UNIMORE
UNIVERSITÀ DEGLI STUDI DI
MODENA E REGGIO EMILIA

DIPARTIMENTO DI SCIENZE E METODI DELL'INGEGNERIA

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA MECCATRONICA

"Caratterizzazione sperimentale di un azionamento elettrico tramite diagrammi di Bode"

Relatori:

Dott. Emilio Carfagna

Prof. Ing. Fabio Immovilli

Tesi di laurea di:

Simone Piazza

Anno Accademico 2024 / 2025

Indice

Sommario 6
1. Impostazione e sviluppo del lavoro sperimentale 8
1.1. Motore stepper 9
1.1.1. Tipologie di motori stepper 10
1.1.2. Modalità di funzionamento e pilotaggio 11
1.1.3. Coppie caratteristiche 15
1.1.4. Modello elettrico e meccanico del motore 17
1.1.5. Controllo vettoriale del motore 19
1.2. Anelli di corrente e diagramma di Bode 21
1.2.1. Struttura e funzione degli anelli di corrente 22
1.2.2. Struttura del regolatore PI e diagramma di Bode 24
1.2.3. Analisi in frequenza tramite FFT 26
1.3. PLECS 28
1.4. Python 32
1.5. Microcontrollori e STM32CubeIDE 34
2. Sviluppo del software per interfacciarsi con PLECS 38
2.1. Main 39
2.2. Funzioni 42
2.1.3. Implementazione del calcolo di Iq in presenza di coppia 47
3. Modifica del software per comunicare con il motore 49
3.1. Architettura principale (Main) 50
3.2. Funzioni per il calcolo e il salvataggio dei risultati 54
3.2.1. Metodo di demodulazione lock-in 61
3.3. Funzioni per la comunicazione con il microcontrollore 65
4. Software per la stima delle funzioni di trasferimento 70
4.1. Main 72
4.2. Funzioni 79
5. Valutazione delle prestazioni del sistema 84
5.1. Risultati numerici da datasheet 88
5.2. Risultati da simulazioni PLECS 91
5.3. Risultati sperimentali su motore reale 107
5.3.1. Setup sperimentale 108
5.3.2. Dati acquisiti 112
5.4. Confronto tra risultati teorici, simulati e sperimentali 125

6. Conclusioni 137
Bibliografia 140
Ringraziamenti 143

Sommario

Questo lavoro di tesi si è svolto presso il laboratorio di Azionamenti Elettrici del Dipartimento di Scienze e Metodi dell'Ingegneria (DISMI-UNIMORE), presso il campus San Lazzaro di Reggio Emilia; l'obiettivo principale di questa tesi è stato lo studio e l'estrazione sperimentale dei diagrammi di Bode degli anelli di corrente di un azionamento elettrico.

L'azionamento analizzato è costituito da un motore stepper pilotato da una scheda di controllo ST, la quale comunica con un calcolatore esterno tramite connessione Ethernet. Il sistema di comunicazione è stato gestito attraverso un'interfaccia software sviluppata in linguaggio Python, che consente di impostare i parametri del segnale di ingresso, in particolare ampiezza e frequenza. Il segnale vero e proprio viene invece generato all'interno del firmware in linguaggio C eseguito sul microcontrollore, mentre l'interfaccia Python permette inoltre di acquisire le risposte in corrente del sistema.

Prima di procedere con la sperimentazione su un motore reale, è stata condotta una fase preliminare di validazione dello script Python sviluppato per l'iniezione delle sinusoidi di prova, il controllo del motore e l'elaborazione dei dati necessari alla costruzione dei diagrammi di Bode. A tal fine è stato utilizzato un modello del motore in ambiente PLECS, impiegato per eseguire simulazioni al calcolatore che hanno consentito di verificare il corretto funzionamento del codice e delle procedure di analisi prima dell'applicazione sperimentale.

L'attività sperimentale successiva ha avuto come scopo la determinazione delle caratteristiche dinamiche degli anelli di corrente, imponendo sinusoidi di diverse frequenze in ingresso al sistema di controllo. La risposta del sistema è stata quindi registrata e analizzata per ricavare sperimentalmente il modulo e la fase della funzione di trasferimento in anello chiuso, consentendo così di ottenere i corrispondenti diagrammi di Bode.

I dati raccolti durante le prove sono stati elaborati mediante script Python per l'automazione delle procedure di calcolo e la rappresentazione grafica dei risultati. L'analisi dei diagrammi ottenuti ha infine permesso di valutare le prestazioni dinamiche del sistema di controllo di corrente e di confrontarle con le aspettative teoriche, fornendo un utile riscontro sperimentale sul comportamento dell'azionamento.

Successivamente alla determinazione sperimentale degli anelli di corrente e alla costruzione dei relativi diagrammi di Bode, è stato sviluppato un ulteriore software in linguaggio Python finalizzato all'identificazione automatica delle funzioni di trasferimento del sistema. Questo programma, utilizzando i dati sperimentali raccolti, consente di stimare in modo automatico i parametri dinamici equivalenti del sistema, determinando la funzione di trasferimento associata e calcolandone i poli e gli zeri. L'intera metodologia è stata infine validata sperimentalmente su tre diversi motori stepper e per differenti configurazioni dei parametri K_p e K_i del regolatore PI (proporzionale-integrale), al fine di verificare la robustezza dell'approccio e la coerenza dei risultati ottenuti al variare del tuning del controllore.

1. Impostazione e sviluppo del lavoro sperimentale

L'obiettivo di questo lavoro di tesi è lo sviluppo di un programma in Python finalizzato all'estrazione degli anelli di corrente e dei diagrammi di Bode di un motore stepper. Questi strumenti di analisi costituiscono una base fondamentale per la progettazione futura di un controllo efficiente del motore, in particolare per applicazioni che richiedono elevata precisione e una regolazione accurata delle due correnti di fase.

Il lavoro si colloca nell'ambito della modellazione e della simulazione dei sistemi elettromeccanici, con un approccio orientato alla generazione automatizzata di dati e grafici analitici utili per verificare il comportamento dinamico del motore. In particolare, l'estrazione degli anelli di corrente e dei diagrammi di Bode permette di analizzare la risposta in frequenza del sistema e di fornire indicazioni cruciali per la progettazione di regolatori proporzionali-integrali (PI) e di sistemi di controllo vettoriale.

Dopo lo sviluppo del software in Python, progettato per l'estrazione dei diagrammi di Bode degli anelli di corrente, il programma è stato inizialmente testato su un modello PLECS del motore stepper, già precedentemente validato. In questa prima fase è stata implementata una comunicazione diretta tra Python e PLECS, al fine di verificare la correttezza dei dati scambiati e valutare la bontà e l'accuratezza del software sviluppato.

Una volta validato il funzionamento del programma in ambiente simulato, si è passati all'applicazione sul motore reale. In questa seconda fase la comunicazione con PLECS è stata sostituita da un'interfaccia diretta con il microcontrollore STM, incaricato del controllo del motore. Il microcontrollore dialoga con il software Python attraverso una connessione Ethernet, permettendo lo scambio dei dati necessari per l'estrazione sperimentale degli anelli di corrente e del diagramma di Bode.

Per il progetto sono stati impiegati diversi strumenti software complementari:

- Python, per lo sviluppo del programma di calcolo e visualizzazione degli anelli di corrente e dei diagrammi di Bode;
- PLECS, utilizzato per simulare il modello elettromeccanico del motore stepper e generare i dati di corrente di fase;

- STM32CubeIDE, per contestualizzare l'implementazione futura su microcontrollore, anche se in questa tesi non viene realizzato il controllo diretto del motore.

Nei sottocapitoli successivi vengono illustrati in dettaglio:

- il funzionamento del motore stepper e le grandezze elettriche e meccaniche fondamentali per la sua modellazione;
- la teoria degli anelli di corrente e dei diagrammi di Bode, con spiegazione di come questi strumenti permettono di valutare la stabilità e la risposta dinamica del motore;
- le caratteristiche principali dei software utilizzati e il loro ruolo nel flusso di lavoro.

In sintesi, questo capitolo illustra le basi teoriche e i principi fondamentali che hanno costituito il fondamento del lavoro di tesi, fornendo le conoscenze necessarie alla realizzazione del progetto di estrazione degli anelli di corrente e del diagramma di Bode del motore stepper. Queste nozioni rappresentano inoltre il punto di partenza per futuri sviluppi orientati alla progettazione di un controllo più efficiente e ottimizzato.

1.1. Motore stepper

Un motore stepper[15] , o motore "passo-passo", è una macchina elettrica a moto incrementale: ogni impulso di comando corrisponde a un movimento angolare discreto del rotore, detto *passo*, e a una successiva posizione di equilibrio stabile. Comandando una sequenza di impulsi, il motore ruota in modo controllato e preciso, passo dopo passo. L'ampiezza del passo (Equazione 1), indicata con α, è caratteristica del motore e può variare da 0,9° a 30°, determinando il numero di passi per giro

$$S = \frac{360°}{\alpha}$$

Equazione 1-Ampiezza del passo

Uno schema semplificato della sua struttura è mostrato in Figura 1.

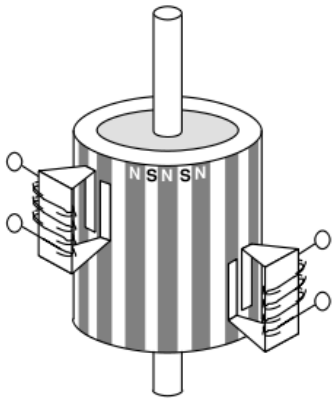

Figura 1-Motore stepper a magneti permanenti

Quando è alimentato, il motore si allinea nella posizione di equilibrio generata dal campo magnetico statorico e può mantenere la posizione anche in assenza di ulteriori impulsi, grazie alla coppia di mantenimento. Questa caratteristica lo rende particolarmente adatto ad applicazioni dove sono richieste precisione e affidabilità, con costi e complessità contenuti.

1.1.1. Tipologie di motori stepper

I motori stepper si classificano principalmente in tre categorie:

- Motori a magneti permanenti (PM) – Il rotore è costituito da un magnete permanente e lo statore da poli salienti alimentati in sequenza. Sono semplici e robusti, ma offrono coppie relativamente basse.

- Motori a riluttanza variabile (VR) – Il rotore è realizzato in materiale ferromagnetico dentato e non contiene magneti permanenti. La rotazione avviene perché il rotore tende ad allinearsi con il campo magnetico generato dallo statore, spostandosi verso la posizione di minima riluttanza magnetica. Questi motori sono generalmente economici e semplici dal punto di vista costruttivo, ma presentano una bassa coppia di mantenimento e, quando non sono alimentati, non riescono a mantenere la posizione.

- Motori ibridi (HY)[11] – Rappresentano una combinazione delle due tipologie precedenti. Hanno un rotore dentato magnetizzato assialmente e uno statore con poli dentati; forniscono coppia elevata, precisione e stabilità di posizione. Sono i più diffusi in ambito industriale e di automazione (Figura 2).

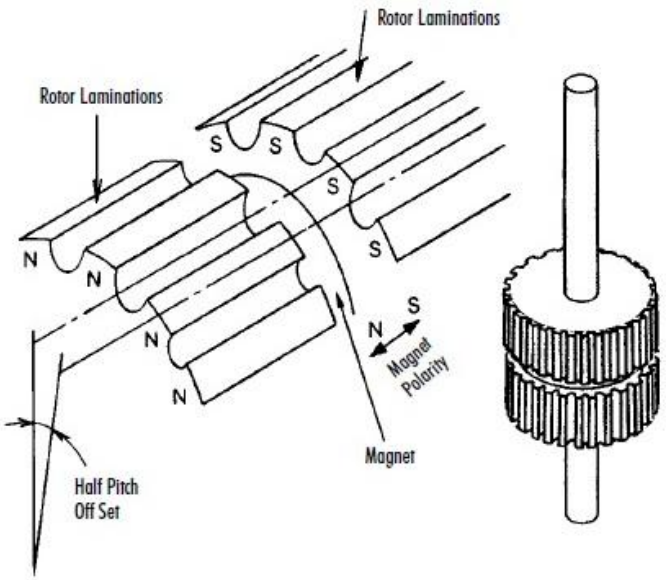

Figura 2-Schema motore stepper ibrido

1.1.2. Modalità di funzionamento e pilotaggio

Il funzionamento si basa sull'eccitazione sequenziale delle fasi dello statore. Si consideri un motore a due fasi, A e B, alimentate con correnti i_A e i_B. La sequenza di eccitazione delle fasi genera un campo magnetico rotante che trascina il rotore passo dopo passo.

Le due principali modalità di pilotaggio sono:

- Unipolare, dove la corrente scorre sempre nello stesso verso in ciascun avvolgimento, richiedendo più terminali ma un'elettronica di pilotaggio più semplice.
- Bipolare, dove la direzione della corrente è invertita tramite ponti H. È più efficiente e consente una coppia maggiore, per cui è la scelta più comune nelle applicazioni moderne.

In commercio esistono entrambe le tipologie che si distinguono l'una dall'altra per il numero di avvolgimenti, quattro per il bipolare e sei per l'unipolare (Figura 3). I più diffusi sono i motori stepper bipolari per la loro semplicità di pilotaggio, per questo motivo si ipotizza che il motore del progetto sia di questo tipo.

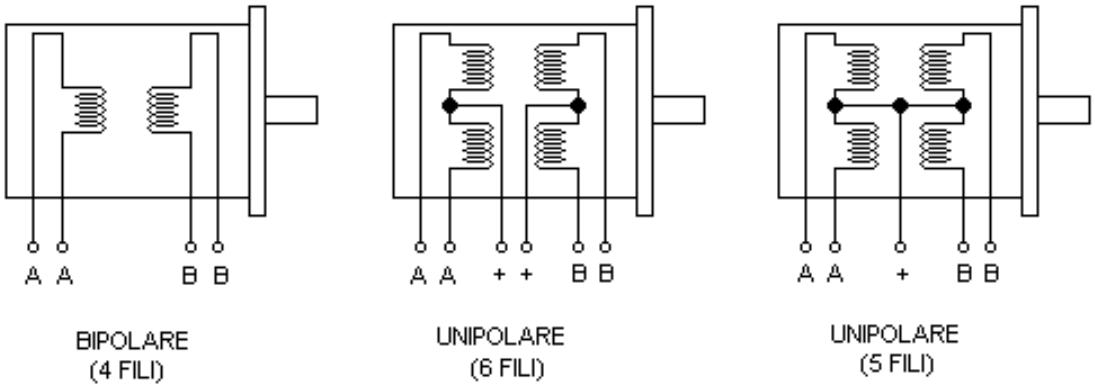

Figura 3-Tipologie costruttive in base alle modalità di pilotaggio

Il comportamento dinamico di un motore passo-passo dipende fortemente dal modo in cui vengono eccitate le sue fasi di statore. Le principali modalità operative sono due: "One phase on" e "Two phase on", ciascuna caratterizzata da differenti prestazioni in termini di coppia, consumo energetico e regolarità di rotazione.

Modalità "One phase on"

In questa modalità di funzionamento viene eccitata una sola fase per volta, mantenendo disattivate le altre. Il motore risulta quindi alimentato in modo alternato secondo quattro configurazioni di base: A, $\bar{A}$, B, $\bar{B}$.

Per comprendere il principio di funzionamento, si consideri il motore costituito da due semistatori:

- un semistatore anteriore, corrispondente alla fase A;
- un semistatore posteriore, corrispondente alla fase B.

La sequenza di pilotaggio avviene secondo i seguenti passaggi (Figura 4):

1. Eccitamento della corrente A → il rotore si allinea ai poli del semistatore anteriore;
2. Eccitamento della corrente B → il rotore ruota fino ad allinearsi ai poli del semistatore posteriore;
3. Eccitamento della corrente $\bar{A}$ → il rotore si riallinea ai poli opposti del semistatore anteriore;

4. Eccitamento della corrente $\bar{B}$ → il rotore completa l'allineamento con i poli opposti del semistatore posteriore.

Al termine della sequenza di quattro passi il rotore compie un ciclo completo. Questo tipo di pilotaggio è particolarmente vantaggioso per la sua semplicità di implementazione e per il basso consumo energetico, poiché solo una fase è attiva alla volta. Tuttavia, la coppia disponibile risulta inferiore rispetto ad altre modalità, e il moto può presentare oscillazioni più marcate, soprattutto alle basse velocità, a causa della minore rigidezza magnetica.

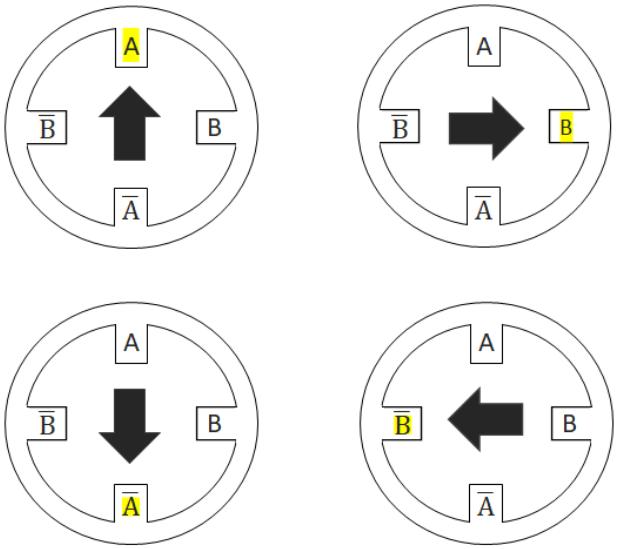

Figura 4-Schema "One phase on"

Modalità "Two phase on"

Nella modalità "two phase on", vengono eccitate entrambe le fasi contemporaneamente. Le configurazioni possibili in un ciclo completo sono quattro: AB, $\bar{A}B$, $\bar{A}\bar{B}$, $A\bar{B}$.

Il principio di funzionamento è analogo a quello della modalità precedente, ma con una differenza sostanziale: la presenza simultanea di corrente in entrambe le fasi genera un campo magnetico risultante più intenso, che permette al rotore di posizionarsi in un equilibrio intermedio rispetto alle direzioni dei campi di fase singola (schema di funzionamento mostrato in Figura 5).

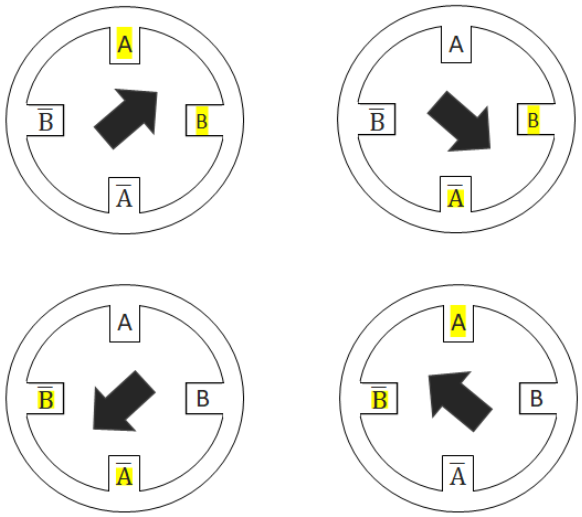

Figura 5-Schema funzionamento "Two phase on"

Questo comporta diversi vantaggi:

- Maggiore coppia sviluppata, a parità di corrente di alimentazione delle bobine;
- Riduzione delle vibrazioni e delle oscillazioni dell'albero motore, migliorando la regolarità di rotazione;
- Maggiore precisione angolare, utile in applicazioni che richiedono controllo fine della posizione.

