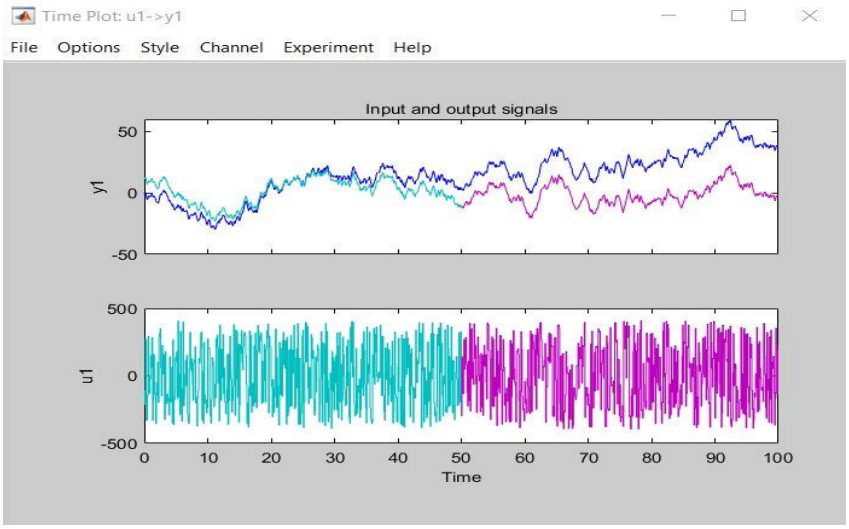

Figura 5.2.4: segnale iniziale e divisione dei segnali

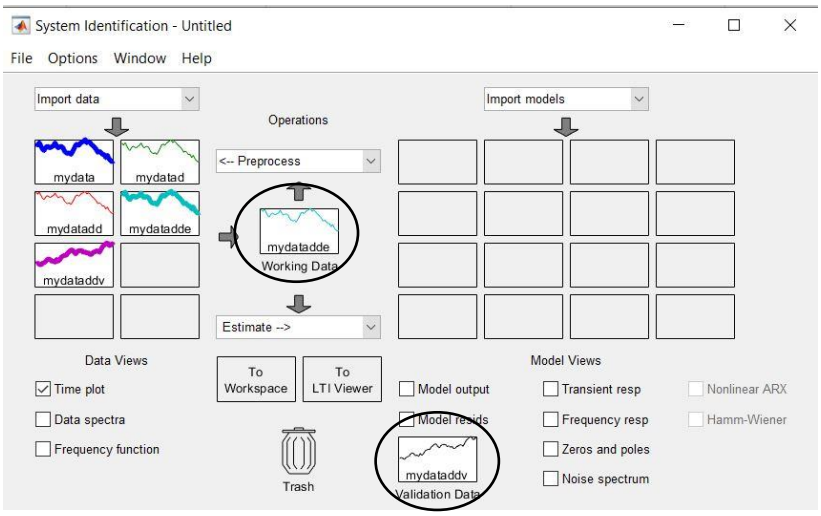

Figura 5.2.5: finestra di Matlab per l'identificazione

metà per la cross validazione finale: si effettuerà quindi l'identificazione solo della prima parte del segnale trascinandola nel "working data" indicato nel cerchio in *figura 5.2.5*, mentre la seconda parte verrà trascinata come da figura nella parte della "validation data". Nella *figura 5.2.4* si può notare come in realtà il segnale generato dalla composizione dei segnali azzurro e viola sia diverso dal segnale importato inizialmente e corrispondente alla linea blu, questo perché sono state applicate due trasformazioni del segnale per renderlo più correttamente identificabile: una prima operazione è stata quella di togliere la media al segnale di ingresso per averlo a media nulla come si nota in figura ed una seconda operazione è stata quella di togliergli i trend cioè disturbi a bassa frequenza spesso periodici. Queste operazioni portano ad un segnale che risulta leggermente