Per contro, l'assorbimento di potenza è leggermente superiore, in quanto due fasi vengono alimentate simultaneamente.

Per queste ragioni, nella pratica progettuale, in particolare nel sistema considerato in questo elaborato, si preferisce la modalità "Two phase on", che garantisce prestazioni dinamiche superiori e maggior stabilità del moto rispetto alla modalità "One phase on".

1.1.3. Coppie caratteristiche

Tre grandezze fondamentali descrivono il comportamento statico e dinamico di un motore passo-passo, e ne determinano le prestazioni complessive in termini di precisione, forza e stabilità di movimento:

- Coppia residua (Detent Torque, DT):

  È la coppia resistente che si oppone alla rotazione dell'albero anche quando il motore non è alimentato.

  Essa è dovuta principalmente all'interazione magnetica tra i magneti permanenti del rotore e la struttura dentata dello statore, che genera un piccolo momento torcente di attrazione verso le posizioni di equilibrio meccanico. Sebbene di entità ridotta, questa coppia risulta utile in applicazioni che richiedono una leggera resistenza passiva al movimento (ad esempio per mantenere la posizione in assenza di alimentazione), ma può diventare indesiderata in sistemi in cui è necessario un libero scorrimento dell'albero a motore spento. Il suo valore dipende principalmente dal materiale magnetico del rotore e dalla geometria dei denti statorici.

- Coppia di mantenimento (Holding Torque, HT):

  È la coppia massima che deve essere applicata all'albero del motore per spostare il rotore dalla sua posizione di equilibrio quando il motore è energizzato ma non in rotazione.

  In altre parole, rappresenta la capacità del motore di mantenere una posizione fissa sotto carico.

  Questa grandezza è direttamente proporzionale alla corrente di eccitazione delle fasi e dipende dal tipo di pilotaggio utilizzato ("One phase on", "Two phase on" o microstepping).

  La coppia di mantenimento costituisce uno dei parametri principali per la selezione del motore stepper, in quanto definisce la forza di tenuta disponibile durante il posizionamento statico.

  In presenza di regolazioni a controllo vettoriale o a corrente costante, la coppia di mantenimento risulta più stabile e ripetibile nel tempo.

- Coppie dinamiche (Pull-in Torque e Pull-out Torque):

  Queste due grandezze descrivono il comportamento del motore in condizioni di rotazione e sono strettamente legate alla capacità del sistema di mantenere il sincronismo tra il campo magnetico statorico e il rotore (Figura 6).

- La Pull-in Torque rappresenta la coppia massima alla quale il motore può avviarsi da fermo e raggiungere la velocità impostata senza perdere passi.

- La Pull-out Torque, invece, indica la coppia limite oltre la quale il motore perde il sincronismo durante il funzionamento continuo, ovvero non riesce più a seguire la sequenza di eccitazione imposta dal controllore. Entrambe dipendono fortemente dalla velocità di pilotaggio, dalla corrente di alimentazione e dal carico meccanico applicato. In generale, la curva della coppia dinamica decresce all'aumentare della velocità, delineando un'area di funzionamento stabile entro la quale il motore può operare senza perdita di passo. L'adozione di tecniche di controllo vettoriale con regolatore PI consente di mantenere la coppia disponibile più costante lungo l'intero intervallo di velocità, migliorando le prestazioni dinamiche complessive.

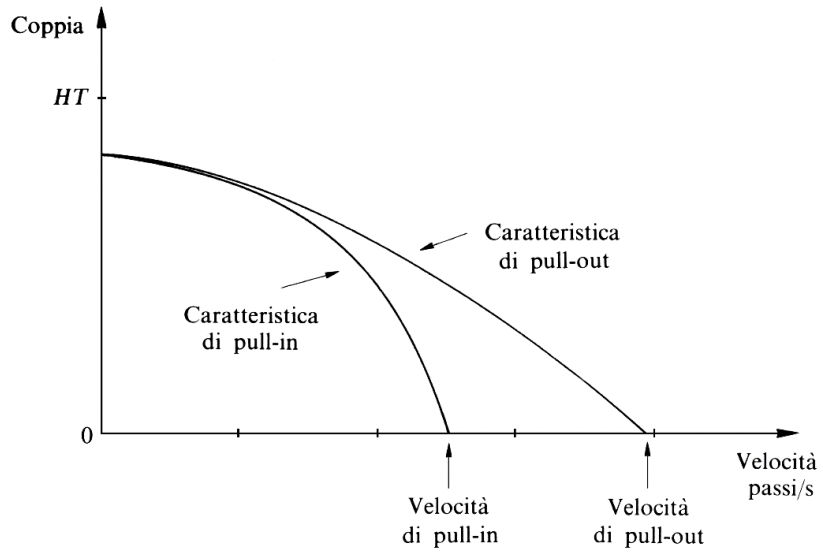

Figura 6-Caratteristica coppia-velocità di un motore stepper

1.1.4. Modello elettrico e meccanico del motore

Il comportamento dinamico di un motore passo-passo può essere descritto attraverso un modello matematico che tiene conto sia degli aspetti elettrici delle fasi statoriche sia della dinamica meccanica del rotore[5].

Questi modelli rappresentano la base concettuale su cui si fondano le strategie di controllo più evolute, come il controllo vettoriale (Field-Oriented Control, FOC).

Partiamo con il descrivere il modello elettrico: ogni fase del motore può essere rappresentata elettricamente come un circuito resistivo-induttivo (RL) in serie con una forza controelettromotrice indotta, proporzionale alla velocità del rotore (Figura 7). L'Equazione 2 che descrive la tensione applicata a una fase può essere scritta come:

$$u_j(t) = R_w i_j(t) + L_w \frac{di_j(t)}{dt} + e_j(t)$$

Equazione 2-Descrive la tensione applicata a una fase

dove:

- u(t) è la tensione applicata alla fase,
- R è la resistenza dell'avvolgimento,
- L è l'induttanza della fase,
- i(t) è la corrente istantanea,
- e(t) è la forza controelettromotrice, proporzionale alla velocità angolare del rotore secondo la relazione $e(t) = K_e \omega(t)$ con $K_e$costante di forza controelettromotrice e $\omega(t)$ velocità del rotore.

La forza controelettromotrice nasce dall'interazione tra il campo magnetico statorico e quello generato dai magneti del rotore; essa agisce in opposizione alla tensione applicata e cresce linearmente con la velocità di rotazione. Questo modello consente di analizzare i transitori elettrici e la risposta della corrente in funzione della velocità, parametri fondamentali per il controllo della coppia.

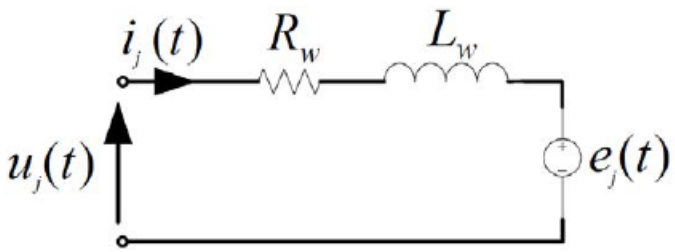

Figura 7-Circuito elettrico equivalente di una fase del motore

Dal punto di vista meccanico, il motore stepper può essere assimilato a un sistema rotazionale soggetto alla dinamica del moto angolare, descritta dall'Equazione 3:

$$J\frac{d\omega}{dt} + B\omega = T_e - T_L$$

Equazione 3-Equazione del moto rotazionale di un albero di motore elettrico

dove:

- J è il momento d'inerzia del sistema rotore-carico,
- B è il coefficiente di attrito viscoso,
- T_e è la coppia elettromagnetica sviluppata dal motore,
- T_L è la coppia resistente del carico.

La coppia T_e è direttamente proporzionale alla corrente statorica e al flusso magnetico, e rappresenta la grandezza di controllo principale.

Questo modello, unito a quello elettrico, fornisce una descrizione semplificata ma efficace della dinamica complessiva del motore, utile per lo sviluppo di controlli in retroazione.

1.1.5. Controllo vettoriale del motore

Il controllo vettoriale[7], o Field-Oriented Control (FOC), è una tecnica originariamente sviluppata per motori sincroni e asincroni, ma applicabile anche ai motori stepper ibridi per migliorare la fluidità del moto e la precisione di posizionamento. Il principio alla base di questo approccio consiste nel rappresentare il sistema in un sistema di riferimento rotante solidale al rotore, in modo da poter controllare in maniera indipendente le grandezze responsabili della generazione di flusso magnetico e di coppia.

Nel caso di un motore stepper bifase, le correnti di fase sono già definite su due assi ortogonali. Esse possono quindi essere trasformate da un sistema di riferimento fisso (α, β) a un sistema di riferimento rotante (d, q) mediante la trasformata di Park (Figura 8). In questo nuovo sistema di riferimento, le componenti di corrente assumono un significato fisico ben definito:

- la componente lungo l'asse d è associata alla generazione del flusso magnetico;
- la componente lungo l'asse q è responsabile della generazione della coppia elettromagnetica.

Questa rappresentazione consente di ottenere un disaccoppiamento tra le due grandezze, rendendo la regolazione della coppia e del flusso più semplice e intuitiva. Le due componenti di corrente possono infatti essere regolate in modo indipendente mediante due regolatori PI.

I segnali di uscita dei regolatori vengono successivamente ritrasformati nel sistema di riferimento fisso tramite la trasformata inversa di Park e quindi utilizzati per generare i segnali di modulazione PWM impiegati per il pilotaggio degli inverter di fase.

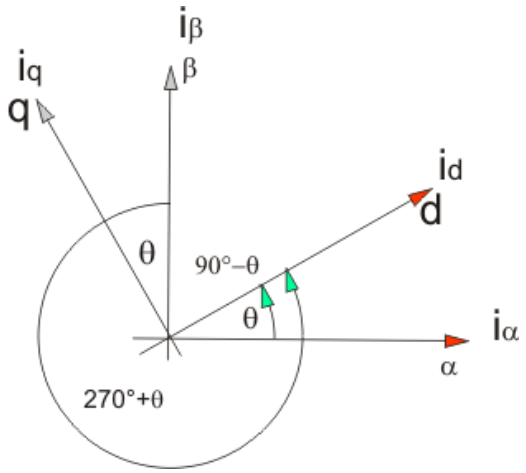

Figura 8-Rappresentazione grafica della trasformata di Park

Nel contesto dei motori passo-passo, questa tecnica consente di trasformare il tradizionale comportamento "a passi" in una rotazione continua e priva di vibrazioni, realizzando il cosiddetto microstepping vettoriale.

In tale modalità, le correnti di fase vengono regolate in modo sinusoidale per ottenere una distribuzione continua del campo magnetico rotante, riducendo rumorosità, coppie pulsanti e risonanze meccaniche. Il risultato è un funzionamento più regolare, preciso e adatto anche a controlli di velocità e posizione in anello chiuso, tipici delle applicazioni industriali avanzate.

1.1.6. Regolatore PI nel controllo vettoriale

Il cuore del controllo vettoriale è costituito dai regolatori PI[19] (Proporzionale–Integrale), uno dedicato a ciascun asse del riferimento rotante, ovvero l'asse d (flusso) e l'asse q (coppia). Ciascun regolatore confronta la corrente misurata (i_d e i_q) con il rispettivo valore di riferimento ($i_{d,ref}$ e $i_{q,ref}$) generando le tensioni di riferimento necessarie per compensare l'errore e mantenere le grandezze controllate ai valori desiderati.

Il termine proporzionale K_p agisce sull'errore istantaneo, determinando la rapidità di risposta del sistema e contribuendo alla stabilità complessiva, mentre il termine integrale K_i ha il compito di eliminare l'errore statico in regime stazionario, assicurando che le correnti misurate coincidano con i valori di riferimento.

I segnali di uscita dei due regolatori PI costituiscono le tensioni di comando nel riferimento d–q, che, attraverso la trasformazione inversa di Park, vengono riconvertite nel sistema di riferimento fisso (α–β o a–b) e successivamente applicate al modulatore PWM per il pilotaggio degli inverter di potenza.

Inoltre, la corretta taratura dei parametri K_p e K_i è fondamentale per garantire un buon compromesso tra prontezza dinamica, stabilità e reiezione dei disturbi. Un valore eccessivo di K_p può rendere il sistema instabile o soggetto a oscillazioni, mentre un termine integrale troppo elevato può introdurre sovraelongazioni e ritardi nella risposta. Per questo motivo, la regolazione dei PI avviene tipicamente in modo empirico o semi-analitico, sfruttando modelli semplificati o procedure di auto-tuning fornite dagli ambienti di sviluppo. In Figura 9 è mostrato lo schema di controllo in retroazione.

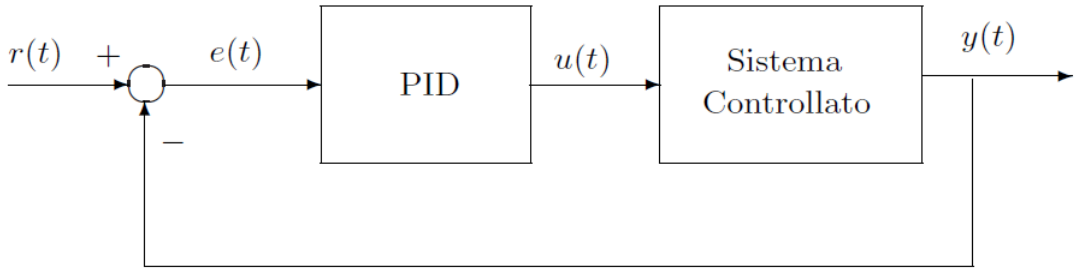

Figura 9-Schema di un sistema di controllo in retroazione

È importante sottolineare che i regolatori PI contribuiscono anche a mantenere il decoupling dinamico tra le componenti *d* e *q*, assicurando che la regolazione del flusso magnetico non interferisca con il controllo della coppia.

In questa sede non si entra nel dettaglio della progettazione degli anelli di corrente né dell'analisi in frequenza (diagrammi di Bode), che saranno trattati nel capitolo dedicato 1.2.

1.2. Anelli di corrente e diagramma di Bode

Nei sistemi di controllo dei motori elettrici, e in particolare nei motori stepper con controllo vettoriale, la precisione e la stabilità del comportamento dinamico dipendono fortemente dal progetto degli anelli di regolazione. Tra questi, gli anelli di corrente rappresentano la parte più interna e veloce della catena di controllo, poiché sono direttamente responsabili della generazione del campo magnetico statorico e, di conseguenza, della coppia erogata dal motore.

Una corretta progettazione di tali anelli consente di ottenere una risposta pronta e stabile alle variazioni di carico, migliorando sensibilmente la fluidità del moto e la precisione del posizionamento.

L'analisi della risposta in frequenza mediante diagrammi di Bode è uno degli strumenti fondamentali per la taratura e la verifica della stabilità degli anelli di corrente. Essa permette di valutare il comportamento dinamico del sistema in catena aperta e di ottimizzare i parametri del controllore, in genere di tipo PI, in modo da garantire un adeguato compromesso tra rapidità di risposta, stabilità e reiezione dei disturbi.

Nelle sezioni seguenti vengono descritte la struttura generale degli anelli di corrente, la loro modellazione, la funzione dei regolatori PI e infine l'utilizzo dei diagrammi di Bode come strumento di analisi e progettazione.

1.2.1. Struttura e funzione degli anelli di corrente

Negli azionamenti elettrici moderni, gli anelli di corrente costituiscono il livello di controllo più interno e più rapido della catena di regolazione, immediatamente a monte dell'inverter. Essi hanno la funzione di regolare le correnti di fase del motore affinché seguano fedelmente i valori di riferimento generati dal controllo vettoriale FOC (Field Oriented Control), che definisce le componenti di flusso i_de di coppia i_q.

Ogni anello di corrente agisce su una singola componente della corrente, confrontando il valore misurato con il riferimento e calcolando la tensione di controllo da applicare al motore tramite un regolatore PI. L'azione proporzionale del regolatore consente di ottenere una risposta rapida alle variazioni del riferimento, mentre l'azione integrale garantisce l'annullamento dell'errore a regime, assicurando un inseguimento accurato della corrente richiesta.

L'obiettivo del controllo di corrente è quindi minimizzare l'errore tra corrente di riferimento e corrente misurata, garantendo al tempo stesso una risposta dinamica rapida e una buona stabilità del sistema.

Poiché la coppia elettromagnetica risulta direttamente proporzionale alla componente di corrente i_q, la qualità del controllo di corrente influisce direttamente sulla precisione del controllo di coppia e, di conseguenza, delle variabili meccaniche superiori come velocità e posizione.

Nel caso dei motori stepper controllati mediante tecniche vettoriali, la presenza degli anelli di corrente consente di superare il tradizionale funzionamento "a passi", caratterizzato da una dinamica discreta e dalla possibile insorgenza di vibrazioni. Il controllo continuo della corrente permette infatti una regolazione più fluida della coppia elettromagnetica, risultando particolarmente vantaggioso nelle applicazioni di microstepping vettoriale e nei sistemi di posizionamento di precisione.

All'interno dell'architettura complessiva dell'azionamento, l'anello di corrente rappresenta il livello più interno di un sistema di controllo multilivello. Esso è affiancato da anelli di controllo più esterni dedicati alla regolazione della velocità e della posizione, i quali generano i riferimenti di corrente necessari al funzionamento del sistema.

Per la progettazione dell'anello di corrente è necessario considerare la dinamica elettrica del motore. Come discusso nel paragrafo 1.1.4, una fase del motore può essere modellata mediante un circuito equivalente di tipo resistivo-induttivo con la presenza della forza controelettromotrice. A partire da tale modello è possibile ricavare la relazione dinamica tra tensione applicata e corrente di fase, che evidenzia come la risposta elettrica del sistema sia assimilabile a quella di un sistema del primo ordine caratterizzato da una specifica costante di tempo elettrica.

La conoscenza di questa dinamica è fondamentale per la corretta sintesi del regolatore di corrente, in quanto la costante di tempo elettrica del sistema determina la velocità massima con cui la corrente può variare e quindi la banda passante realizzabile per l'anello di controllo. Di conseguenza, essa influenza direttamente le prestazioni dinamiche complessive dell'azionamento.

Dal punto di vista implementativo, la tensione di controllo calcolata dall'anello di corrente non viene applicata direttamente al motore, ma è generata attraverso un convertitore di potenza elettronico. In particolare, i moderni azionamenti utilizzano tipicamente un inverter a ponte ad H, costituito da dispositivi di commutazione controllati (come MOSFET o IGBT), che consente di sintetizzare le tensioni di fase a partire da una sorgente di tensione continua.

Il regolatore di corrente fornisce quindi i riferimenti di tensione necessari al controllo del motore, i quali vengono tradotti nei segnali di pilotaggio dei dispositivi di potenza tramite tecniche di modulazione, generalmente basate su modulazione di larghezza d'impulso (PWM).

Attraverso questa modulazione è possibile regolare il valore medio della tensione applicata agli avvolgimenti del motore, consentendo così di controllare con precisione l'andamento della corrente di fase e, di conseguenza, la coppia elettromagnetica sviluppata dal motore.

1.2.2. Struttura del regolatore PI e diagramma di Bode

Il regolatore PI (Proporzionale–Integrale) è il cuore dell'anello di corrente. La sua funzione di trasferimento, nel dominio di Laplace, è espressa dall'Equazione 4:

$$C(s) = K_p + \frac{K_i}{s} = \frac{K_p\, s + K_i}{s}$$

Equazione 4-Funzione di trasferimento di un regolatore PI, con termini proporzionale e integrale

dove:

- K_p è il guadagno proporzionale, responsabile della prontezza di risposta;
- K_i è il guadagno integrale, che annulla l'errore a regime.

Il sistema complessivo in anello chiuso è quindi descritto dalla Equazione 5:

$$T(s) = \frac{C(s)\, G(s)}{1 + \ C(s)\, G(s)} = \frac{(K_p\, s + K_i)/sL}{s + \frac{R + K_p}{L} + \frac{K_i}{Ls}}$$

Equazione 5-Funzione di trasferimento in anello chiuso della corrente con regolatore PI e modello RL della fase del motore

Questo modello consente di analizzare come i parametri K_p e K_i influenzano banda passante, stabilità e margine di fase dell'anello. Un corretto dimensionamento del regolatore consente di aumentare la rapidità di risposta del sistema senza introdurre oscillazioni indesiderate.

Passiamo adesso a descrivere l'analisi in frequenza e il diagramma di Bode[6]. Per studiare la stabilità e le prestazioni dinamiche dell'anello di corrente, si ricorre alla rappresentazione in frequenza della funzione di trasferimento in catena aperta (Equazione 6):

$$L(s) = C(s)\, G(s)$$

Equazione 6-Funzione di trasferimento in catena aperta

Il diagramma di Bode mostra come il sistema risponde a segnali sinusoidali di diversa frequenza, rappresentando (Figura 10):

- nel modulo, misurato in decibel (dB), l'ampiezza del guadagno $|L(j\omega)|$;
- nella fase (in gradi) lo sfasamento introdotto dal sistema.

Questa analisi risulta fondamentale per valutare:

- la frequenza di attraversamento del guadagno (dove $|L| = 0$ dB);
- il margine di fase, ovvero la distanza in gradi dalla condizione di instabilità (−180°);
- la banda passante del sistema, cioè l'intervallo di frequenze in cui il controllo è efficace.

Nel caso di un anello di corrente, si desidera generalmente una banda passante elevata, affinché il controllo possa rigettare rapidamente le variazioni di carico o le perturbazioni dovute alla commutazione PWM. Un margine di fase compreso tra 45° e 60° rappresenta un buon compromesso tra rapidità e stabilità.

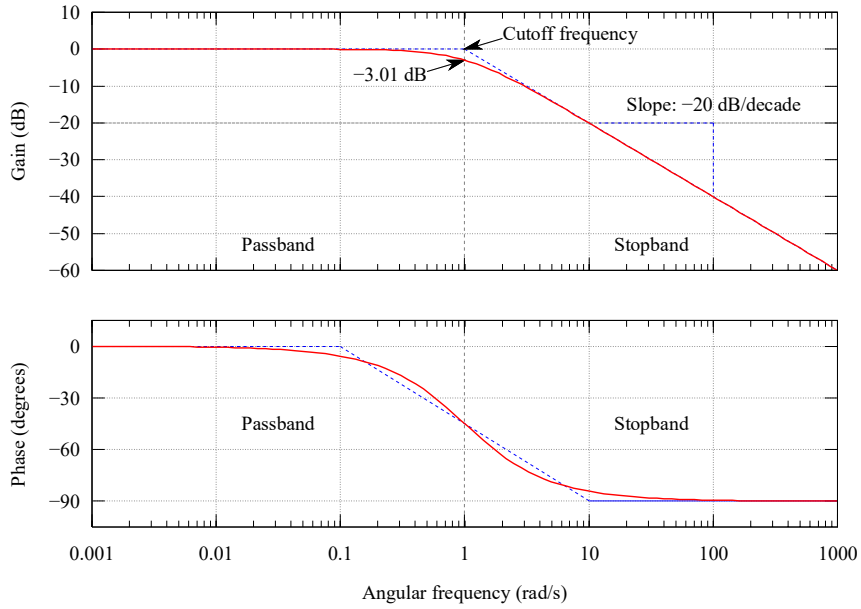

Figura 10-Esempio di diagramma di Bode di un anello di corrente con evidenza dei margini di guadagno e di fase

Gli anelli di corrente rappresentano la base dinamica su cui si costruiscono gli anelli di velocità e posizione. La loro corretta taratura, supportata da un'analisi in frequenza mediante diagrammi di Bode, garantisce che il motore risponda in modo rapido, preciso e stabile alle sollecitazioni del controllo vettoriale.

Nel caso del motore stepper con controllo FOC, l'analisi in frequenza consente di:

- ottimizzare la regolazione delle correnti sinusoidali (i_d, i_q) per ridurre vibrazioni e rumore;
- massimizzare la coppia disponibile a bassa velocità;
- migliorare la linearità della risposta angolare del motore.

1.2.3. Analisi in frequenza tramite FFT

L'analisi in frequenza rappresenta uno strumento essenziale per la caratterizzazione dei segnali e dei sistemi dinamici, consentendo di identificare le componenti spettrali, valutare la presenza di disturbi e confrontare i comportamenti sperimentali con i modelli teorici. In particolare, essa permette di studiare come l'energia di un segnale si distribuisce tra le diverse frequenze, fornendo informazioni cruciali per l'analisi delle prestazioni di sistemi di controllo, motori elettrici e altri dispositivi elettromeccanici.

La Fast Fourier Transform (FFT) è l'algoritmo comunemente utilizzato per realizzare l'analisi in frequenza in modo efficiente. Si tratta di una procedura che trasforma un segnale temporale discreto in un dominio di frequenza, riducendo drasticamente il numero di operazioni necessarie rispetto alla trasformata di Fourier diretta. L'output della FFT consiste nello spettro del segnale, che rappresenta l'ampiezza e la fase delle diverse componenti sinusoidali che lo costituiscono. Questa rappresentazione consente di rilevare picchi di frequenza predominanti, armoniche e fenomeni come risonanze o rumore, che possono non essere evidenti nell'analisi temporale.

A titolo esemplificativo, si consideri un segnale costituito dalla somma di due sinusoidi a frequenze distinte. Nel dominio del tempo, tale segnale appare come un'oscillazione complessa, nella quale non è immediatamente distinguibile il contributo delle singole componenti. Applicando la FFT, il segnale viene trasformato nel dominio della frequenza, dove emergono chiaramente picchi corrispondenti alle frequenze delle sinusoidi originarie.

Questo esempio evidenzia come la trasformata consenta di separare le componenti spettrali di un segnale e di identificarne le caratteristiche principali, rendendo l'analisi più intuitiva rispetto alla sola osservazione nel dominio temporale. In Figura 11 è riportata una rappresentazione grafica di tale processo.

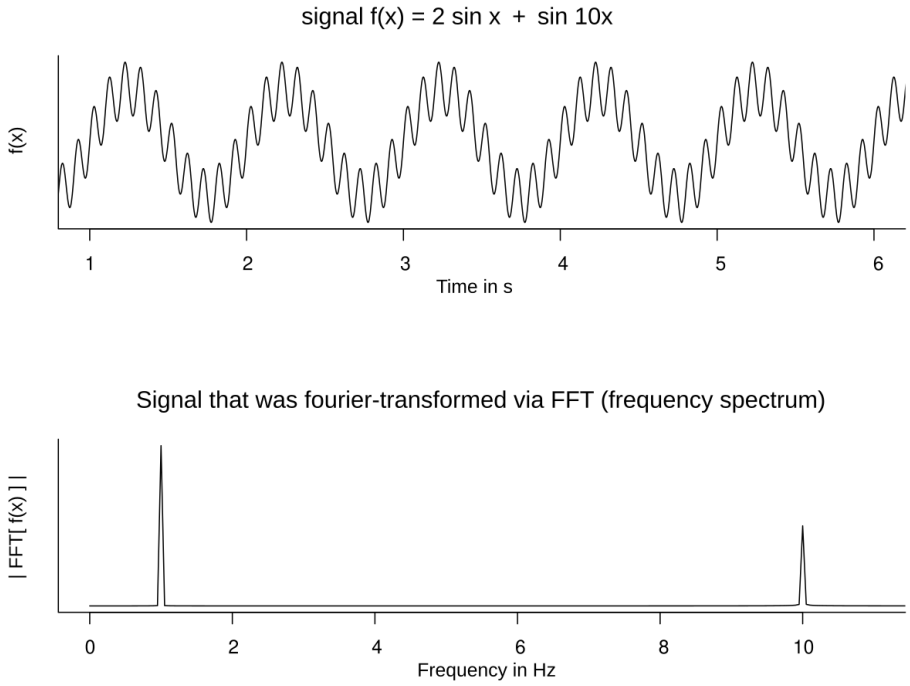

Figura 11-Esempio applicazione FFT

L'applicazione della FFT richiede alcuni accorgimenti pratici: i segnali devono essere opportunamente campionati con una frequenza sufficientemente elevata rispetto alla massima componente presente (rispettando il teorema di Nyquist) e, spesso, vengono applicate finestre di analisi per ridurre gli effetti di leakage spettrale. Inoltre, la risoluzione in frequenza dipende dal numero di campioni considerati e dalla durata della finestra temporale scelta, rendendo possibile un compromesso tra dettaglio spettrale e sensibilità al rumore.

L'analisi tramite FFT è ampiamente utilizzata in ambito ingegneristico per la progettazione e la validazione di sistemi elettronici ed elettromeccanici, l'ottimizzazione dei regolatori, il monitoraggio delle prestazioni e la diagnostica predittiva, fornendo una panoramica completa delle caratteristiche dinamiche di un sistema.

Nel contesto di questa tesi, la FFT è stata impiegata per trasformare i segnali di corrente acquisiti dal modello PLECS in domini di frequenza, supportando l'analisi delle risposte del motore stepper e la costruzione dei corrispondenti diagrammi di Bode, fornendo un riferimento quantitativo per la validazione dei modelli teorici.

Per i segnali acquisiti dai motori reali, invece, è stato preferito utilizzare il metodo del lock-in[3], in quanto consente di ottenere risultati più puliti e accurati. La descrizione dettagliata di questo approccio, nonché le motivazioni alla base della scelta, verranno illustrate nel capitolo in cui è stato effettivamente implementato tramite Python.

1.3. PLECS

PLECS[2] (Piecewise Linear Electrical Circuit Simulation) è un software di simulazione sviluppato dalla società svizzera Plexim GmbH, specificamente progettato per la modellazione e l'analisi di sistemi elettrici ed elettronici di potenza. Si tratta di uno strumento di livello sistemico, che permette di simulare reti elettriche e sistemi di controllo in modo rapido e accurato.

PLECS (logo mostrato in Figura 12) è descritto dal produttore come *"the tool of choice for high-speed simulations of power electronic systems"*, e il suo obiettivo principale è quello di fornire un ambiente in cui il processo di modellazione dei sistemi di potenza sia intuitivo, modulare e basato su rappresentazioni circuitali.

Figura 12-Logo PLECS

Una delle peculiarità che distinguono PLECS da altri ambienti di simulazione, come SPICE o MATLAB/Simulink, è il modo in cui gestisce i dispositivi di commutazione (ad esempio transistor, MOSFET, IGBT).

Nella maggior parte dei simulatori tradizionali, gli interruttori vengono modellati come elementi fortemente non lineari, il che comporta tempi di simulazione più lunghi a causa dei rapidi transitori di tensione e corrente; PLECS adotta, invece, un approccio piecewise-linear, modellando i dispositivi di commutazione come interruttori ideali che commutano istantaneamente tra uno stato aperto e uno chiuso.

Questo metodo presenta due vantaggi fondamentali:

1. Linearità a tratti del sistema: il circuito rimane lineare tra due istanti di commutazione, semplificando notevolmente la risoluzione numerica delle equazioni del sistema.
2. Efficienza computazionale: durante la commutazione, il simulatore deve gestire soltanto due punti di discontinuità (prima e dopo la commutazione), riducendo così il numero di passi di integrazione necessari e migliorando significativamente la velocità di simulazione.

Questa filosofia di modellazione rende PLECS particolarmente adatto per la simulazione di convertitori statici, azionamenti elettrici, inverter, raddrizzatori, convertitori DC/DC e, più in generale, per tutti i sistemi basati sull'elettronica di potenza.

PLECS, inoltre, è un ambiente multi-dominio, che consente di rappresentare e simulare non solo sistemi elettrici, ma anche fenomeni termici, magnetici e meccanici, mantenendo la stessa logica di modellazione circuitale.

Le principali librerie disponibili comprendono:

- Libreria elettrica, che include componenti passivi, dispositivi di potenza, modelli di semiconduttori ideali e reali, trasformatori e modelli di macchine elettriche di vario livello di dettaglio;
- Libreria termica, che permette di modellare i flussi di calore mediante circuiti equivalenti, includendo fenomeni di dissipazione e accoppiamento termico con i componenti elettrici;
- Libreria magnetica, utilizzata per rappresentare induttori, trasformatori e circuiti magnetici complessi;
- Libreria meccanica, particolarmente utile per l'interfacciamento con modelli di macchine elettriche e per la simulazione di azionamenti elettromeccanici completi.

Grazie a questa struttura modulare, PLECS consente di creare modelli complessi che includono l'interazione tra diversi domini fisici, favorendo un approccio di modellazione integrata del sistema (system-level modeling).

Oltre alla modellazione dei sistemi fisici, PLECS offre strumenti avanzati per la progettazione dei controlli.

Il software include blocchi specifici per la regolazione di convertitori di potenza, per la modellazione di controllori analogici e digitali, e un blocco denominato C-Script, che consente di implementare direttamente all'interno del modello algoritmi di controllo in linguaggio ANSI-C.

Questa funzionalità è di particolare interesse per chi si occupa di controllo digitale dei convertitori, poiché permette di testare il comportamento del controllore in un ambiente simulato, prima di implementarlo su hardware reale.

PLECS supporta inoltre la generazione automatica di codice per applicazioni di controllo embedded e sistemi in tempo reale, riducendo il tempo di sviluppo e facilitando la transizione dal modello alla realizzazione fisica del sistema.

Esistono due principali versioni del software:

- PLECS Blockset, progettato per essere utilizzato come *toolbox* all'interno di MATLAB/Simulink. In questa modalità, i modelli e i componenti PLECS sono pienamente compatibili con il solver di Simulink, i suoi blocchi di controllo e gli script MATLAB.
- PLECS Standalone, che opera come ambiente autonomo di simulazione, dotato di un solver proprietario ottimizzato. Questa versione non richiede MATLAB e fornisce un'interfaccia grafica indipendente, mantenendo tuttavia la compatibilità con i modelli e le librerie del Blockset.

Nel contesto di questa tesi è stata utilizzata la versione Standalone, in quanto consente una maggiore efficienza di calcolo e una gestione diretta dei modelli senza necessità di strumenti esterni.

Questa caratteristica lo rende uno strumento particolarmente utile per la validazione sperimentale dei sistemi di controllo e per la formazione in ambito accademico e industriale.

Di seguito in Figura 13 è mostrato un semplice esempio di circuito di ponte ad H implementato in PLECS.

Nel caso specifico del motore stepper che sarà utilizzato nel presente studio, è importante evidenziare che il suo funzionamento richiede il controllo indipendente di due avvolgimenti. Per questo motivo, non è sufficiente un singolo ponte ad H, bensì sono necessari due ponti ad H, uno per ciascuna fase del motore.

Tale configurazione consente di gestire correttamente la sequenza di eccitazione delle fasi, permettendo così il controllo preciso del movimento passo-passo caratteristico di questa tipologia di motore.

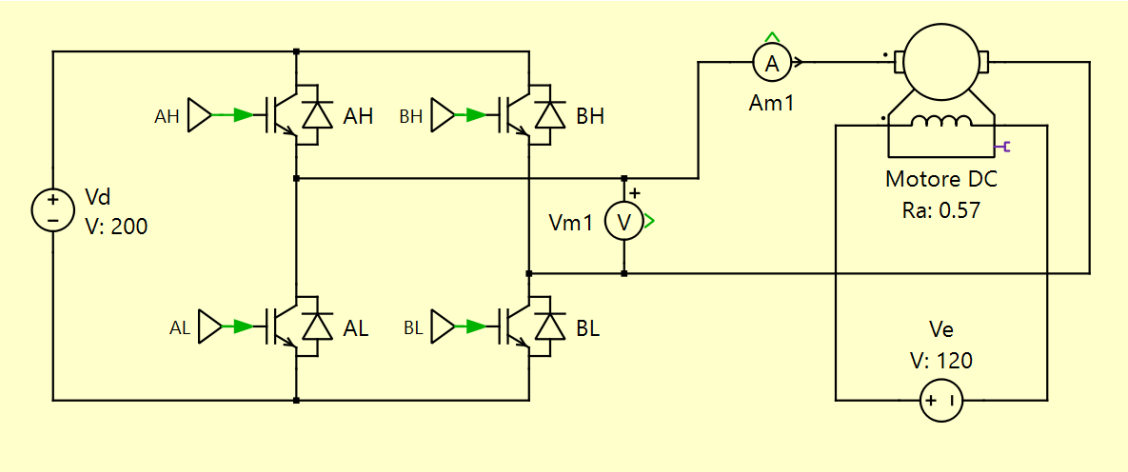

Figura 13-Esempio di ponte ad H simulato

In sintesi, PLECS rappresenta un ambiente di simulazione moderno, efficiente e versatile, progettato per soddisfare le esigenze della progettazione e analisi di sistemi di elettronica di potenza.
L'approccio piecewise-linear, unito alla disponibilità di librerie multi-dominio e alla possibilità di integrare controlli e algoritmi custom, ne fa uno strumento di riferimento sia in ambito accademico che industriale.

1.4. Python

Python[8] (logo mostrato in Figura 14) è un linguaggio di programmazione ad alto livello, interpretato, orientato agli oggetti e caratterizzato da una sintassi semplice, leggibile e vicina al linguaggio naturale. Ideato all'inizio degli anni Novanta dal programmatore olandese Guido van Rossum, Python è stato concepito con l'obiettivo di coniugare potenza espressiva e facilità d'uso, ponendosi come strumento flessibile, dinamico e accessibile sia a programmatori esperti sia a neofiti.