diverso da quello in ingresso. Va notato inoltre come detto che il segnale in ingresso ha ampiezza massima pari a 400 cioè alla velocità massima di rotazione delle eliche. Prima di vedere quali sono le possibili famiglie di modelli si va a controllare il diagramma di Bode del sistema per ricavarne alcune considerazioni sulla banda di frequenza in cui si pensa possa lavorare il sistema: guardando la figura 5.2.6 si nota come verrà preso in considerazione un modello che rispecchi approssimativamente il comportamento in frequenza fino a 5-7 Hz. Si passa dunque alla seconda parte dell'identificazione e cioè quella della valutazione delle possibili famiglie dei modelli: partendo dagli ARX, Matlab permette di stimare dei modelli fisicamente possibili andando a vedere tutte le possibili combinazioni dei parametri (na,nb,k) in un intervallo che va da $[1:10]$ chiaramente per ogni valore. Dalla stima quello che esce fuori è un istogramma come quello mostrato in *figura 5.2.7* in cui si può notare che con colori diversi sono stati riportati, come da legenda, i modelli che rappresentano un miglior indice MDL, AIC e una miglior corrispondenza con i dati il modello evidenziato in rosso. Come già espresso il metodo di scelta del modello deve tenere sicuramente in conto la corrispondenza con il segnale, ma non deve essere neanche troppo complesso. I modelli importati per essere poi valutati sono stati i tre appena elencati e come da teoria quei modelli che hanno un "salto" maggiore (per sicurezza sono stati presi anche altri modelli poiché il salto è quasi costante per tutti). Per prima cosa si è iniziata a valutare la posizione degli zeri e dei poli con la relativa incertezza dovuta alla varianza dei parametri dei modelli ARX: sono stati scartati tutti quei sistemi in cui i poli non ricadessero all'interno dei cerchio di raggio unitario oppure che abbiano considerando anche la varianza dei valori corrispondenti con alcuni zeri, questo perché una sovrapposizione significa in molti casi una cancellazione e quindi una stima troppo complessa dei parametri. In breve si sta capendo che il sistema dovrebbe essere più semplice rispetto a quello preso in esame: verrà di seguito riportato un esempio nella *figura 5.2.8* dell'analisi di quanto spiegato su un solo modello, ma chiaramente è un' analisi che va svolta per ogni modello eliminando quelli che non rispettino le condizioni esposte. A questo punto si va a valutare quella che è la risposta in frequenza del modello che si sta identificando confrontandola con quella espressa nella *figura 5.2.6*: bisogna osservare se nella banda presa in considerazione (nel caso in esame fino a 5-7 Hz) il comportamento sia paragonabile.