Figura 14-Logo Python

Negli ultimi anni, la popolarità di Python è cresciuta in modo esponenziale, fino a diventare uno dei linguaggi più diffusi in ambito scientifico, accademico, industriale e formativo. Tale successo è dovuto principalmente alla sua versatilità d'impiego: Python trova applicazione nello sviluppo di applicazioni distribuite, nello scripting di sistema, nel calcolo numerico e tecnico-scientifico, nella data science, nell'intelligenza artificiale, nel machine learning e nello sviluppo web. In ambito accademico e di ricerca, Python si è affermato come linguaggio di riferimento grazie alla sua capacità di integrare strumenti di analisi, calcolo e visualizzazione dei dati in un unico ambiente coerente.

Dal punto di vista tecnico, Python non è un linguaggio compilato come ad esempio C, i programmi vengono eseguiti tramite un interprete, che traduce il codice sorgente in bytecode e lo esegue in tempo reale; tale architettura presenta un notevole vantaggio in fase di sviluppo, poiché consente di testare e validare rapidamente il codice, permettendo al programmatore di concentrarsi maggiormente sulla logica e sulla correttezza dell'algoritmo piuttosto che sugli aspetti implementativi.

D’altra parte, i linguaggi interpretati come Python non generano un eseguibile stand-alone: il codice necessita della presenza dell’interprete per poter essere eseguito. Tuttavia, la possibilità di distribuire programmi direttamente in bytecode consente di ottenere versioni del software non modificabili e più efficienti.

Un ulteriore punto di forza di Python è la sua portabilità: il linguaggio è disponibile su tutti i principali sistemi operativi (Windows, macOS e Linux) e garantisce un’elevata compatibilità tra piattaforme. Inoltre, essendo un progetto open source, Python è liberamente utilizzabile, modificabile e redistribuibile; ad esempio viene già incluso nella maggior parte delle distribuzioni Linux.

Dal punto di vista sintattico, Python si distingue per alcune caratteristiche peculiari:

- utilizza l’indentazione per delimitare i blocchi di codice, sostituendo le tradizionali parentesi graffe di altri linguaggi;
- non richiede la dichiarazione esplicita dei tipi di variabile (tipizzazione dinamica);
- supporta diversi paradigmi di programmazione, tra cui quello imperativo, orientato agli oggetti e funzionale;
- offre costrutti avanzati come list comprehension, slicing, overloading di operatori e funzioni, e un’ampia libreria standard di moduli e funzioni.

Python impiega un sistema di tipizzazione forte e dinamica: ogni variabile è un riferimento a un oggetto, e i controlli di tipo vengono eseguiti a run-time. La gestione della memoria è automatizzata grazie a un garbage collector che si occupa del rilascio delle risorse non più utilizzate.
L’interprete Python include inoltre un ambiente interattivo (REPL – Read-Eval-Print Loop), che consente l’esecuzione immediata di comandi e frammenti di codice, facilitando l’esplorazione, il debug e l’apprendimento.

In ambito tecnico-scientifico, Python è divenuto lo standard de facto per lo scripting e l’automazione, grazie alla sua sintassi pulita e alla disponibilità di librerie specializzate come NumPy, SciPy, Pandas, Matplotlib e SymPy, che permettono di eseguire calcoli numerici, analisi statistiche, simulazioni e visualizzazioni di dati in modo efficiente.

Gli script Python vengono spesso impiegati per automatizzare flussi di lavoro complessi, ad esempio:

- l'elaborazione e la preparazione di file di input per simulazioni;
- l'esecuzione di programmi esterni e la gestione dei relativi output;
- l'analisi, l'archiviazione e la visualizzazione dei risultati.

Sebbene le prestazioni di esecuzione di Python siano generalmente inferiori rispetto ai linguaggi compilati staticamente, tipizzati come C o C++, è possibile ovviare a tali limitazioni integrando moduli scritti in C/C++ nelle parti critiche del codice. Inoltre, progetti come PyPy (un interprete alternativo con compilatore Just-In-Time) consentono di migliorare significativamente la velocità di esecuzione di determinati algoritmi.

Infine, Python include strumenti nativi per il testing e la validazione del software, come il framework Unittest, che supporta la scrittura e l'esecuzione di test automatici, favorendo l'adozione di metodologie di sviluppo moderne quali il Test-Driven Development (TDD).

In conclusione, Python rappresenta oggi uno strumento di riferimento non solo per lo sviluppo di applicazioni software generali, ma anche per la ricerca scientifica e l'educazione, grazie alla sua combinazione unica di semplicità, potenza e comunità di supporto attiva. La sua evoluzione costante e l'ampia disponibilità di librerie specialistiche ne consolidano il ruolo come linguaggio privilegiato per la computazione tecnica e scientifica contemporanea.

1.5. Microcontrollori e STM32CubeIDE

Un microcontrollore[16] (MCU, MicroController Unit) è un dispositivo digitale integrato su un singolo chip, progettato per svolgere funzioni di controllo in sistemi embedded. A differenza dei microprocessori tradizionali, che richiedono componenti esterni per memoria e I/O, le MCU integrano CPU, memoria (RAM, ROM o Flash) e periferiche di input/output, costituendo un sistema autonomo e compatto. Questo consente di ridurre dimensioni, costi e consumi, rendendole ideali per applicazioni industriali, domotiche e di consumo, con capacità di interagire direttamente con segnali digitali e analogici (Figura 15).

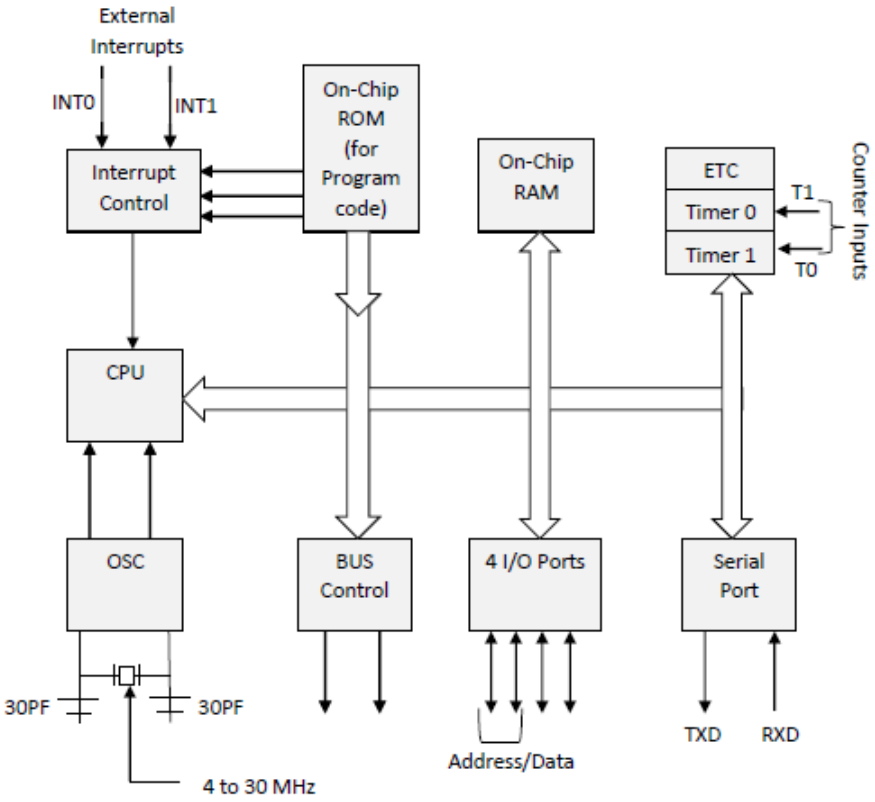

Figura 15-Architettura tipica microcontrollore

I microcontrollori sono nati negli anni Settanta, parallelamente ai primi microprocessori, con dispositivi come il TMS1000 di Texas Instruments (1974), che integrava RAM, ROM e porte I/O sullo stesso chip. Negli anni successivi, l'evoluzione della tecnologia CMOS e l'introduzione di interfacce di programmazione come JTAG hanno permesso l'uso di linguaggi ad alto livello come C e C++, favorendo la diffusione delle MCU in ambito industriale e domestico.

Strutturalmente, una MCU comprende una CPU per l'elaborazione dei dati, memoria volatile e non volatile per programma e dati, periferiche integrate (ADC, temporizzatori, PWM, interfacce seriali come UART, SPI, I²C, CAN e USB) e linee di I/O per interagire con sensori e attuatori. Grazie a questa architettura, le MCU possono gestire applicazioni real-time con tempi di risposta deterministici, rendendole fondamentali in automazione industriale, controllo motori, sistemi automotive e IoT. Oggi sono disponibili architetture a 8, 16 e 32 bit, con quest'ultima – tipica dei core ARM Cortex – standard nei sistemi embedded moderni per il compromesso ottimale tra prestazioni, consumo e costo.

Tra le famiglie più diffuse, gli STM32 di STMicroelectronics rappresentano MCU a 32 bit basate su core ARM Cortex, con elevate prestazioni, efficienza energetica e ampia scalabilità per applicazioni industriali, medicali, IoT e multimediali. Ogni MCU STM32 integra core Cortex (M0, M3, M4, M7, M33 o M55), memoria Flash e SRAM, periferiche analogiche e digitali, interfacce seriali e di rete, oltre a un'interfaccia di debug JTAG/SWD. Le serie STM32 (F, L, G, H, WB, MP) sono ottimizzate per specifici requisiti di consumo, prestazioni o connettività wireless. L'architettura ARM Cortex-M fornisce unità a 32 bit, basso consumo, controller NVIC per gestione efficiente degli interrupt e, nei modelli M4 e superiori, supporto a FPU e istruzioni DSP, rendendo le MCU idonee anche per elaborazioni complesse e AI embedded (Figura 16).

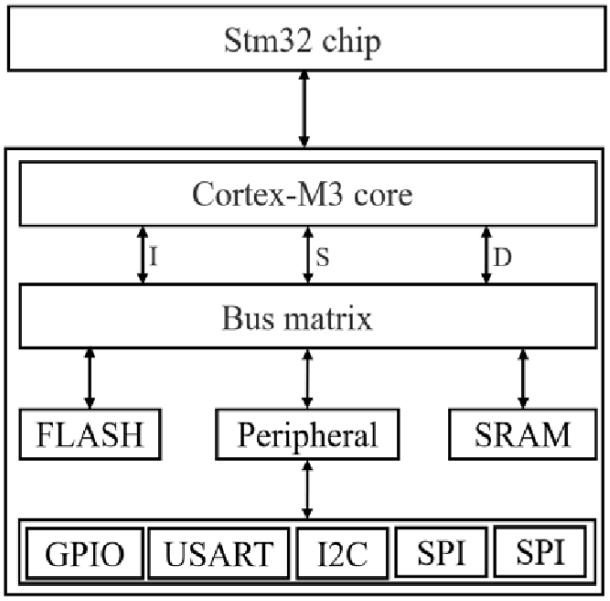

Figura 16-Architettura base STM32

L'ecosistema STM32 include strumenti software e hardware, come librerie HAL/LL, tool di configurazione e schede Nucleo e Discovery, che semplificano progettazione, programmazione e debug. Lo sviluppo del firmware è tipicamente svolto tramite STM32CubeIDE, l'ambiente di sviluppo integrato ufficiale fornito da STMicroelectronics per la programmazione e il debug dei microcontrollori STM32 (Figura 17). Basato su Eclipse e GNU GCC, STM32CubeIDE consente di configurare l'hardware del microcontrollore (pin, periferiche, clock) tramite STM32CubeMX, generare automaticamente il codice di inizializzazione, scrivere e compilare il firmware in linguaggio C/C++ e eseguire il debug on-chip con monitoraggio in tempo reale di variabili e periferiche.

L’IDE permette inoltre di integrare librerie HAL, LL e middleware come FreeRTOS, USB, FATFS e LwIP, e di profilare le prestazioni delle applicazioni real-time, accelerando il ciclo di sviluppo e semplificando la diagnosi di eventuali errori.

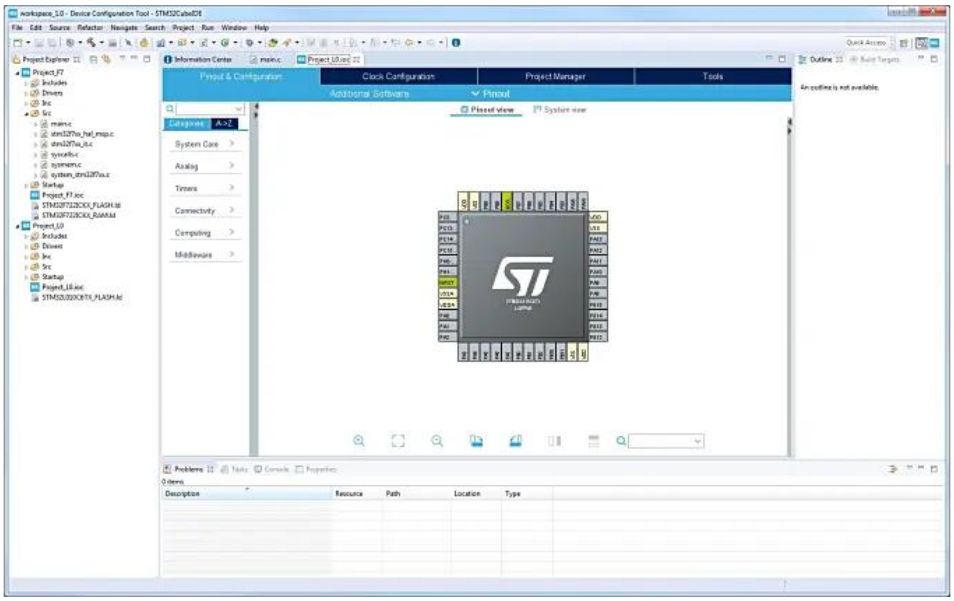

Figura 17-Interfaccia iniziale STM32CubeIDE

2. Sviluppo del software per interfacciarsi con PLECS

In questo capitolo viene descritto nel dettaglio lo sviluppo del software realizzato in linguaggio Python, progettato per automatizzare il processo di estrazione degli anelli di corrente del motore stepper a diverse frequenze operative e per consentire la successiva costruzione del relativo diagramma di Bode. L'obiettivo è quello di dimostrare come, attraverso l'integrazione tra il programma Python e l'ambiente di simulazione PLECS, sia possibile ottenere in modo accurato, ripetibile e automatizzato le grandezze necessarie per l'analisi dinamica del sistema.

Il software è stato progettato secondo una struttura modulare, in modo da garantire scalabilità, riutilizzabilità e facilità di integrazione sia con l'ambiente di simulazione sia, in una fase successiva, con il sistema fisico reale. È importante sottolineare, infatti, che il programma sviluppato, pur essendo inizialmente utilizzato per la comunicazione con PLECS[1], mantiene un'architettura sostanzialmente identica a quella che verrà impiegata per le prove sperimentali sul motore reale. L'utilizzo della simulazione consente quindi non solo di validare il corretto funzionamento del software, ma anche di ottenere valori di riferimento affidabili, utili per il confronto e la verifica delle future misure sperimentali.

Il capitolo è articolato in modo da descrivere progressivamente l'organizzazione e il funzionamento del software. In una prima parte viene illustrata l'architettura generale del programma, con particolare attenzione al modulo principale (Main), responsabile della gestione del flusso operativo complessivo e della comunicazione con PLECS. Successivamente, vengono analizzate nel dettaglio le funzioni ausiliarie che costituiscono il nucleo funzionale del software, evidenziandone il ruolo, la logica di implementazione e le modalità di interazione con il modulo principale.

Questa descrizione consente di comprendere in modo completo il funzionamento del software e le scelte progettuali adottate, fornendo le basi necessarie per il suo successivo impiego nell'acquisizione e nell'analisi dei dati provenienti dal sistema reale.

2.1. Main

Il file principale, main.py, costituisce l'unità di controllo del programma sviluppato in Python[10]. Esso ha il compito di gestire l'interfaccia di comunicazione con PLECS, coordinare le operazioni di analisi alle diverse frequenze di prova, elaborare i risultati e salvarli in formato tabellare e grafico.

Lo script fa uso di moduli standard della libreria Python, come os per la gestione dei percorsi e delle directory, e xmlrpc.client per stabilire la connessione remota con PLECS tramite protocollo XML-RPC, oltre a importare dal modulo Functions_2 le funzioni di analisi e post-processing (fixed_analysis, save_results_excel, plot_bode, print_results).

Struttura generale

Nella parte iniziale del codice sono definite le costanti e i parametri principali di simulazione (Figura 18):

- l'URL del server RPC di PLECS (PLECS_RPC_URL);
- il percorso e il nome del modello PLECS da utilizzare (model_file, model_name);
- gli indici dei segnali di riferimento e misurati per le correnti d e q, necessari per l'estrazione degli anelli di corrente;
- il vettore di frequenze di prova (freqs), espresso in hertz, su cui viene eseguita l'analisi;
- i parametri di simulazione come il numero di periodi da analizzare, il tempo di transitorio, l'ampiezza del segnale di ingresso e il passo di integrazione.

Vengono inoltre generati in modo automatico i percorsi per il salvataggio dei risultati numerici (file Excel) e grafici (diagrammi di Bode), creando le relative cartelle nel caso non esistano già.

```python
import os
import xmlrpc.client
from Functions_2 import fixed_analysis, save_results_excel, plot_bode, print_results

# ---------------------------
# PARAMETRI
# ---------------------------
PLECS_RPC_URL = "http://localhost:1080/RPC2"
model_file = "C:/Users/simon/Desktop/Materiale Tesi/Stepper_Speed_Control_DatiReal.plecs"
model_name = "Stepper_Speed_Control_DatiReal"

idx_id_ref = 0
idx_id_meas = 1
idx_iq_ref = 2
idx_iq_meas = 3

freqs = [100.0, 150.0, 250.0, 400.0, 550.00, 700.00, 850.00, 1000.00, 1150.00, 1300.00, 1450.00]

periods_to_analyze_base = 6
transient_time = 1.0

# Ampiezza fissa
amp = 0.3
fixed_step = 1e-5

coh_target = 0.8

# Percorsi output
base_dir = os.path.dirname(model_file)
excel_path = os.path.join(base_dir, "RisultatiProve.xlsx")
diag_dir = os.path.join(base_dir, "DiagBode")
os.makedirs(diag_dir, exist_ok=True)
```

Figura 18-Definizione dei parametri e inizializzazione dei percorsi di output

Funzione main()

La funzione main() racchiude il flusso operativo del programma:

1. Connessione a PLECS.

   Viene istanziato un oggetto ServerProxy per stabilire la comunicazione RPC e caricare il modello specificato. Al termine del caricamento, un messaggio conferma l'avvenuta connessione tra Python e PLECS; come mostrato in Figura 19.

```python
def main():
    proxy = xmlrpc.client.ServerProxy(PLECS_RPC_URL, allow_none=True)
    proxy.plecs.load(model_file)
    print("Modello caricato e connessione OK.")
```

Figura 19-Inizializzazione della connessione XML-RPC e caricamento del modello PLECS

2. Analisi alle varie frequenze (Figura 20).

   Il codice scorre l'elenco di frequenze predefinite e, per ciascuna di esse, richiama la funzione fixed_analysis(). Tale funzione esegue la simulazione del modello a frequenza costante, calcola la risposta del sistema per le componenti d e q, e restituisce i risultati principali dell'anello di corrente. I dati vengono salvati in due liste separate (results_id e results_iq), una per ciascun asse di controllo.

Dopo ogni simulazione, la funzione print_results() mostra a schermo i risultati intermedi per la frequenza corrente, consentendo un rapido controllo dell'andamento dei dati.

```
for f in freqs:
    res_id, res_iq = fixed_analysis(
        proxy, model_name, f,
        idx_id_ref, idx_id_meas, idx_iq_ref, idx_iq_meas,
        amp, periods_to_analyze_base,
        transient_time, fixed_step, coh_target
    )
    if res_id:
        results_id.append(res_id)
    if res_iq:
        results_iq.append(res_iq)
    print_results(f, res_id, res_iq)

save_results_excel(results_id, results_iq, excel_path)
plot_bode(results_id, results_iq, os.path.join(diag_dir, "Bode.png"))
```

Figura 20-Ciclo for di analisi sulle frequenze e chiamata alla funzione fixed_analysis().

3. Salvataggio ed elaborazione dei risultati.

   Terminata l'analisi di tutte le frequenze, i risultati vengono:

   - esportati in un file Excel tramite save_results_excel(), per una consultazione e analisi numerica;
   - rappresentati graficamente mediante plot_bode(), che genera e salva il diagramma di Bode delle due componenti di corrente, fornendo una visione immediata della risposta in frequenza del sistema.

L'esecuzione dello script è infine avviata dal costrutto standard mostrato in Figura 21; che garantisce l'avvio automatico della procedura principale solo in caso di esecuzione diretta del file.

```
if __name__ == "__main__":
    main()
```

Figura 21-Costrutto per eseguire lo script

È importante sottolineare che lo script è stato inizialmente implementato e testato per la comunicazione con PLECS, allo scopo di validare la correttezza e la precisione del software. In una fase successiva, il medesimo codice verrà utilizzato, senza modifiche sostanziali, anche per la comunicazione con il microcontrollore STM durante le prove sul motore reale, sostituendo la connessione RPC con un collegamento via Ethernet.

In questo modo, il programma mantiene la stessa logica di acquisizione ed elaborazione, garantendo coerenza tra la fase di simulazione e quella sperimentale.

2.2. Funzioni

Il software sviluppato è organizzato in moduli funzionali che si occupano, in modo coordinato, della gestione della simulazione, analisi dei segnali, estrazione dei parametri dinamici e presentazione dei risultati.

Di seguito vengono descritte nel dettaglio le funzioni che costituiscono la struttura del programma.

Funzione fixed_analysis()

La funzione fixed_analysis() integra le due precedenti funzioni e costituisce il blocco operativo principale di misura (Figura 22).

Per ogni frequenza impostata:

- esegue la simulazione del motore in PLECS tramite run_simulation();
- separa le componenti di corrente i_d e i_q, distinguendo rispettivamente segnali di riferimento e misurati;
- richiama la funzione analyze_response() per calcolare guadagno, fase e coerenza di ciascun asse;
- restituisce i risultati in forma strutturata, pronti per l'elaborazione e la visualizzazione.

Questa funzione è richiamata iterativamente dal file principale (main.py) per ogni frequenza del vettore freqs, consentendo di costruire l'intero spettro di risposta del sistema.

```
def fixed_analysis(proxy, model_name, f, idx_id_ref, idx_id_meas, idx_iq_ref, idx_iq_meas,
                   amp, periods_to_analyze_base, transient_time, fixed_step,
                   coh_target, td_correction=0.0):

    periods_to_analyze = periods_to_analyze_base
    periods_to_sim = max(periods_to_analyze * 2, 12)

    Time, Values, _ = run_simulation(proxy, model_name, f, amp, periods_to_sim, transient_time, fixed_step)

    id_ref = Values[idx_id_ref]
    id_meas = Values[idx_id_meas]
    iq_ref = Values[idx_iq_ref]
    iq_meas = Values[idx_iq_meas]

    res_id, coh_id = analyze_response(Time, id_ref, id_meas, f, periods_to_analyze,
                                      transient_time, coh_target=coh_target, td_correction=td_correction)
    res_iq, coh_iq = analyze_response(Time, iq_ref, iq_meas, f, periods_to_analyze,
                                      transient_time, coh_target=coh_target, td_correction=td_correction)

    return res_id, res_iq
```

Figura 22-Codice della funzione fixed_analysis()

Funzione run_simulation()

Questa funzione gestisce il collegamento diretto con PLECS tramite protocollo XML-RPC. A partire dai parametri di ingresso (frequenza, ampiezza, durata della simulazione, passo temporale), la funzione (mostrata in Figura 23):

- definisce i parametri di simulazione da inviare a PLECS, configurando il solutore a passo fisso e le variabili di ingresso (ampiezza e frequenza del segnale sinusoidale);
- avvia la simulazione del modello specificato e riceve in risposta i dati temporali e i valori simulati;
- restituisce i vettori temporali e di segnale necessari per la successiva analisi.

```
def run_simulation(proxy, model_name, f, amp, periods_to_sim, transient_time, fixed_step):
    T = 1.0 / f
    sim_time = periods_to_sim * T + transient_time
    opts = {
        "StopTime": float(sim_time),
        "Solver": "FixedStep",
        "FixedStep": float(fixed_step),
        "ModelVars": {
            "SineAmp": float(amp),
            "SineFreq": float(f)
        }
    }
    result = proxy.plecs.simulate(str(model_name), opts)
    Time = np.array(result["Time"], dtype=np.float64)
    Values = np.array(result["Values"], dtype=np.float64)
    return Time, Values, amp
```

Figura 23-Codice della funzione run_simulation()

Funzione analyze_response()

Questa funzione rappresenta il cuore dell'elaborazione numerica, il suo compito è analizzare la risposta del sistema (corrente misurata) rispetto al segnale di riferimento, per una specifica frequenza di eccitazione, e determinare i parametri fondamentali necessari alla costruzione del diagramma di Bode (Figura 24).

In particolare:

- riceve in ingresso i vettori temporali e i segnali di riferimento e misura (time, ref, meas), insieme alla frequenza di analisi f;
- elimina automaticamente la parte iniziale di transitorio, in base al parametro transient_time;
- segmenta il segnale nei periodi utili all'analisi e applica una finestra di tipo Hann, riducendo l'effetto delle discontinuità ai bordi del segnale;
- calcola le trasformate di Fourier veloci (FFT) di riferimento e misura, estraendo l'ampiezza e la fase alla frequenza di eccitazione;
- valuta il guadagno (in dB) e la fase (in gradi) del sistema, insieme alle rispettive deviazioni standard;

- calcola la coerenza spettrale tra i due segnali (mediante la funzione coherence), utilizzata per determinare la validità della misura;
- restituisce un dizionario contenente tutti i risultati principali, inclusi il numero di periodi analizzati, la frequenza di campionamento e l'indicatore di validità della misura.

L'output della funzione rappresenta quindi un singolo punto della curva di Bode, comprendente modulo e fase per una specifica frequenza di analisi.

```
for r, m in zip(ref_windows, meas_windows):
    R = np.fft.rfft(r * win)
    M = np.fft.rfft(m * win)
    freqs_fft = np.fft.rfftfreq(samples_per_period, 1/fs)
    k = np.argmin(np.abs(freqs_fft - f))

    A_ref = np.abs(R[k]) / (np.sum(win)/2)
    A_meas = np.abs(M[k]) / (np.sum(win)/2)
    A_ref_list.append(A_ref)
    A_meas_list.append(A_meas)

    if np.abs(R[k]) < 1e-12:
        continue
    ratios.append(M[k] / R[k])

ratios = np.array(ratios, dtype=np.complex128)
if len(ratios) == 0:
    return None, 0.0

R_mean = ratios.mean()
gain_db = 20*np.log10(np.abs(R_mean))
phase_deg = np.angle(R_mean, deg=True)

ratios_abs_std = max(np.abs(ratios).std(), 1e-30)
gain_std_db = 20*np.log10(ratios_abs_std)
phase_std_deg = np.degrees(np.std(np.angle(ratios)))

if td_correction and td_correction > 0:
    phase_deg += 360.0 * f * td_correction

f_coh, Cxy = coherence(ref_seg, meas_seg, fs=fs, nperseg=samples_per_period)
coh_at_f = Cxy[np.argmin(np.abs(f_coh - f))]
```

Figura 24-Estratto del codice della funzione analyze_response()

Funzione save_results_excel()

Dopo l'elaborazione, i risultati ottenuti per le due correnti i_d e i_q vengono salvati in un file Excel (RisultatiProve.xlsx), utile per successive analisi e confronti.

La funzione:

- crea un file .xlsx con intestazioni predefinite;
- salva per ogni frequenza i valori di guadagno, fase, ampiezze, coerenza e altri parametri calcolati;
- distingue le misure relative agli anelli *d* e *q* tramite una colonna identificativa ("loop").

Funzione plot_bode()

La funzione plot_bode() si occupa della rappresentazione grafica dei risultati. Utilizzando la libreria Matplotlib, costruisce due grafici sovrapposti:

- il modulo del guadagno in dB in scala logaritmica;
- la fase in gradi in funzione della frequenza.

I due anelli di corrente vengono distinti graficamente mediante differenti stili e marker. Il grafico finale viene esportato automaticamente in formato .png e salvato nella cartella dedicata (Figura 25).

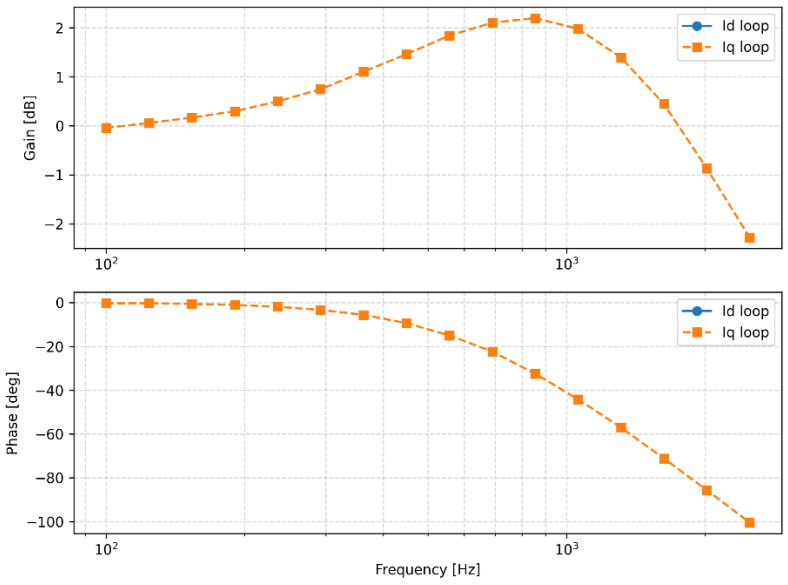

Figura 25-Esempio di diagramma di Bode generato automaticamente dal software

2.1.3. Implementazione del calcolo di Iq in presenza di coppia

Per estendere le capacità del software Python sviluppato, è stata implementata una versione del programma in grado di eseguire le simulazioni dell'anello di corrente i_q anche in presenza di una coppia applicata sul motore stepper. Nella versione originale, infatti, le simulazioni potevano essere condotte assumendo il motore senza carico, oppure risultava possibile determinare solo l'anello i_d in caso di coppia applicata al motore. Tuttavia, per ottenere dati più realistici e rappresentativi del comportamento dinamico del motore sotto carico, è stato necessario introdurre un offset di corrente i_q corrispondente alla coppia impostata.

Per questo scopo, è stata aggiunta la funzione calculate_iq0(), che calcola la corrente media i_{q0} necessaria a generare la coppia desiderata, conoscendo la costante di coppia del motore km (Figura 26). Questo valore rappresenta il punto di lavoro attorno al quale viene sovrapposto il segnale sinusoidale di test.

```
# CALCOLO PUNTO DI LAVORO Iq
# ---------------------------
def calculate_iq0(torque, km):
    return torque / km
```

Figura 26-Funzione calculate_iq0()

La funzione run_simulation() è stata modificata per includere il parametro IqOffset, che viene passato al modello PLECS durante la simulazione. In questo modo, il modello simula correttamente la corrente media necessaria per bilanciare la coppia applicata, mantenendo intatta la componente dinamica del segnale. Grazie a questa modifica, è possibile ottenere simulazioni che riflettono fedelmente il comportamento del motore sotto carico, senza alterare la struttura generale del software.

L'analisi FFT dei segnali è stata adattata per sottrarre l'offset i_{q0} prima del calcolo della risposta in frequenza, garantendo che ampiezza, fase e coerenza siano determinate solo sulla componente dinamica della corrente. In particolare, nella funzione fixed_analysis() vengono passati i parametri torque e km, con il calcolo automatico di i_{q0} e la compensazione dei segnali tramite il parametro offset nella funzione analyze_response(). Questo approccio consente di ottenere risultati più accurati per l'anello i_q anche in presenza di carico.

Le funzioni di salvataggio dei risultati (save_results_excel), tracciamento dei diagrammi di Bode (plot_bode) e stampa dei risultati (print_results) rimangono invariate nella struttura, ma elaborano segnali centrati attorno a i_{q0}, producendo output più rappresentativi delle condizioni operative reali del motore.

In sintesi, la principale differenza rispetto alla versione precedente del software riguarda l'inclusione di un punto di lavoro i_q non nullo per simulazioni con coppia applicata, permettendo di isolare la componente dinamica del segnale e garantendo così la possibilità di ottenere anelli di corrente e diagrammi di Bode realistici.

Il codice descritto è stato quindi implementato per consentire, quando necessario, l'acquisizione e l'analisi della corrente i_q anche in presenza di coppia applicata. Tuttavia, tale funzionalità non verrà utilizzata nelle prove sperimentali presentate in questo lavoro, che saranno svolte esclusivamente sull'anello di corrente i_d.

3. Modifica del software per comunicare con il motore

In questo capitolo viene presentato il software sviluppato in linguaggio Python per eseguire l'identificazione degli anelli di corrente direttamente sul motore reale, attraverso la comunicazione con il microcontrollore che gestisce l'elettronica di potenza. Il programma deriva dallo script originariamente realizzato e validato in ambiente PLECS e ne rappresenta un'evoluzione, adattata per operare sul sistema fisico. Il passaggio dalla simulazione al motore reale[12] ha infatti richiesto una revisione della struttura del software, in particolare per quanto riguarda la gestione della comunicazione, l'acquisizione dei dati e l'integrazione con le routine operative del microcontrollore, al fine di garantire uno scambio di informazioni affidabile e coerente con le tempistiche del sistema reale.

Lo script è stato progettato secondo una struttura modulare, che consente di gestire in modo ordinato e flessibile tutte le fasi dell'esperimento, dalla generazione dei segnali di eccitazione fino all'elaborazione dei dati acquisiti. In particolare, il software comprende:

• un blocco principale (Main), responsabile dell'organizzazione della sequenza delle prove, dell'impostazione delle frequenze di test e del coordinamento delle funzioni di comunicazione con il microcontrollore;

• un insieme di funzioni dedicate allo scambio dati, che permettono di inviare i comandi di eccitazione (quali ampiezza e frequenza) e di ricevere i campioni delle correnti misurate, organizzandoli in modo strutturato per le successive elaborazioni;

• procedure di elaborazione dei segnali acquisiti, che includono la rimozione dei transitori, l'applicazione di opportune finestre di analisi e l'elaborazione dei dati mediante la tecnica del lock-in per la ricostruzione delle sinusoidi misurate, al fine di calcolare i parametri di interesse quali guadagno e fase. È inoltre presente una funzione per la verifica della qualità del segnale, utile a garantire l'affidabilità della stima anche in presenza di rumore;

• moduli per la generazione automatica dei grafici e per l'esportazione dei risultati su file Excel, facilitando il confronto con i dati ottenuti in simulazione e consentendo una documentazione completa delle prove eseguite.

Nei sottocapitoli successivi verranno descritti nel dettaglio i singoli componenti del software e le principali scelte implementative adottate, evidenziando le soluzioni sviluppate per assicurare stabilità, affidabilità e robustezza delle misure sul sistema reale. Questo capitolo è dedicato esclusivamente alla descrizione dello sviluppo e dell'adattamento del software, che costituisce lo strumento fondamentale per l'esecuzione delle prove sperimentali; i risultati ottenuti verranno invece presentati e discussi nel capitolo specificamente dedicato all'analisi sperimentale.

3.1. Architettura principale (Main)

Il file principale del codice Python sviluppato per la comunicazione con il microcontrollore rappresenta il centro di coordinamento dell'intero sistema di acquisizione e analisi sperimentale. Esso ha il compito di gestire la comunicazione UDP con l'hardware, configurare e inviare i comandi necessari all'esecuzione delle prove, raccogliere i campioni restituiti dal microcontrollore e, infine, elaborare i dati per ricavare le caratteristiche dinamiche degli anelli di corrente del motore reale. Analogamente alla struttura adottata nella versione precedente utilizzata per l'interfaccia con PLECS, anche in questo caso il Main si appoggia a moduli Python standard, come os, time e multiprocessing, oltre che alle funzioni personalizzate importate dai moduli udp_functions e Functions, che implementano rispettivamente la comunicazione e l'analisi dei segnali acquisiti.