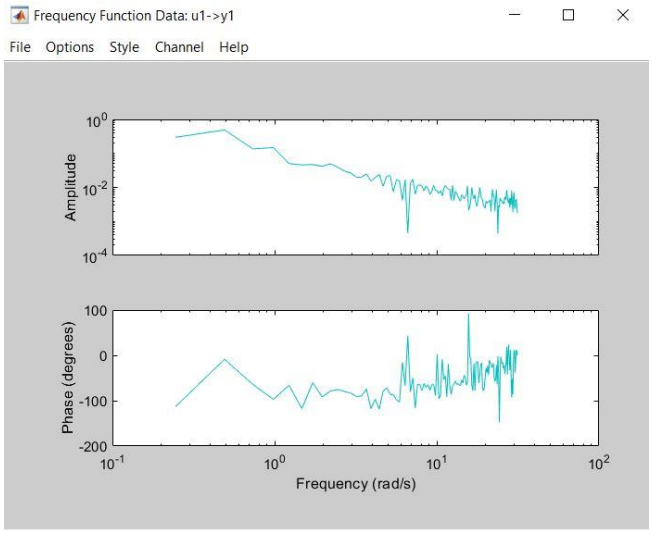

Figura 5.2.6: diagramma di Bode del sistema formato dalla prima metà dei valori

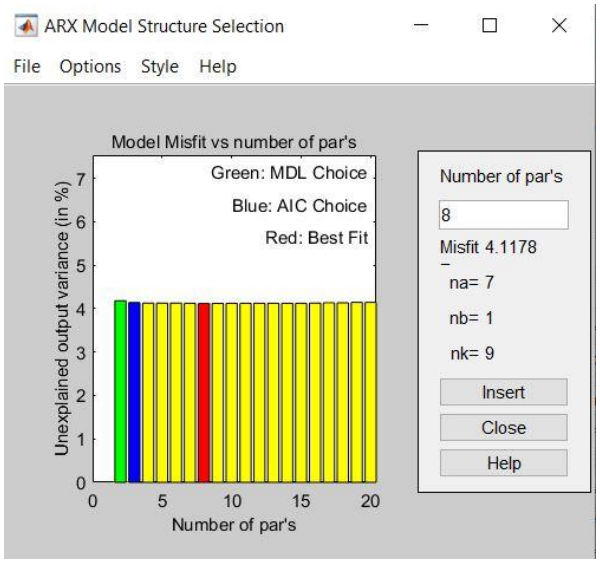

Figura 5.2.7: istogramma dei possibili modelli ARX

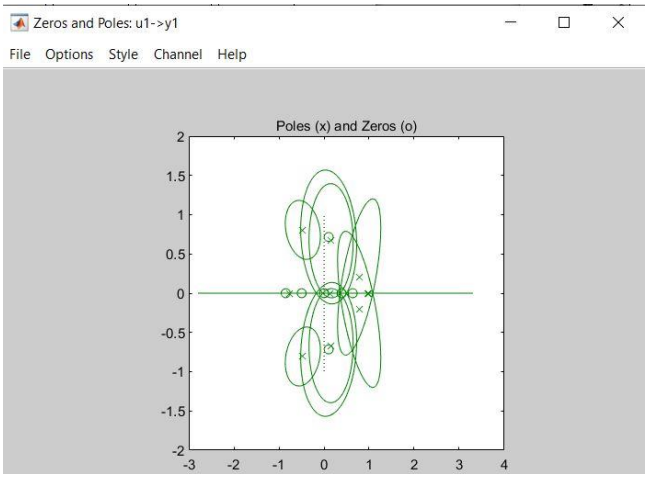

Figura 5.2.8: poli e zeri di un sistema complesso

A questo punto si arriva alla parte conclusiva ove si vede se il modello che si sta valutando generi come residuo un rumore bianco grazie al test di Anderson (N=20 preimpostato da Matlab e quindi c=0 con grado di confidenza del 99%), andando a contare i valori uscenti dall'intervallo di confidenza e una volta presi i modelli che rispettano tutte queste condizioni si va a scegliere il modello che ha una migliore adesione al segnale della validazione(si ricorda infatti che la validazione viene fatta sul secondo gruppo di campioni dei dati). Studiando gli ARX come ci si può aspettare dovendo gestire un modello non lineare si giunge a valutare dei modelli la cui adesione ai dati è quasi nulla: si iniziano allora a valutare quegli ARMAX di ordine simile agli ARX visti. Qua di seguito vengono riportati i grafici, che descrivo le proprietà elencate in precedenza valide anche per i modelli ARMAX, del modello trovato e che meglio rappresenti il modello del pesce in esame: tale modello risulta essere armax(3 1 6 0) (armax sta per ARMAX).

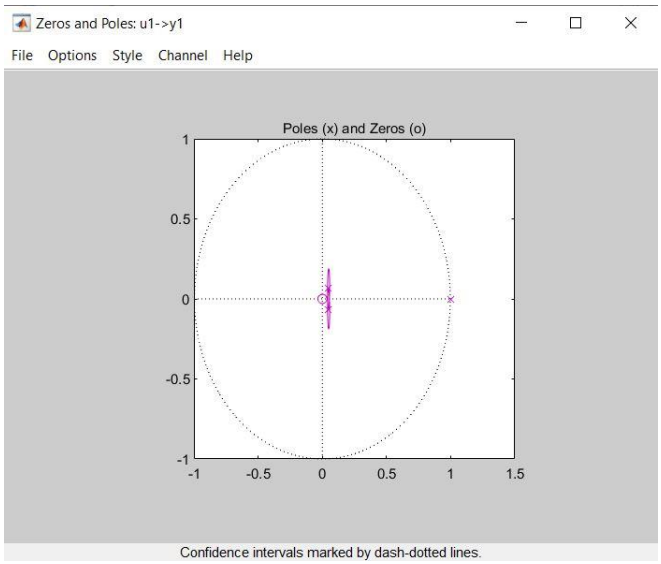

Figura 5.2.9: poli e zeri del modello armax(3 1 6 0) con relativa circonferenza unitaria

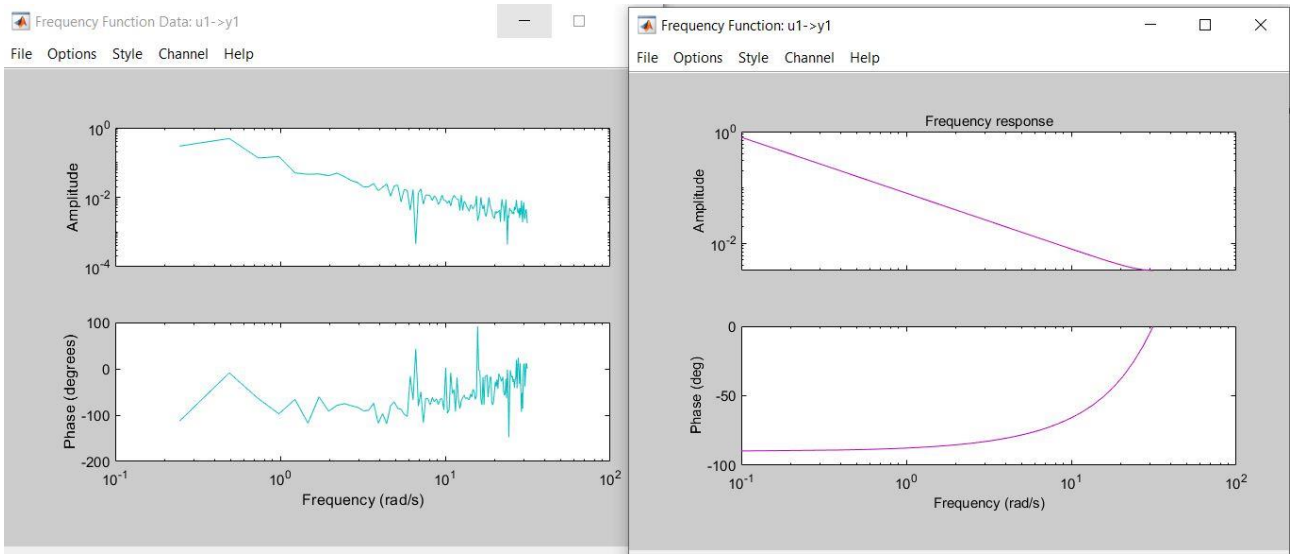

Figura 5.2.10: confronto tra le risposte in frequenza due modelli: reale in azzurro e identificato in viola

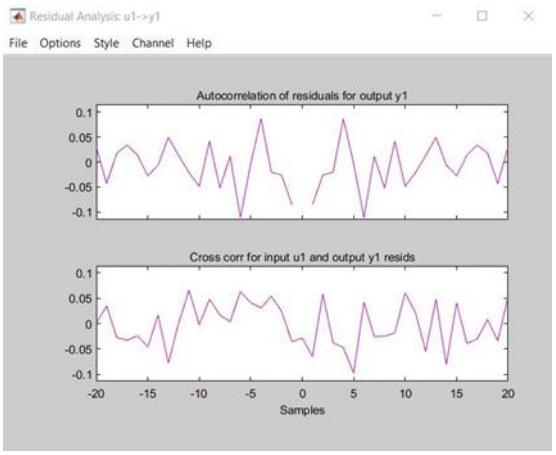

Figura 5.2.11: valutazione residui d'equazione con test di Anderson

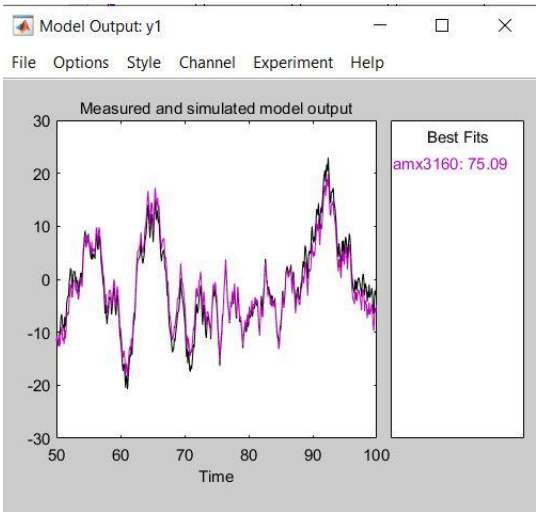

Figura 5.2.12: confronto tra il segnale di uscita del sistema pesce e quello del modello armax(3 1 6 0)

Quelli riportati nelle *figure dalla 5.2.9 alla 5.2.12* sono i risultati riportati dalla modellazione del sistema pesce circa i quali brevemente verranno fatte alcune precisazioni: nella prima immagine il polo di destra risulta essere dentro il cerchio di raggio unitario quindi il modello risulta essere stabile; nella seconda immagine occorre ricordare di valutare l'andamento tramite l'ordine sull'asse delle ordinate e di valutare fino a circa 7 Hz l'andamento; il test di Anderson riportato in figura risulta essere verificato poiché nessun valore esce dai limiti consentiti; da ultimo il modello scelto risulta essere aderente ai dati e questo è possibile notarlo dalla *figura 5.2.12* ove in viola viene riportato l'andamento del modello mentre in nero quello del sistema reale. Importando il modello nel workspace di Matlab è possibile andare a vedere quelli che sono sia i valori dei singoli parametri: per completezza verranno riportati qui di seguito:

$$A(z) = 1 - 1.097\,z\text{^}-1 + 0.1037\,z\text{^}-2 - 0.006513\,z\text{^}-3$$

$$B(z) = 0.007187$$

$$C(z) = 1 - 0.004911\,z\text{^}-1 - 0.01251\,z\text{^}-2 + 0.01594\,z\text{^}-3 - 0.02215\,z\text{^}-4 - 0.003673\,z\text{^}-5 + 0.09182\,z\text{^}-6$$

6. Il controllo del pesce-robot

Si vuole presentare quella che è stata la tecnica e la modalità per la simulazione del controllo del pesce tramite ambiente Simulink. Con il termine “controllo” si intende la procedura con la quale si fornisce al pesce-robot la profondità da raggiungere in tempi accettabili: tale valore viene comunemente detto riferimento. Per affrontare il problema nella sua completezza viene riportato il progetto in Simulink in *figura 6.1* e *figura 6.2*: quello che verrà fatto sarà descrivere in maniera completa tutto il progetto riportando anche in *figura 6.3* il secondo file Matlab, presentato nei primi capitoli e necessario al controllo.