Nella parte iniziale dello script vengono definiti tutti i parametri fondamentali della prova (Figura 27): l'indirizzo IP e le porte utilizzate per la comunicazione UDP, l'elenco delle frequenze sinusoidali su cui eseguire il test, l'ampiezza del disturbo, la frequenza di campionamento e le durate previste per la raccolta dei dati e per lo scarto del transitorio iniziale. In aggiunta, vengono creati automaticamente i percorsi per il salvataggio dei risultati numerici e grafici, con la generazione della directory dedicata ai diagrammi di Bode.

```
# ---------------------------
# PARAMETRI
# ---------------------------
server_host = "192.168.0.111"
server_port = 7
client_host = "192.168.0.11"
client_port = 55151

freqs = [100.0, 150.0, 250.0, 400.0, 550.00, 700.00, 850.00, 1000.00, 1150.00, 1300.00, 1450.00]
ampiezza = 0.3

ki = 100.0
kp = 0.1
kdesat = 0.0
satmax = 0.0
satmin = 0.0

fs = 50000.0
seconds_to_save = 12.0
seconds_to_discard = 10.0
periods_to_analyze = 16

# Percorsi output
base_dir = os.path.dirname(os.path.abspath(__file__))
excel_path = os.path.join(base_dir, "RisultatiProve.xlsx")
diag_dir = os.path.join(base_dir, "DiagBode")
os.makedirs(diag_dir, exist_ok=True)
bode_path = os.path.join(diag_dir, "Bode.png")

variables_to_send = ["id"]
plot_mode = "ANALYSIS"
```

Figura 27-Definizione dei parametri e inizializzazione dei percorsi di salvataggio

All'avvio del Main vengono creati tre processi paralleli che gestiscono in modo asincrono la comunicazione con il microcontrollore (Figura 28). Il primo processo riceve i pacchetti UDP contenenti i campioni delle correnti misurate e dei riferimenti; il secondo si occupa di decodificare i pacchetti, accumulare i dati in buffer interni e inviarli al Main tramite code, gestendo anche eventuali comandi di reset; il terzo processo invia al microcontrollore i pacchetti contenenti i parametri della prova, quali ampiezza e frequenza del segnale sinusoidale. Questa architettura consente di evitare blocchi dovuti alle operazioni di I/O e garantisce continuità nella ricezione e gestione dei dati.

```
if __name__ == "__main__":

    # Per sapere se salvare o no l'Excel finale
    interrupted = False

    stop_flag = multiprocessing.Event()

    send_queue = multiprocessing.Queue()
    raw_data_queue = multiprocessing.Queue()
    data_queue = multiprocessing.Queue()
```

Figura 28-Creazione processi per la comunicazione con il microcontrollore

Il cuore del Main consiste nel ciclo di prova che scorre tutte le frequenze da testare (Figura 29). Per ciascuna frequenza il programma svuota le code contenenti eventuali dati residui dalla prova precedente, invia un pacchetto di stop intermedio per riportare il sistema a condizioni stazionarie e quindi trasmette i nuovi parametri di ampiezza e frequenza per la prova corrente. Durante la fase di acquisizione, i campioni delle grandezze trasmesse (i_d, i_q, i_{d_ref}, i_{q_ref}) vengono accumulati fino a raggiungere il numero previsto. Terminata la raccolta, viene scartata la parte iniziale del segnale per eliminare il transitorio iniziale, isolando così la risposta a regime.

```
# ------------------------------------------------------------
# LOOP SULLE FREQUENZE
# ------------------------------------------------------------
for f in freqs:
    print(f"\n--- Frequenza {f} Hz ---")

    # Svuotamento code
    for q in (data_queue, raw_data_queue):
        while True:
            try:
                q.get_nowait()
            except queue.Empty:
                break

    # Stop intermedio
    stop_packet_data = [ki, kp, kdesat, satmax, satmin, 0.0, 0.0]
    for var in variables_to_send:
        send_queue.put((var, *stop_packet_data))
    raw_data_queue.put(b'RESET')
    time.sleep(1.0)

    # Comando nuovo test
    packet_data = [ki, kp, kdesat, satmax, satmin, ampiezza, f]
    for var in variables_to_send:
        send_queue.put((var, *packet_data))

    # RACCOLTA DATI
    id_meas, iq_meas, id_ref_meas, iq_ref_meas = [], [], [], []
    expected_samples = int(fs * seconds_to_save)
    total_samples = expected_samples

    start_time = time.time()
    max_wait = seconds_to_save * 2.0
```

Figura 29-Svuotamento buffer con dati vecchi e impostazione nuove frequenze di analisi

Durante la raccolta, il programma verifica continuamente che sia stato raggiunto un numero sufficiente di campioni o che non sia stato superato un time-out massimo, evitando blocchi in caso di problemi di comunicazione. I campioni raccolti vengono quindi smistati in strutture interne e inviati alle funzioni di elaborazione (di cui si parlerà nel capitolo successivo), mentre un riepilogo dei dati viene stampato a video per un controllo immediato dell'andamento della prova. Per garantire sicurezza, i risultati parziali vengono salvati su file Excel durante il ciclo di acquisizione, assicurando che i dati non vadano persi in caso di interruzioni, mentre al termine di tutte le frequenze viene generato un file Excel completo con tutti i dati e i diagrammi di Bode vengono salvati nella cartella dedicata (Figura 30).

```
            # SALVA SUBITO I RISULTATI SU EXCEL (parziale)
            append_result_excel(res_id, "Id", excel_path)
            append_result_excel(res_iq, "Iq", excel_path)

        else:
            print("[WARN] Dati insufficienti.")
            print_results(f, None, None)

        # Stop prova e reset micro
        for var in variables_to_send:
            send_queue.put((var, *stop_packet_data))
        raw_data_queue.put(b'RESET')
        time.sleep(1.0)

    # Plot finale bode
    plot_bode(results_id, results_iq, bode_path)

except KeyboardInterrupt:
    print("\n[INTERRUPT] Interruzione manuale.")
    interrupted = True

except Exception as e:
    print(f"[ERROR] Errore non gestito: {e}")
    interrupted = True

finally:

    # SALVATAGGIO FINALE SOLO SE NON INTERROTTO
    if not interrupted:
        print("\n[SAVE] Programma concluso senza errori → salvo Excel finale...")
        save_results_excel(results_id, results_iq, excel_path)
    else:
        print("\n[SAVE] Programma interrotto → mantengo solo l'Excel parziale.")
```

Figura 30-Salvataggio parziale e finale dei risultati

Il Main gestisce correttamente anche l'interruzione dei processi. In caso di interruzione manuale o di errore non previsto, vengono mantenuti solo i dati parziali salvati, mentre al termine regolare tutti i processi paralleli vengono fermati tramite un flag di stop, assicurando il corretto rilascio delle risorse.

Un elemento chiave di questa versione rispetto alla precedente per PLECS riguarda la gestione dei tempi di acquisizione. Infatti, il microcontrollore non trasmette alcuna informazione sul tempo di esecuzione o sul numero di campioni inviati, quindi la determinazione della durata effettiva della raccolta è calcolata interamente dal programma, conoscendo la frequenza di campionamento del sistema, fissata a 50kHz. In questo modo, il programma mantiene una finestra temporale costante e adeguata per tutte le frequenze, adattando la logica già validata in ambiente simulato a un contesto asincrono di comunicazione con hardware reale.

In sintesi, il main.py rappresenta la regia dell'intero esperimento. Coordina la sequenza delle prove, gestisce la comunicazione in tempo reale, garantisce la raccolta ordinata dei dati e ne assicura il salvataggio sicuro. Pur rimanendo fedele alla logica di acquisizione e controllo adottata in PLECS, il Main si adatta alla natura asincrona della comunicazione con il microcontrollore, fornendo una base solida e affidabile per le successive fasi di elaborazione e analisi dei dati sperimentali.

3.2. Funzioni per il calcolo e il salvataggio dei risultati

Il codice dedicato all'elaborazione dei dati acquisiti dal microcontrollore mantiene una struttura generale coerente con quella già adottata per l'analisi dei risultati provenienti dall'ambiente di simulazione PLECS, preservando la stessa organizzazione modulare e la medesima logica operativa. Tuttavia, alcune modifiche significative sono state introdotte per adattare l'algoritmo alle caratteristiche dei segnali provenienti dal sistema reale, che presentano un livello di rumore sensibilmente superiore rispetto a quelli simulati.

In particolare, la funzione principale responsabile dell'analisi è analyze_lockin, il cui codice è riportato in Figura 31. Essa riceve in ingresso il segnale di riferimento e quello misurato, la frequenza di campionamento, la frequenza di eccitazione e il numero di periodi da analizzare, e restituisce i principali parametri dinamici dell'anello di corrente, tra cui il guadagno, la fase e la coerenza.

Analogamente alla versione utilizzata in simulazione, l'analisi viene eseguita considerando esclusivamente gli ultimi periodi del segnale acquisito, in modo da escludere il transitorio iniziale e garantire che la stima venga effettuata in condizioni di regime stazionario. Il numero di campioni per periodo viene determinato a partire dalla frequenza di campionamento, fissata a 50 kHz, consentendo di identificare con precisione la porzione di segnale da utilizzare per l'analisi.

```
def analyze_lockin(ref, meas, fs, f, periods_to_analyze, coh_target=0.8):
    ref = ref.astype(np.float64)
    meas = meas.astype(np.float64)

    samples_per_period = int(round(fs / f))
    n_periods_total = len(ref) // samples_per_period
    if n_periods_total < periods_to_analyze:
        periods_to_analyze = n_periods_total
    if periods_to_analyze < 1:
        return None, 0.0

    start_idx = -periods_to_analyze * samples_per_period
    ref_seg = ref[start_idx:]
    meas_seg = meas[start_idx:]

    amp_ref, phase_ref = fundamental_amp_phase(ref_seg, f, fs)
    amp_meas, phase_meas = fundamental_amp_phase(meas_seg, f, fs)
    rms_error = np.sqrt(np.mean((ref_seg - meas_seg)**2))

    gain_db = 20.0 * np.log10(amp_meas / amp_ref) if amp_ref > 1e-12 else -np.inf
    phase_deg = phase_ref - phase_meas
    while phase_deg > 180.0: phase_deg -= 360.0
    while phase_deg < -180.0: phase_deg += 360.0

    f_coh, Cxy = coherence(ref_seg, meas_seg, fs=fs, nperseg=samples_per_period)
    coh_at_f = Cxy[np.argmin(np.abs(f_coh - f))]
```

Figura 31-Codice analyze_lockin

A differenza della versione precedente basata su FFT, in questa implementazione la stima dell'ampiezza e della fase della componente fondamentale viene eseguita mediante una tecnica di demodulazione sincrona di tipo lock-in, implementata nella funzione fundamental_amp_phase, riportata in Figura 32. Questa scelta è stata motivata dalla natura dei segnali acquisiti dal motore reale, che risultano significativamente più rumorosi rispetto a quelli ideali generati in ambiente simulato. In presenza di rumore, l'analisi basata su FFT può fornire stime meno accurate, in quanto l'energia del disturbo si distribuisce su più componenti spettrali, rendendo meno precisa l'identificazione della componente fondamentale.

Il metodo lock-in, invece, consente di estrarre selettivamente la componente sinusoidale alla frequenza di eccitazione, migliorando la robustezza della stima anche in condizioni di segnale degradato.

Il principio di funzionamento dettagliato di questa tecnica verrà approfondito in un apposito sottocapitolo.

```
def fundamental_amp_phase(x, f0, fs):
    N = len(x)
    t = np.arange(N) / fs
    cos_ref = np.cos(2*np.pi*f0*t)
    sin_ref = np.sin(2*np.pi*f0*t)
    Xc = 2 * np.sum(x * cos_ref) / N
    Xs = 2 * np.sum(x * sin_ref) / N
    amp = np.sqrt(Xc**2 + Xs**2)
    phase = np.arctan2(Xs, Xc) * 180/np.pi
    if phase > 180.0: phase -= 360.0
    elif phase < -180.0: phase += 360.0
    return amp, phase
```

Figura 32-Codice funzione fundamental_amp_phase

Una volta stimate ampiezza e fase dei segnali di riferimento e misurato, la funzione analyze_lockin calcola il guadagno dell'anello in decibel e la differenza di fase tra i due segnali, fornendo direttamente i parametri necessari per la costruzione del diagramma di Bode. Inoltre, viene calcolata la funzione di coerenza mediante la funzione coherence della libreria SciPy, valutata alla frequenza di eccitazione. Questo parametro rappresenta un indicatore della qualità della misura e consente di verificare l'affidabilità dei risultati ottenuti. La funzione restituisce infine una struttura dati contenente tutti i parametri stimati, inclusi il numero di periodi analizzati e i campioni per periodo, oltre a un indicatore booleano che segnala la validità della misura sulla base della coerenza.

Accanto alla funzione principale di analisi, il codice include la funzione analyze_compare_fundamental, mostrata in Figura 33, che utilizza gli stessi parametri stimati mediante il metodo lock-in per ricostruire le sinusoidi fondamentali del segnale di riferimento e di quello misurato. Questa funzione genera un grafico temporale delle due componenti fondamentali, consentendo una verifica visiva della qualità della stima e del corretto allineamento tra ingresso e uscita.

Oltre alla generazione del grafico, la funzione calcola anche l'errore quadratico medio tra i due segnali, fornendo un ulteriore indicatore quantitativo della bontà della misura.

```
def analyze_compare_fundamental(ref, meas, fs, f, periods_to_analyze, save_dir, loop_name):
    samples_per_period = int(round(fs / f))
    start_idx = -periods_to_analyze * samples_per_period
    ref_seg = ref[start_idx:]
    meas_seg = meas[start_idx:]

    amp_ref, phase_ref = fundamental_amp_phase(ref_seg, f, fs)
    amp_meas, phase_meas = fundamental_amp_phase(meas_seg, f, fs)
    rms_error = np.sqrt(np.mean((ref_seg - meas_seg)**2))

    t = np.arange(len(ref_seg)) / fs
    ref_fund = amp_ref * np.sin(2*np.pi*f*t - np.deg2rad(phase_ref))
    meas_fund = amp_meas * np.sin(2*np.pi*f*t - np.deg2rad(phase_meas))

    plt.figure(figsize=(10,5))
    plt.plot(t, ref_fund, 'r--', label=f"{loop_name}_ref_fund")
    plt.plot(t, meas_fund, 'b-', label=f"{loop_name}_meas_fund")
    plt.xlabel("Tempo [s]")
    plt.ylabel("Corrente [A]")
    plt.title(f"Fondamentale {loop_name} - {int(f)} Hz")
    plt.grid(True)
    plt.legend()
    plt.tight_layout()
    file_path = os.path.join(save_dir, f"{loop_name}Fund_{int(f)}Hz.png")
    plt.savefig(file_path)
    plt.close()
```

Figura 33-Codice funzione analyze_compare_fundamental

Il codice include inoltre la funzione plot_current_vs_time, riportata in Figura 34, che consente di visualizzare l'andamento temporale delle correnti misurate e dei relativi riferimenti per entrambi gli assi i_d e i_q. Questa funzione può operare sia sull'intera acquisizione sia sulla sola porzione utilizzata per l'analisi, permettendo di verificare visivamente la presenza del regime stazionario e l'assenza di anomalie nei segnali acquisiti.

```
def plot_current_vs_time(id_meas, id_ref_meas, iq_meas, iq_ref_meas, fs, freq, save_dir,
                         periods_to_analyze=16, plot_mode="ALL"):

    if plot_mode.upper() == "ANALYSIS":
        samples_per_period = int(round(fs / freq))
        start_idx = max(len(id_meas) - periods_to_analyze * samples_per_period, 0)
        id_meas_plot = id_meas[start_idx:]
        id_ref_plot = id_ref_meas[start_idx:]
        iq_meas_plot = iq_meas[start_idx:]
        iq_ref_plot = iq_ref_meas[start_idx:]
    else:
        id_meas_plot = id_meas
        id_ref_plot = id_ref_meas
        iq_meas_plot = iq_meas
        iq_ref_plot = iq_ref_meas

    t_id = np.arange(len(id_meas_plot)) / fs if len(id_meas_plot) > 0 else np.array([])
    t_iq = np.arange(len(iq_meas_plot)) / fs if len(iq_meas_plot) > 0 else np.array([])
```

Figura 34-Codice funzione plot_current_vs_time

Per quanto riguarda la memorizzazione dei risultati, la funzione append_result_excel, mostrata in Figura 35, consente di salvare progressivamente i parametri stimati all'interno di un file Excel, aggiungendo una nuova riga per ciascuna frequenza analizzata. Questa modalità di salvataggio incrementale permette di preservare i risultati anche in caso di interruzione prematura della prova. La funzione save_results_excel, già introdotta e utilizzata nella versione precedente del codice sviluppato per PLECS, mantiene invece lo stesso funzionamento e consente di salvare l'intero insieme dei risultati al termine della procedura di identificazione.

```python
def append_result_excel(r, loop_name, path):
    headers = ["f [Hz]", "gain_db", "phase_deg", "gain_std_db", "phase_std_deg",
               "A_ref", "A_meas", "coherence", "periods", "fs", "samples_per_period",
               "loop", "valid", "amp_ref_lockin", "amp_meas_lockin",
               "phase_ref_lockin", "phase_meas_lockin", "RMS_error"]

    create_new = not os.path.exists(path)

    if create_new:
        wb = Workbook()
        ws = wb.active
        ws.title = "Risultati"
        ws.append(headers)
    else:
        wb = load_workbook(path)
        ws = wb.active

    if r is None:
        row = ["-"] * 17 + [loop_name, "-"]
    else:
        row = [
            r.get("f"), r.get("gain_db"), r.get("phase_deg"), r.get("gain_std_db"), r.get("phase_std_deg"),
            r.get("A_ref"), r.get("A_meas"), r.get("coherence"), r.get("periods"), r.get("fs"),
            r.get("samples_per_period"), loop_name, r.get("valid"),
            r.get("amp_ref_lockin"), r.get("amp_meas_lockin"),
            r.get("phase_ref_lockin"), r.get("phase_meas_lockin"),
            r.get("RMS_error")
        ]

    ws.append(row)
    wb.save(path)
```

Figura 35-Codice funzione append_result_excel

La generazione dei diagrammi di Bode è affidata alla funzione plot_bode, il cui comportamento rimane invariato rispetto alla versione utilizzata per i dati provenienti da PLECS (Figura 36). Essa rappresenta il guadagno e la fase in funzione della frequenza su scala logaritmica, consentendo una visualizzazione immediata delle caratteristiche dinamiche degli anelli di corrente.

```python
def plot_bode(results_id, results_iq, path):
    plt.figure(figsize=(8,6))
    plt.subplot(2,1,1)
    plt.semilogx([r["f"] for r in results_id if r and r["valid"]],
                 [r["gain_db"] for r in results_id if r and r["valid"]], 'o-', label="Id loop")
    if results_iq:
        plt.semilogx([r["f"] for r in results_iq if r and r["valid"]],
                     [r["gain_db"] for r in results_iq if r and r["valid"]], 's--', label="Iq loop")
    plt.ylabel("Gain [dB]")
    plt.grid(True, which="both", ls='--', alpha=0.5)
    plt.legend()

    plt.subplot(2,1,2)
    plt.semilogx([r["f"] for r in results_id if r and r["valid"]],
                 [r["phase_deg"] for r in results_id if r and r["valid"]], 'o-', label="Id loop")
    if results_iq:
        plt.semilogx([r["f"] for r in results_iq if r and r["valid"]],
                     [r["phase_deg"] for r in results_iq if r and r["valid"]], 's--', label="Iq loop")
    plt.xlabel("Frequency [Hz]")
    plt.ylabel("Phase [deg]")
    plt.grid(True, which="both", ls='--', alpha=0.5)
    plt.legend()
    plt.tight_layout()
    plt.savefig(path, dpi=300)
    plt.show()
```

Figura 36-Codice funzione plot_bode

Analogamente, la funzione print_results, già descritta nel capitolo precedente, mantiene la stessa struttura e permette la visualizzazione sintetica dei risultati direttamente nella console, facilitando il monitoraggio in tempo reale dell'avanzamento della procedura di identificazione.

Nel complesso, la struttura delle funzioni di elaborazione è rimasta coerente con quella già validata in ambiente simulato, garantendo continuità metodologica tra la fase di simulazione e quella sperimentale. Le modifiche introdotte riguardano principalmente la sostituzione dell'analisi basata su FFT con la tecnica lock-in, scelta per migliorare l'accuratezza della stima in presenza di segnali rumorosi, e l'aggiunta di funzioni di supporto per la visualizzazione e il salvataggio dettagliato delle componenti fondamentali. Questa evoluzione del codice ha consentito di adattare efficacemente il metodo di identificazione alle condizioni operative del sistema reale, mantenendo al contempo la stessa logica di analisi utilizzata nella fase di simulazione.