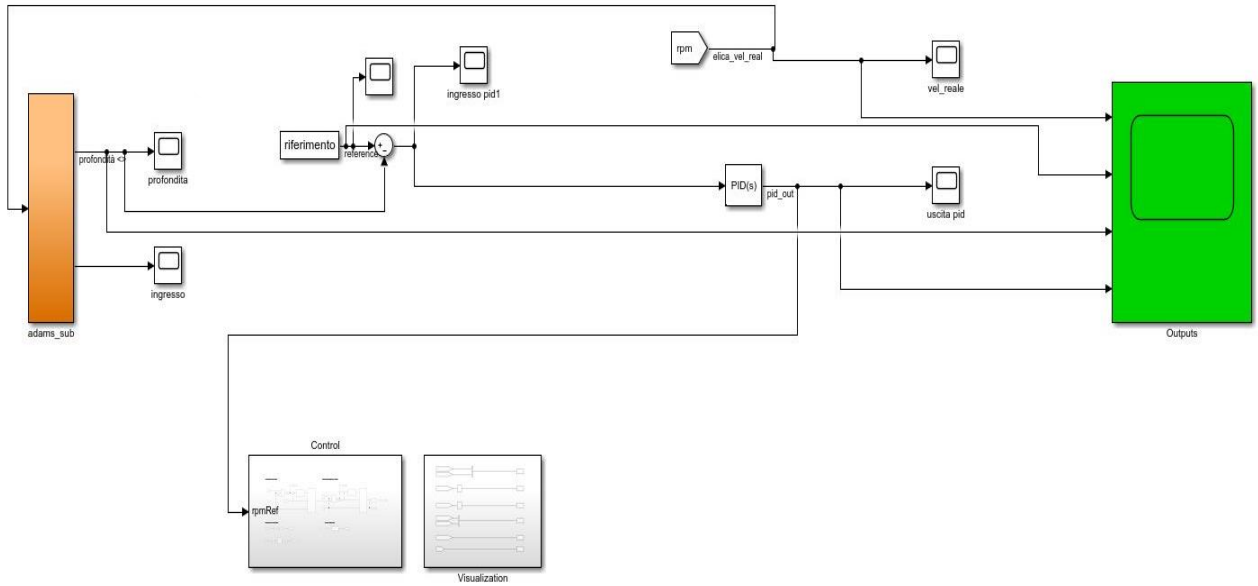

Figura 6.1: schema di controllo del robot-pesce in Simulink

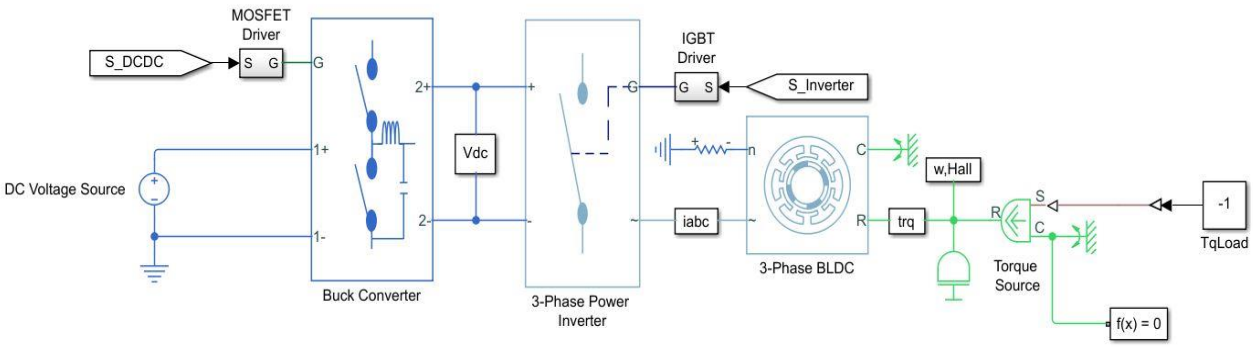

Figura 6.2: schema in Simulink di un motore brushless

Data una profondità impostata come riferimento, si vuole intervenire sul motore a seconda della differenza tra il valore di riferimento e la profondità che il robot ha raggiunto: tale differenza viene chiamata errore e, a seconda di come esso varia, si dovrà far girare più o meno velocemente il motore.

Come prima cosa si è modificato il progetto in Adams aggiungendo sia la forza peso sia la spinta di Archimede: rispetto al caso precedente, si lavorerà dunque con un modello leggermente differente che tenga conto di una dinamica che meglio si avvicina alla realtà in modo tale da verificare il controllo anche in situazioni reali e non solo a livello di simulazione.

Una volta modificato ed esportato il progetto in Simulink con la procedura già vista, si è creato un modello Simulink del motore brushless: quello trovato e rappresentato in *figura 6.2* è il modello che Simscape Electrical pone a disposizione tramite il comando Matlab "open_system('scdbldcspeedcontrol')". In realtà, tale modello possiede un controllo interno, avendo come parte superiore i due blocchi in basso evidenziati in *figura 6.1*: tutto il sistema complessivo riceve un valore in ingresso che è la velocità desiderata di rotazione dell'elica e tra le cose che elabora in output vi è la velocità reale del motore. Questo perché il motore, a seconda dei suoi parametri interni e caratteristiche costruttive, non gira propriamente alla velocità desiderata ma anch'esso con un errore che deve essere fatto tendere a zero nel minor tempo possibile: solo in questo caso, infatti, la velocità reale di rotazione del motore sarà quella desiderata in ingresso.

La velocità desiderata che deve assumere il motore logicamente dipende da quanto si è "distanti" dalla profondità di riferimento: se si è molto distanti il motore dovrà ruotare molto velocemente, chiaramente rispettando i parametri reali ma, a mano a mano che ci si avvicinerà al riferimento, la velocità del motore dovrà essere costante e tale da permettere l'equilibrio al robot. Si intuisce quindi che la velocità desiderata dipende dall'errore di profondità cioè dalla differenza tra riferimento e profondità attuale del pesce: in particolare il controllo viene gestito tramite blocco PID.

Il controllo PID cioè Proporzionale-Integrativo-Derivativo è un sistema in retroazione negativo utilizzato nei sistemi di controllo che prende in ingresso un errore e fornisce in uscita un valore che in questo caso è la velocità desiderata dell'elica. Il blocco del PID è riassumibile nel dominio di Laplace dall'eq. (29):

$$K(s) = K\left(\frac{\tau_I * s+1}{\tau_I * s}\right) * \left(\frac{\tau_D * s+1}{\alpha * \tau_D * s+1}\right) \quad (29)$$

Quello che deve essere impostato nel controllore PID sono le costanti integrative, derivative e proporzionali che vanno poi a modificare i valori τ dell'eq. 29 e quindi l'uscita del PID. Si può quindi andare ad incidere sulle diverse componenti ed ognuna di esse ha una conseguenza specifica sul controllo.

La prima componente è quella proporzionale la quale permette una variazione veloce del controllo ed anche una rapidità con la quale viene raggiunto il riferimento: il problema risulta essere che con solo un controllore proporzionale non si riesce a far tendere a zero l'errore.

Per ovviare a tale problema, vi è il controllo integrativo che tiene conto dell'errore passato in modo tale che, essendo un integrale dei valori degli errori passati, riesca a far tendere a zero l'errore. L'applicazione dell'integratore si nota sotto due aspetti. Il primo è relativo al fatto che esso smorza il comportamento ondulatorio che introduce il proporzionale; infatti, non riusciamo ad avere stabilità con il solo controllo proporzionale poiché avremmo il valore oscillante sul riferimento. Il secondo aspetto del comportamento dell'integratore è l'andamento a regime che non risulta mai essere nullo anche se è nullo il valore in ingresso, e quindi l'errore: questo perché l'integratore integra appunto l'errore e quindi tiene conto anche dei valori precedenti e non nulli.

Per ovviare a questa minima variazione a regime, si utilizza il derivativo che come dice il nome fa dipendere il suo contributo dalla velocità di variazione dell'errore. Questo è utile per il motivo già presentato ma bisogna prestare attenzione poiché una variazione molto rapida del riferimento potrebbe far tendere all'infinito il contributo del derivativo creando quindi problemi al controllo.

Nel lavoro svolto i valori delle costanti dei singoli parametri sono riportati nell'eq. (30-32):

$$(P)K = 50 \quad (30)$$

$$(I)K_I = 1 \quad (31)$$

$$(D)K_D = 0.1 \quad (32)$$

Ove vale la relazione rispetto all'eq. 29:

$$K_I = \frac{K}{\tau_I} \quad (33)$$

$$K_D = K * \tau_D \quad (34)$$

```
t=0:0.05:3;
t=t';
rife=[t,-10*ones(length(t),1)];


t=3.05:0.05:6;
t=t';
rife2=[t,-4*ones(length(t),1)];

t=6.05:0.05:9;
t=t';
rife3=[t,0*ones(length(t),1)];

t=9.05:0.05:12;
t=t';
rife4=[t,-0.5*ones(length(t),1)];

t=12.05:0.05:15;
t=t';
rife5=[t,0*ones(length(t),1)];

t=15.05:0.05:18;
t=t';
rife6=[t,-1*ones(length(t),1)];

t=18.05:0.05:21;
t=t';
rife7=[t,0*ones(length(t),1)];
```