3.2.1. Metodo di demodulazione lock-in

Per l'analisi dei segnali acquisiti dal sistema reale è stata adottata una tecnica di demodulazione sincrona di tipo *lock-in*[18] [4], utilizzata per stimare con elevata precisione l'ampiezza e la fase della componente fondamentale alla frequenza di eccitazione. Questo metodo risulta particolarmente efficace in presenza di rumore, poiché consente di estrarre selettivamente la componente sinusoidale di interesse, filtrando implicitamente tutte le altre componenti spettrali non sincronizzate con il segnale di riferimento.

Il principio alla base del metodo lock-in consiste nel proiettare il segnale misurato su due segnali di riferimento sinusoidali ortogonali alla frequenza di eccitazione: un seno e un coseno della stessa frequenza. Sia $x(t)$ il segnale acquisito e sia f_0 la frequenza della sinusoide di eccitazione. Vengono costruiti due segnali di riferimento (Equazione 7):

$$\cos(2\pi f_0 t)$$

$$\text{sen}(2\pi f_0 t)$$

Equazione 7-Segnali sinusoidali di prova a frequenza

Il segnale acquisito viene quindi moltiplicato separatamente per ciascuno di questi due riferimenti e successivamente mediato nel tempo. In forma discreta, considerando N campioni e una frequenza di campionamento f_s, si ottengono le due componenti (Equazione 8), dove $x[n]$rappresenta la versione discreta del segnale continuo $x(t)$.

$$X_c = \frac{2}{N}\sum_{n=0}^{N-1} x[n]\cos\left(2\pi f_0 \frac{n}{f_s}\right)$$

$$X_s = \frac{2}{N}\sum_{n=0}^{N-1} x[n]\text{sen}\left(2\pi f_0 \frac{n}{f_s}\right)$$

Equazione 8-Componenti sinusoidali di x[n] a frequenza f_0

Queste due quantità rappresentano le proiezioni del segnale sulle componenti in fase e in quadratura alla frequenza di eccitazione. A partire da esse è possibile ricavare direttamente l'ampiezza e la fase della componente fondamentale:

$$A_{meas} = \sqrt{X_C^2 + X_S^2}$$

Equazione 9-Ampiezza del segnale a frequenza

$$\varphi_{meas} = arctg2(X_s, X_c)$$

Equazione 10-Fase del segnale a frequenza

dove A rappresenta l'ampiezza della sinusoide (Equazione 9) e φ la sua fase rispetto al riferimento temporale (Equazione 10).

Dal punto di vista interpretativo, questa operazione equivale a calcolare la correlazione del segnale con due sinusoidi ortogonali alla frequenza di interesse. Tutte le componenti del segnale a frequenze diverse da f_0, inclusi rumore e armoniche, hanno un contributo medio nullo quando integrate su un numero intero di periodi, mentre la componente alla frequenza esatta di eccitazione produce un contributo costante. Questo comportamento rende il metodo intrinsecamente selettivo e robusto al rumore.

Un ulteriore vantaggio della tecnica lock-in è che essa non richiede il calcolo dell'intero spettro del segnale, come avviene nel caso della trasformata di Fourier, ma si concentra esclusivamente sulla frequenza di interesse. Ciò consente di ottenere una stima più stabile e meno sensibile al rumore, soprattutto quando il segnale presenta disturbi o fluttuazioni che renderebbero meno affidabile l'individuazione del picco spettrale tramite FFT.

Una volta note ampiezza e fase del segnale di riferimento e di quello misurato, è possibile calcolare direttamente il guadagno e la fase dell'anello di corrente. Il guadagno in decibel è espresso dall'Equazione 11:

$$G_{dB} = 20log_{10}(\frac{A_{meas}}{A_{ref}})$$

Equazione 11-Guadagno in decibel calcolato sul rapporto tra ampiezza misurata e ampiezza di riferimento

mentre la fase dell'anello è data dalla differenza tra la fase del riferimento e quella del segnale misurato (Equazione 12):

$$\varphi_{anello} = \varphi_{ref} - \varphi_{meas}$$

Equazione 12-Fase dell'anello calcolata come $\varphi_{ref} - \varphi_{meas}$

Questi parametri costituiscono i punti del diagramma di Bode dell'anello di corrente.

Nel codice sviluppato, questa tecnica è implementata nella funzione fundamental_amp_phase, che calcola ampiezza e fase della componente fondamentale a partire dai campioni acquisiti.

La funzione viene richiamata dalla procedura principale di analisi per stimare i parametri dinamici dell'anello a ciascuna frequenza di prova.

L'utilizzo della tecnica lock-in ha permesso di migliorare significativamente l'affidabilità della stima rispetto all'approccio basato su FFT utilizzato nella fase di simulazione. In particolare, la sua capacità di estrarre selettivamente la componente fondamentale ha reso possibile ottenere misure accurate anche in presenza del rumore introdotto dal sistema reale, garantendo una corretta identificazione delle caratteristiche dinamiche degli anelli di corrente.

A conferma dell'efficacia del metodo di demodulazione lock-in, nelle seguenti figure viene mostrato un confronto diretto tra il segnale acquisito senza elaborazione sincrona e la sinusoide ricostruita tramite la procedura di demodulazione descritta.

La Figura 37 riporta il segnale temporale misurato direttamente dal sistema reale. È possibile osservare come la componente sinusoidale risulti fortemente contaminata dal rumore di misura e da disturbi non sincronizzati con la frequenza di eccitazione, rendendo difficile una stima affidabile dell'ampiezza e della fase mediante una semplice osservazione nel dominio del tempo.

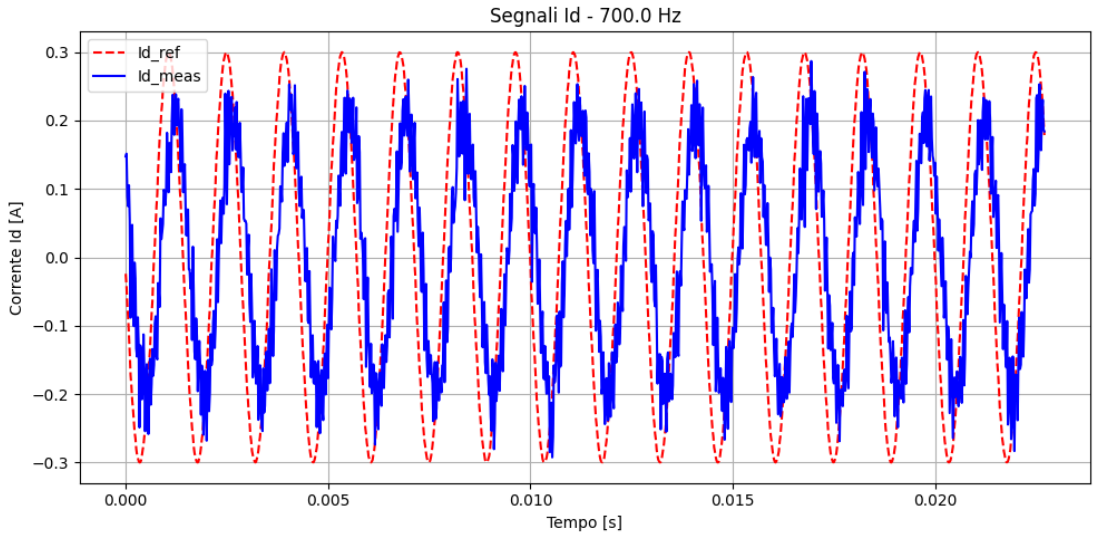

Figura 37-Segnale acquisito senza demodulazione lock-in, affetto da rumore

La Figura 38 mostra invece la sinusoide ricostruita a partire dai parametri stimati tramite il metodo lock-in. In questo caso il segnale appare regolare e privo delle componenti spurie presenti nella misura grezza, evidenziando come la demodulazione sincrona permetta di isolare selettivamente la componente fondamentale alla frequenza di eccitazione.

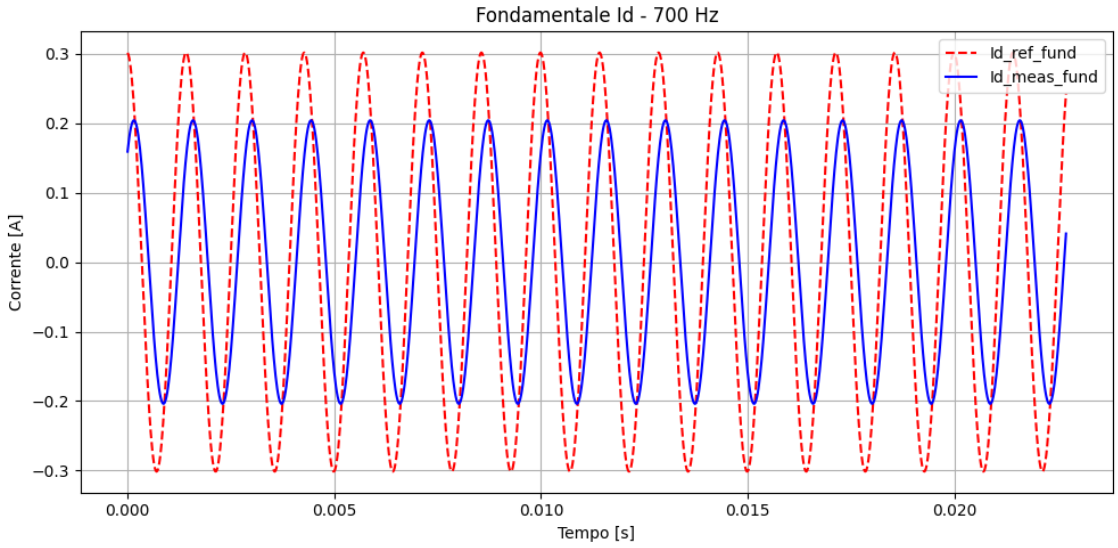

Figura 38-Sinusoide ricostruita a partire da ampiezza e fase stimate mediante demodulazione sincrona.

Questo confronto mette in evidenza il principale vantaggio della tecnica lock-in: la possibilità di ottenere una stima robusta dei parametri dinamici anche in presenza di un rapporto segnale-rumore sfavorevole, condizione tipica delle misure effettuate su sistemi reali.

3.3. Funzioni per la comunicazione con il microcontrollore

In questo capitolo vengono descritte tutte le funzioni necessarie per la comunicazione tra il computer e il microcontrollore via ethernet[17] tramite le funzioni send, udp_server[14] e print_received_packets; queste funzioni costituiscono l'infrastruttura per la comunicazione UDP[13].

- send gestisce l'invio dei comandi verso il microcontrollore,
- udp_server raccoglie i pacchetti provenienti dal micro,
- print_received_packets decodifica i pacchetti, accumula i dati e invia solo i campioni nuovi alla coda di elaborazione per l'analisi degli anelli di corrente.

Questa architettura consente un flusso continuo di dati, con gestione sicura dei buffer, diagnostica integrata e piena compatibilità con il resto del programma Python, mantenendo la stessa logica di acquisizione e post-processing utilizzata precedentemente con il modello PLECS.

debug_bytes

Questa funzione è uno strumento di diagnostica utile per verificare i pacchetti UDP prima dell'invio o durante la ricezione. Riceve un'etichetta (label) e una sequenza di byte (data), e stampa sia la rappresentazione esadecimale che quella numerica intera dei byte (Figura 39).

- Scopo: consentire di verificare che i dati siano correttamente codificati e conformi al formato atteso.
- Uso tipico: chiamata subito dopo la creazione di un pacchetto o alla ricezione di dati dal server, per debug.

```
def debug_bytes(label, data):
    print(f"{label} (len={len(data)}):")
    print("  Hex:", ' '.join(f'{b:02x}' for b in data))
    print("  Int:", list(data))
    print()
```

Figura 39-Funzione debug_bytes

Send

Questa funzione gestisce l'invio dei pacchetti UDP verso il microcontrollore o altro dispositivo di destinazione (Figura 40).

- Connessione: crea un socket UDP e imposta il TTL per multicast.
- Logica principale: in un ciclo infinito, controlla se nella coda tcp_queue sono presenti pacchetti da inviare.
- Formattazione dei dati:
    - La prima variabile della tupla definisce il tipo di segnale (i_d, i_q).
    - Viene associato un identificatore di 1 byte tramite var_map.
    - Gli altri valori numerici (ad esempio k_i, k_p, ampiezza, frequenza) vengono convertiti in float e completi o troncati a 7 elementi.
    - Il pacchetto viene impacchettato con struct.pack secondo il formato =cfffffff.
- Invio: i pacchetti sono inviati verso l'host e porta specificati.
- Diagnostica: la funzione chiama debug_bytes per mostrare i dati inviati.

Questo ciclo include un piccolo ritardo (time.sleep(0.001)) per evitare l'occupazione completa della CPU.

```python
def send(client_host, client_port, tcp_queue, stop_flag):
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    s.setsockopt(socket.IPPROTO_IP, socket.IP_MULTICAST_TTL, 20)

    while not stop_flag.is_set():
        if not tcp_queue.empty():
            tcp = tcp_queue.get()
            if isinstance(tcp, (list, tuple)) and len(tcp) >= 2:
                name = str(tcp[0])
                identifier = var_map.get(name)
                if not identifier:
                    print(f"[WARN] Variabile '{name}' non supportata")
                    continue

                data = [float(v) for v in tcp[1:]]
                if len(data) < 7:
                    data += [0.0] * (7 - len(data))
                elif len(data) > 7:
                    data = data[:7]

                fmt = "=cfffffff"
                try:
                    packed_data = struct.pack(fmt, identifier.encode(), *data)
                except struct.error as e:
                    print(f"[ERROR] Pack fallito: {e}")
                    continue
                debug_bytes("Pacchetto inviato", packed_data)
                try:
                    s.sendto(packed_data, (client_host, client_port))
                except Exception as e:
                    print(f"[ERROR] sendto fallito: {e}")

        time.sleep(0.001)
    s.close()
```

Figura 40-Funzione Send

udp_server

Funzione dedicata alla ricezione dei pacchetti UDP dal microcontrollore (Figura 41).

- Bind: lega il socket UDP all'host e porta specificati.
- Loop infinito: riceve i pacchetti tramite recvfrom.
- Accodamento: ogni pacchetto ricevuto viene inserito nella coda raw_data_queue per l'elaborazione successiva.
- Gestione errori: eventuali eccezioni durante la ricezione sono intercettate e stampate.

Questa funzione serve a garantire che tutti i pacchetti inviati dal micro vengano raccolti in modo affidabile senza perdita di dati.

```python
def udp_server(host, port, buffer_size, result_queue, raw_data_queue, stop_flag):
    with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as udp_socket:
        udp_socket.bind((host, port))
        print(f"UDP server listening on {host}:{port}")

        while not stop_flag.is_set():
            try:
                data, addr = udp_socket.recvfrom(buffer_size)
                raw_data_queue.put(data)
            except Exception as e:
                print(f"[ERROR] recvfrom fallito: {e}")
```

Figura 41-Funzione udp_server

print_received_packets

Questa funzione gestisce la decodifica dei pacchetti ricevuti e la preparazione dei dati per l'analisi degli anelli di corrente (Figura 42).

- Accumulo dati: mantiene un dizionario YDisplay per accumulare tutti i campioni ricevuti per ciascuna variabile (i_d, i_q, i_{d_ref}, i_{q_ref}).
- Gestione delta: tramite last_idx calcola solo i nuovi campioni ricevuti dal micro e li invia alla coda data_queue.
- Reset dei buffer: intercetta il pacchetto sentinel b'RESET' per azzerare l'accumulatore interno, utile tra prove successive.
- Validazione pacchetti: controlla che siano binari e di lunghezza corretta.
- Stampa diagnostica: per i primi 3 pacchetti ricevuti stampa i valori decodificati per ciascun segmento, utile per verificare la correttezza del formato.
- Decodifica: suddivide il pacchetto in segmenti, deserializza i float e aggiorna il cumulativo YDisplay (Figura 42).
- Invio delta: calcola le differenze rispetto all'ultimo indice pubblicato e invia solo i nuovi campioni alla coda di elaborazione.
- Debug periodico: ogni secondo stampa il totale dei campioni accumulati per monitorare il flusso dati.

```
# Decodifica pacchetto e accumula cumulativo
seg_len = len(raw_data) // 4
for idx, var_name in enumerate(segment_to_var):
    start = idx * seg_len
    end = start + seg_len
    segment = raw_data[start:end]
    if len(segment) < 5:
        continue  # almeno 1 byte header + 4 byte float
    num_floats = (len(segment) - 1) // 4
    if num_floats <= 0:
        continue
    try:
        data = struct.unpack("f" * num_floats, segment[1:1 + num_floats * 4])
    except struct.error:
        print(f"[WARN] Unpack fallito per {var_name} len={len(segment)}")
        continue
    if var_name not in YDisplay:
        YDisplay[var_name] = []
    YDisplay[var_name].extend(data)
```

Figura 42-Parte della funzione print_received_packets che decodifica e accumula i pacchetti ricevuti

4. Software per la stima delle funzioni di trasferimento

Il presente capitolo si concentra sul calcolo e sull'analisi delle funzioni di trasferimento (FdT) e dei modelli dinamici $G(s)$ nei sistemi di controllo dei motori elettrici, con particolare attenzione agli anelli di corrente. Lo studio delle FdT costituisce uno strumento fondamentale nella progettazione, nella verifica e nell'ottimizzazione dei regolatori, poiché permette di descrivere il comportamento dinamico del sistema in risposta a ingressi variabili nel dominio della frequenza.

Il calcolo delle funzioni di trasferimento è essenziale per diversi motivi. In primo luogo, consente di ottenere una rappresentazione quantitativa e continua delle caratteristiche dinamiche del motore e del controllore, inclusi guadagno, fase e stabilità. Questo tipo di analisi facilita l'identificazione dei limiti prestazionali, come margini di guadagno e di fase, frequenze di attraversamento e tempi di risposta, che sono elementi critici nella progettazione di regolatori PI o PID efficienti.

In secondo luogo, la determinazione della funzione di trasferimento $L(s)$ in anello aperto e della funzione di trasferimento close-loop $T(s)$ permette di confrontare direttamente il comportamento reale del sistema con la modellazione teorica e con i risultati ottenuti in simulazione. La relazione tra anello chiuso e anello aperto, espressa dalla Equazione 13:

$$T(s) = \frac{L(s)}{1 + L(s)}, L(s) = \frac{T(s)}{1 - T(s)},$$

Equazione 13-Relazione tra le funzioni di trasferimento in anello aperto e chiuso di un sistema di controllo

rappresenta un elemento chiave per la ricostruzione dell'anello aperto a partire da misure sperimentali, consentendo di analizzare la stabilità e la robustezza del regolatore implementato.

Un altro aspetto cruciale riguarda l'identificazione di un modello razionale $G(s)$ che approssimi il comportamento osservato sperimentalmente. La funzione di trasferimento continua così ottenuta fornisce una rappresentazione matematica compatta del sistema, utile non solo per le analisi di stabilità in frequenza, ma anche per la progettazione di nuovi regolatori, la predizione del comportamento sotto condizioni variabili e l'integrazione in strumenti di simulazione avanzati.