Figura 6.3: esempio riferimento creato da Matlab

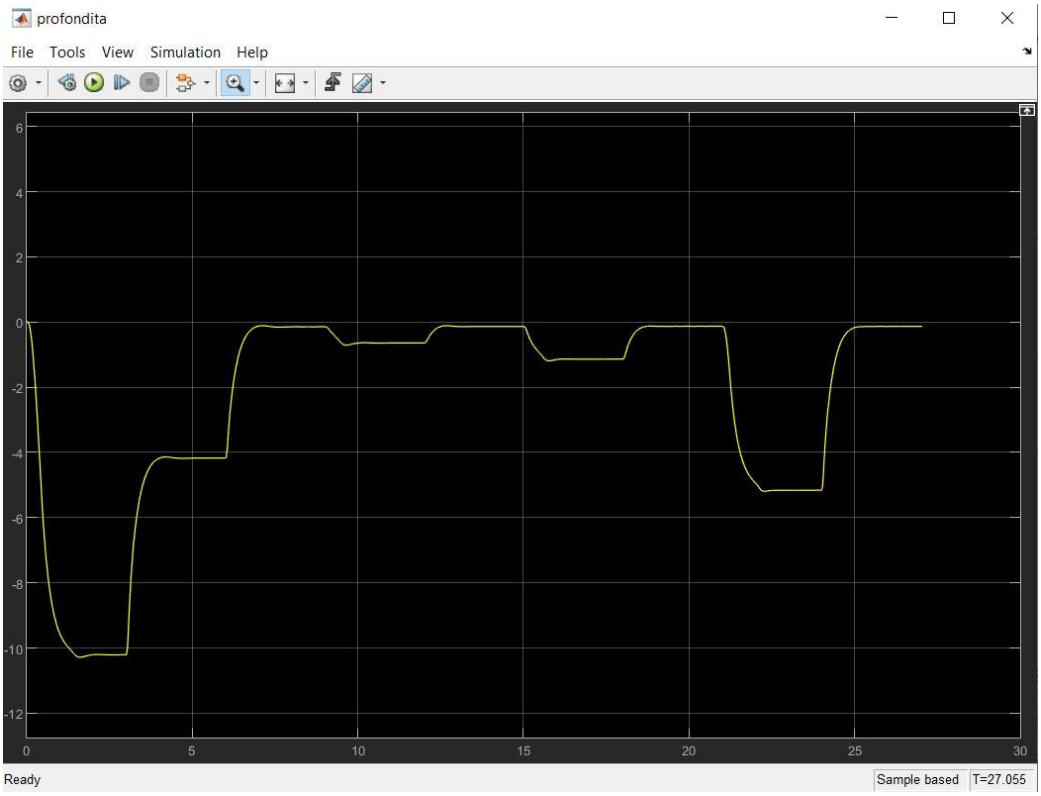

Figura 6.4: risultato controllo sulla profondità

Ora rimane da descrivere ciò che viene ricevuto in ingresso da Adams e quale sia la forma del riferimento. Adams riceve in ingresso il valore reale della rotazione del motore brushless ricavato dallo schema importato da Simscape: in realtà, in ingresso al blocco, si deve avere una matrice costituita da una prima colonna rappresentante i tempi del sistema e una seconda colonna relativa ai valori assunti dalla rotazione del motore. A tal proposito, anche il riferimento della profondità deve essere una matrice in cui si hanno nella prima colonna i tempi del di variazione del riferimento e nella seconda colonna i valori del riferimento.

In *figura 6.3* è presentata una parte di codice che va a formare la matrice riferimento: si può notare come l'asse dei tempi sia in realtà discreto con tempo di step pari allo 0.05. Nel caso presentato in cui si varia il riferimento ogni 3 secondi, poiché il tempo necessario al sistema per stabilizzarsi risulta essere circa un secondo come si nota dalla *figura 6.5*, i riferimenti sono in ordine -10, -4, 0, -0.5, 0, -1, 0, -5, 0. Inoltre, la *figura 6.4* risulta essere anche la conclusione del lavoro di controllo tramite controllore PID mostrando un buon tempo di risposta e una buona risposta a regime con errore che tende a zero.

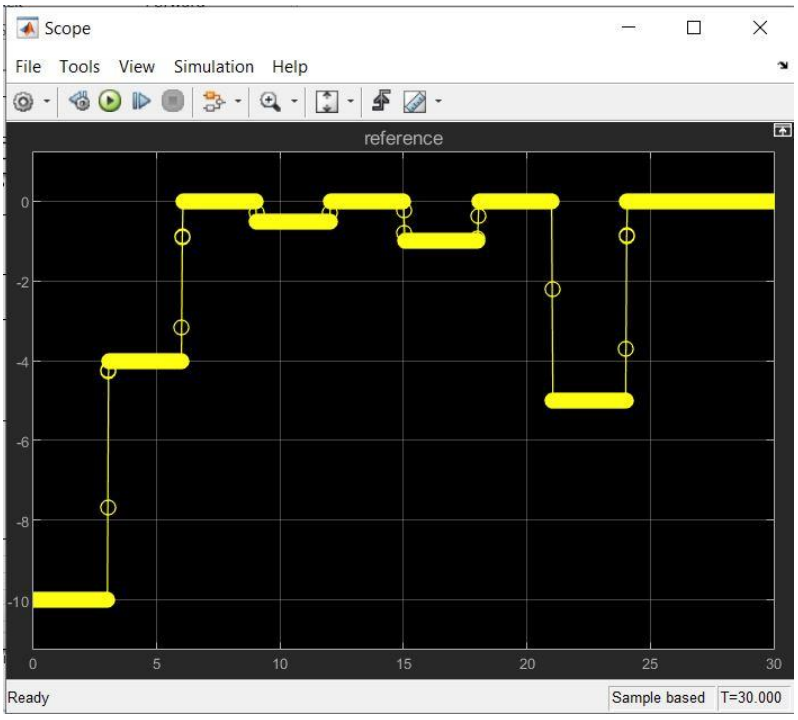

Figura 6.5: esempio di riferimento passato al pesce-robot

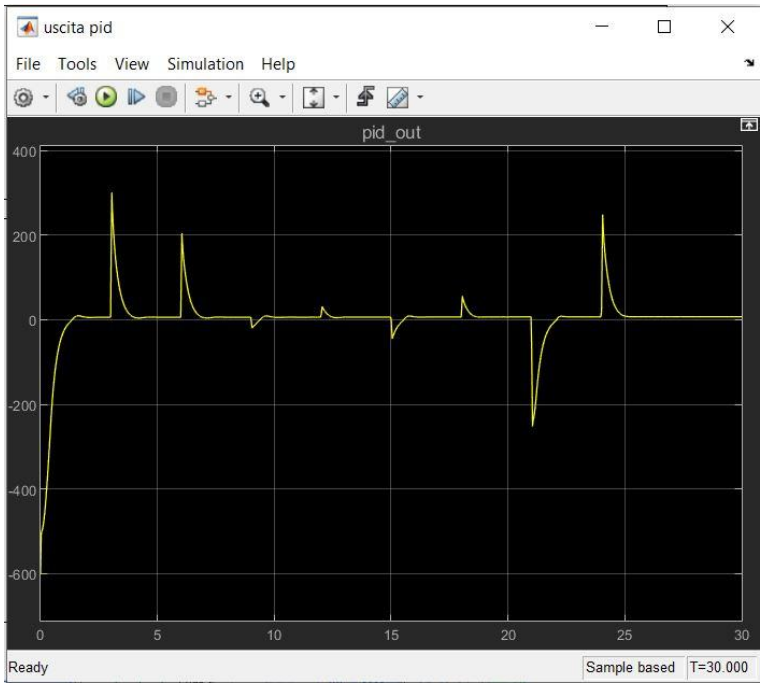

Figura 6.6: grafico velocità reale di rotazione del motore

Conclusione

Ci si è posti il problema dell'identificazione del movimento verticale di un robot pesciforme con tre motori brushless sotto condizioni di equilibrio in acqua. Sviluppando tale problema attraverso uno studio fisico, rifacendosi agli studi di Thor I. Fossen, in ambiente CAD, Adams, e poi in ambiente Matlab per la costruzione di un segnale persistentemente eccitante si è giunti tramite Simulink ad una serie di dati necessari all'identificazione: le coppie ingresso-uscita. A questo punto, mediante il toolbox dell'identificazione in Matlab, si è andati a costruire un modello prendendo in considerazione le famiglie arx e armax: risultato del lavoro è stato il modello armax(3 1 6 0).

Si è effettuato poi, mediante controllore PID, un controllo del modello Adams del pesce, questa volta in condizioni di non equilibrio, considerando cioè la forza peso e la spinta di Archimede di modulo differente. Il controllo è stato fatto sia sulla profondità che il pesce-robot deve raggiungere sia sulla velocità di rotazione del motore brushless, quest'ultimo già implementato in un modello-esempio in Simscape.

In futuro sarà possibile unire tale risultato a quello pervenuto tramite identificazione dell'assetto orizzontale del pesce-robot al fine di avere un modello unico per l'intero robot e poter simulare in maniera reale il suo comportamento.

Altro sviluppo potrebbe essere quello di andare a identificare il vero motore brushless a disposizione effettuando anche un controllo del modello del motore che meglio aderisca a quello reale.

Si potrebbe, in ultima analisi, sviluppare un controllo avanzato in cui, cioè, si andrebbe a controllare non solo la profondità del robot, ma anche altre componenti utili per una simulazione più realistica e precisa; allo stesso modo, partendo da tali risultati, si potrebbe effettuare un controllo time-sharing della profondità.

Bibliografia

- Bittanti, S. (1992). *Identificazione dei modelli e controllo adattivo*. Pitagora Editrice.
- Costa, D., Franciolini, M., Palmieri, G., Crivellini, A., & Scaradozzi, D. (2017, December). Computational fluid dynamics analysis and design of an ostraciiform swimming robot. In *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)* (pp. 135-140). IEEE.
- Costa, D., Palmieri, G., Scaradozzi, D., & Callegari, M. (2018, November). Multi-body analysis of a bio-inspired underwater robot. In *The International Conference of IFToMM ITALY* (pp. 240-248). Springer, Cham.
- Prestero, T. T. J. (2001). *Verification of a six-degree of freedom simulation model for the REMUS autonomous underwater vehicle* (Doctoral dissertation, Massachusetts institute of technology).
- Sfakiotakis, M., Lane, D. M., & Davies, J. B. C. (1999). Review of fish swimming modes for aquatic locomotion. *IEEE Journal of oceanic engineering*, *24*(2), 237-252.
- Fossen, T. I. (2002). Marine Control Systems–Guidance. Navigation, and Control of Ships, Rigs and Underwater Vehicles. *Marine Cybernetics, Trondheim, Norway, Org. Number NO 985 195 005 MVA, www. marinecybernetics. com, ISBN: 82 92356 00 2*.
- Higham, D. J., & Higham, N. J. (2016). *MATLAB guide*. Society for Industrial and Applied Mathematics.
- Adams, M. S. C., & Documentation, C. (2005). Msc. *Software Corporation*.
- Documentation, M. (2005). The MathWorks Inc. *Natick, MA*.