La forma generale del modello razionale può essere espressa come mostrato dall'Equazione 14:

$$G(s) = \frac{b_0 + b_1 s + \cdots + b_n s^n}{1 + a_1 s + \cdots + a_m s^m},$$

Equazione 14-Funzione di trasferimento generica di un sistema lineare, espressa tramite polinomi in s

che rappresenta un'astrazione continua della dinamica osservata, consentendo di passare da una descrizione numerica discreta a una modellazione analitica completa.

Il calcolo delle FdT e dei modelli $G(s)$ riveste un ruolo particolarmente importante nello studio dei motori elettrici e dei loro sistemi di controllo perché permette di:

1. Analizzare la stabilità e la robustezza del sistema, individuando frequenze critiche e margini di sicurezza.

2. Valutare le prestazioni dei regolatori implementati, verificando la coerenza tra comportamento sperimentale e comportamento teorico previsto.

3. Effettuare eventuali interventi di tuning dei parametri di controllo, basati su dati oggettivi e predizioni affidabili.

4. Integrare i risultati sperimentali in un modello matematico utilizzabile per simulazioni future, ottimizzazione del progetto e confronto con scenari ipotetici.

In sintesi, questo capitolo illustra e spiega il codice sviluppato per determinare le funzioni di trasferimento (FdT) e i modelli $G(s)$ del sistema controllato, trasformando i dati sperimentali in strumenti analitici utili per comprendere la dinamica dei motori e dei relativi sistemi di controllo. I due sottocapitoli successivi forniscono una descrizione dettagliata del Main, che coordina l'intero flusso di elaborazione, e delle funzioni dedicate, responsabili dei calcoli fondamentali necessari per ricostruire l'anello aperto, valutare le prestazioni del regolatore e ottenere un modello matematico continuo del sistema.

4.1. Main

Il file principale del programma Python[9] dedicato al calcolo delle funzioni di trasferimento rappresenta il centro di coordinamento dell'intera fase di post-processing dei dati sperimentali. Il suo ruolo è organizzare e automatizzare la sequenza di elaborazioni necessarie per trasformare i risultati dell'identificazione in frequenza degli anelli di corrente in una descrizione dinamica completa del sistema controllato.

Diversamente dal Main impiegato nelle fasi di simulazione o acquisizione sperimentale, questo script non gestisce comunicazioni con dispositivi esterni né operazioni di acquisizione in tempo reale. Tutti i dati utilizzati sono già stati precedentemente salvati su file; il programma assume quindi la funzione di regia dell'analisi numerica, coordinando in modo sequenziale le diverse procedure di elaborazione e demandando alle funzioni dedicate l'esecuzione dei singoli algoritmi di calcolo.

All'inizio dello script viene definita la configurazione generale dell'analisi. In questa sezione sono specificati il percorso del file Excel contenente i risultati sperimentali, il foglio da cui importare i dati e l'anello di corrente oggetto dello studio, permettendo di selezionare alternativamente la dinamica dell'asse d o dell'asse q. Vengono inoltre impostati i parametri del regolatore PI attualmente implementato nel sistema reale, utilizzati nelle successive valutazioni prestazionali (Figura 43).

La presenza di variabili di configurazione centralizzate consente di riutilizzare lo stesso script per differenti campagne sperimentali senza modificare la struttura del codice, garantendo ripetibilità dell'analisi e semplicità di aggiornamento.

```
# ============================================================
# PARAMETRI UTENTE
# ============================================================
excel_file = 'Risultati_finali.xlsx'
sheet_name = 0
anello = 'Id'           # 'Id' o 'Iq'

Kp = 0.19
Ki = 100.0

calculate_open_loop = True       # Se True salva e plotta anche open-loop
plot_compare_PI = True
fit_open_loop_flag = True       # Se True calcola G(s) open-loop

# Parametri fitting razionale CLOSED-LOOP
m = 1  # numero di poli
n = 0  # numero di zeri

# Parametri fitting razionale OPEN-LOOP (indipendente)
m_ol = None  # numero di poli open-loop (None = automatico)
n_ol = None  # numero di zeri open-loop (None = automatico)
m_ol_offset = 1
n_ol_offset = 0

output_dir = "output_fdt"
os.makedirs(output_dir, exist_ok=True)
```

Figura 43-Configurazione iniziale del programma e definizione dei parametri di analisi

Successivamente il Main avvia la fase di importazione dei risultati sperimentali. Il programma richiama le funzioni dedicate al caricamento del file Excel e alla selezione dell'anello di interesse, limitandosi a gestire il flusso dei dati senza entrare nei dettagli delle operazioni di pulizia e validazione, che rimangono demandate alle procedure specifiche descritte nel sottocapitolo successivo. Al termine di questa fase il dataset risulta ordinato secondo la frequenza di prova e pronto per essere utilizzato nelle elaborazioni successive.

Una volta disponibili le misure sperimentali, il Main procede alla costruzione della rappresentazione complessa della risposta in frequenza. I valori di guadagno e fase misurati vengono convertiti in una forma adatta alle analisi nel dominio complesso, passaggio necessario per poter applicare in modo coerente tutte le operazioni di ricostruzione dinamica successive. In particolare, il modulo e la fase vengono trasformati in:

$$| T(j\omega) | = 10^{\frac{G_{dB}}{20}}, T(j\omega) = | T(j\omega) | e^{j\varphi}$$

Equazione 15-Modulazione e fase della funzione di trasferimento $T(j\omega)$ a partire dal guadagno in decibel

dove φ rappresenta la fase in radianti, mentre $T(j\omega)$ è la funzione di trasferimento in anello chiuso (Equazione 15). Il programma si occupa unicamente di orchestrare questa trasformazione e di memorizzare le grandezze risultanti per le fasi seguenti (Figura 44).

```
df = load_excel_clean(excel_file, sheet_name)
df = filter_loop(df, anello)

freq = df['f [Hz]'].values
gain_db = df['gain_db'].values
phase_deg = np.unwrap(np.deg2rad(df['phase_deg'].values))
phase_deg = np.rad2deg(phase_deg)

# ------------------------------------------------------------
# Ricostruzione L e T
# ------------------------------------------------------------
L_jw, T_measured_jw, T_calc_jw = compute_L_from_T(freq, gain_db, phase_deg)
T_mod = np.abs(T_calc_jw)
T_phase = np.angle(T_calc_jw, deg=True)
```

Figura 44-Caricamento dei dati e preparazione della risposta in frequenza

Il primo risultato significativo prodotto dal Main è la ricostruzione dell'anello aperto del sistema. Poiché i dati sperimentali derivano da prove effettuate in anello chiuso, il programma richiama la funzione dedicata alla conversione tra risposta in anello chiuso e anello aperto, ottenendo la funzione di trasferimento dell'anello aperto lungo tutto lo spettro di frequenze analizzato.

Parallelamente, viene ricostruita la risposta teorica in anello chiuso a partire dalla funzione di trasferimento dell'anello aperto, al fine di garantire coerenza interna tra misure sperimentali e ricostruzione. I risultati vengono quindi immediatamente esportati in un nuovo file Excel contenente sia le grandezze sperimentali originali sia le nuove variabili complesse calcolate, assicurando la conservazione di una versione verificabile dell'elaborazione (Figura 45).

```
# ------------------------------------------------------------
# Open-loop opzionale FDT
# ------------------------------------------------------------
if calculate_open_loop:
    df_ol = pd.DataFrame({
        'f [Hz]': freq,
        'L_real': np.real(L_jw),
        'L_imag': np.imag(L_jw),
        'L_gain_abs': np.abs(L_jw),
        'L_gain_db': 20*np.log10(np.abs(L_jw)),
        'L_phase_deg': np.angle(L_jw, deg=True)
    })
    save_dataframe(df_ol, f'fdt_open_loop_{anello}.xlsx')

    fig_ol = plt.figure(figsize=(10,6))
    plt.subplot(2,1,1)
    plt.semilogx(freq, 20*np.log10(np.abs(L_jw)), 'g-o')
    plt.grid(True)
    plt.ylabel("Modulo |L| [dB]")
    plt.title(f"FDT Open-Loop Attuale - {anello}")

    plt.subplot(2,1,2)
    plt.semilogx(freq, np.angle(L_jw, deg=True), 'g-o')
    plt.grid(True)
    plt.ylabel("Fase [°]")
    plt.xlabel("Frequenza [Hz]")

    plt.tight_layout()
    plot_and_save(fig_ol, f"Bode_open_loop_{anello}.png")
    plt.show()
```

Figura 45-Ricostruzione dell'anello aperto e salvataggio dei risultati intermedi

Una volta ottenuta la dinamica in anello aperto, il Main avvia la fase di analisi delle prestazioni del regolatore PI utilizzato nel sistema reale, descritto dall'Equazione 16:

$$C(s) = K_p + \frac{K_i}{s}$$

Equazione 16-Funzione di trasferimento di un regolatore proporzionale-integrale (PI)

Il programma richiama la procedura di diagnostica che calcola automaticamente la frequenza di attraversamento ω_c, il margine di fase PMe il margine di guadagno GM (Equazione 17):

$$| L(j\omega_c) | = 1, PM = 180° + \angle L(j\omega_c), GM = 20\log_{10} \frac{1}{| L(j\omega_\pi) |}$$

Equazione 17-Margini di fase e guadagno in anello aperto e condizione di ampiezza unitaria a frequenza di crossover

Questi valori permettono di valutare la stabilità e robustezza del sistema. Il Main utilizza tali risultati per produrre un riepilogo numerico e per confrontare il comportamento sperimentale con quello previsto in presenza di eventuali modifiche dei parametri di controllo (Figura 46).

```
# ------------------------------------------------------------
# Diagnostica PI e closed-loop con nuovo Kp
# ------------------------------------------------------------
f_c, pm, gm, Kp_new, Ki_new = diagnostic_PI(freq, L_jw, Kp, Ki)

df_pi = pd.DataFrame({
    'Kp_current': [Kp],
    'Ki_current': [Ki],
    'Kp_suggested': [Kp_new],
    'Ki_suggested': [Ki_new],
    'f_crossover_Hz': [f_c],
    'phase_margin_deg': [pm],
    'gain_margin_dB': [gm]
})
save_dataframe(df_pi, f'fdt_PI_suggestion_{anello}.xlsx')
```

Figura 46-Valutazione delle prestazioni del regolatore PI

Successivamente, il Main procede al calcolo delle funzioni di trasferimento del modello identificato sia in forma open-loop che close-loop. A partire dal modello razionale $G(s)$ ottenuto dal fitting della risposta in frequenza (Equazione 18):

$$G(s) = \frac{b_0 + b_1 s + \cdots + b_n s^n}{1 + a_1 s + \cdots + a_m s^m}$$

Equazione 18-Funzione di trasferimento generica di un sistema lineare, espressa tramite polinomi in s

il programma costruisce la funzione open-loop (Equazione 19):

$$L_{\text{model}}(s) = C(s) \cdot G(s)$$

Equazione 19-Funzione di trasferimento in anello aperto del modello $L_{model}(s)$

e la corrispondente funzione close-loop (Equazione 20):

$$T_{\text{model}}(s) = \frac{L_{\text{model}}(s)}{1 + L_{\text{model}}(s)}$$

Equazione 20-Funzione di trasferimento in anello chiuso del modello $T_{model}(s)$

Queste funzioni permettono di confrontare il comportamento teorico e quello sperimentale sia a livello di modulo e fase che a livello di stabilità del sistema (Figura 47).

```
# ------------------------------------------------------------
# Fit razionale G(s) closed-loop
# ------------------------------------------------------------
print("\n### CALCOLO FITTING RAZIONALE G(s) CLOSED-LOOP ###\n")
b_cl, a_cl = fit_rational_model(freq, T_calc_jw, m, n)
G_s_cl = eval_rational_bode(b_cl, a_cl, freq)

df_fit = pd.DataFrame({
    'f [Hz]': freq,
    'G_real': np.real(G_s_cl),
    'G_imag': np.imag(G_s_cl),
    'G_gain_db': 20*np.log10(np.abs(G_s_cl)),
    'G_phase_deg': np.angle(G_s_cl, deg=True)
})
for i, b in enumerate(b_cl):
    df_fit[f'b{i}'] = b
for i, a in enumerate(a_cl):
    df_fit[f'a{i}'] = a
save_dataframe(df_fit, f'fdt_G_s_fit_{anello}.xlsx')

fig_fit = plt.figure(figsize=(10,6))
plt.subplot(2,1,1)
plt.semilogx(freq, 20*np.log10(np.abs(G_s_cl)), 'r-o')
plt.grid(True)
plt.ylabel("Modulo [dB]")
plt.title(f"Bode Fitting Razionale G(s) Closed-Loop - {anello}")

plt.subplot(2,1,2)
plt.semilogx(freq, np.angle(G_s_cl, deg=True), 'r-o')
plt.grid(True)
plt.ylabel("Fase [°]")
plt.xlabel("Frequenza [Hz]")

plt.tight_layout()
plot_and_save(fig_fit, f"Bode_G_s_fit_{anello}.png")
plt.show()
```

Figura 47-Fit razionale G(s)

Il Main valuta quindi la qualità del modello calcolando $G(j\omega)$ lungo le stesse frequenze sperimentali (Equazione 21):

$$G(j\omega) = \frac{N(j\omega)}{D(j\omega)}$$

Equazione 21-Funzione di trasferimento $G(j\omega)$ in forma frazionaria numeratore/denominatore

e confrontando i diagrammi di Bode ottenuti dai dati misurati e dal modello identificato. Questi confronti consentono di verificare l'accuratezza del fitting e di garantire che il modello rappresenti fedelmente la dinamica osservata.

Nella parte finale dello script, il Main coordina l'analisi strutturale del modello ottenuto. Vengono calcolati poli e zeri dei modelli open-loop e close-loop (Equazione 22):

$$D(s) = 1 \text{ (poli)}, N(s) = 0 \text{ (zeri)}$$

Equazione 22-Poli e zeri del sistema, dati dalle condizioni $D(s) = 0$, $N(s) = 0$

e generate automaticamente le mappe poli-zeri e altre figure riepilogative per entrambe le configurazioni, fornendo una visualizzazione chiara della stabilità e delle dinamiche intrinseche del sistema. Tutti i file e le figure prodotti vengono salvati in modo centralizzato, garantendo un'organizzazione coerente degli output all'interno delle directory di lavoro (Figura 48).

```
# Zero / Pole map dal fit
zeros_fit = np.roots(b_cl[::-1])
poles_fit = np.roots(a_cl[::-1])

# Mappa classica Re vs Im
fig_zp = plt.figure(figsize=(6,6))
plt.scatter(np.real(zeros_fit), np.imag(zeros_fit), label='Zeri (fit)')
plt.scatter(np.real(poles_fit), np.imag(poles_fit), marker='x', label='Poli (fit)')
plt.axhline(0); plt.axvline(0)
plt.grid(True)
plt.xlabel("Re{s}")
plt.ylabel("Im{s}")
plt.title("Zero / Pole map – modello matematico G(s)")
plt.legend()
plot_and_save(fig_zp, f"Zero_Pole_map_fit_{anello}.png")
plt.show()
```

Figura 48-Analisi poli-zeri e generazione automatica degli output grafici

Nel suo complesso, il Main realizza una pipeline di elaborazione completamente automatizzata che collega direttamente i diagrammi di Bode sperimentali alla determinazione delle funzioni di trasferimento open e close-loop e dei modelli $G(s)$. La sequenza operativa comprende: importazione dei dati, costruzione della risposta complessa, ricostruzione dell'anello aperto, valutazione del controllore, calcolo di $G(s)$ open-loop e close-loop, identificazione del modello dinamico e generazione dei risultati finali con mappe poli-zeri. Il file principale gestisce l'integrazione delle procedure senza occuparsi dei dettagli algoritmici, migliorando leggibilità, manutenibilità e riutilizzabilità del software, e rappresenta il collegamento finale tra misura sperimentale, modellazione teorica e analisi dei risultati.

4.2. Funzioni

Il funzionamento del programma per il calcolo delle funzioni di trasferimento si basa su un insieme di funzioni modulari dedicate alla gestione dei dati sperimentali, alla ricostruzione delle risposte in frequenza e all'identificazione del modello dinamico del sistema.

L'organizzazione del codice segue una logica fortemente modulare: ciascuna funzione realizza una singola operazione ben definita, permettendo di mantenere separati i livelli di acquisizione dati, analisi numerica e rappresentazione dei risultati. Questa scelta facilita la leggibilità del software e consente di riutilizzare le stesse routine anche in contesti differenti.

La prima fase dell'elaborazione riguarda l'importazione dei risultati sperimentali salvati in formato Excel. Tale operazione è gestita dalla funzione load_excel_clean (Figura 49), che legge il foglio selezionato e rimuove automaticamente eventuali righe vuote o incomplete presenti al termine del file. Durante le campagne sperimentali, infatti, il salvataggio progressivo dei dati può introdurre sezioni non valide che, se non eliminate, comprometterebbero le successive operazioni numeriche. La funzione effettua quindi una scansione sequenziale delle righe e interrompe la lettura quando vengono rilevate più righe vuote consecutive, restituendo un dataset coerente e pronto per l'analisi.

```python
def load_excel_clean(path, sheet):
    df = pd.read_excel(path, sheet_name=sheet, header=0)
    valid_rows = []
    empty_count = 0
    for idx, row in df.iterrows():
        if row.isnull().all():
            empty_count += 1
            if empty_count >= 2:
                break
        else:
            empty_count = 0
            valid_rows.append(idx)
    df = df.loc[valid_rows]
    print(f"[INFO] Righe valide lette: {len(df)}")
    return df
```

Figura 49-Funzione load_excel_clean per l'importazione e pulizia automatica dei dati

Una volta caricati i dati, la selezione dell'anello di corrente da analizzare viene affidata alla funzione filter_loop. Questa routine estrae dal dataset complessivo solamente i punti appartenenti all'anello desiderato (i_d oppure i_q), ordinandoli automaticamente rispetto alla frequenza di eccitazione. In questo modo tutte le elaborazioni successive operano su una sequenza frequenziale corretta, evitando errori legati all'ordine dei campioni sperimentali.

La ricostruzione delle grandezze dinamiche del sistema rappresenta il passaggio centrale dell'elaborazione. La funzione compute_L_from_T converte i valori di guadagno e fase misurati nei diagrammi di Bode nella corrispondente rappresentazione complessa della funzione di trasferimento. Il modulo espresso in decibel viene trasformato in ampiezza lineare, mentre la fase viene convertita in radianti per ottenere la risposta complessa $T(j\omega)$.

Partendo da tale grandezza viene quindi calcolato l'anello aperto mediante la relazione teorica tra risposta close-loop e open-loop. La funzione restituisce simultaneamente tre risultati: la funzione di trasferimento misurata, l'anello aperto ricostruito e la risposta close-loop ricalcolata, utilizzata come verifica interna di coerenza numerica (Figura 50).

```python
def filter_loop(df, loop):
    df_filtered = df[df['loop'].str.strip().str.lower() == loop.lower()].copy()
    if df_filtered.empty:
        raise ValueError(f"[ERRORE] Nessun dato trovato per anello {loop}")
    df_filtered = df_filtered.sort_values(by='f [Hz]')
    print(f"[INFO] Punti selezionati per anello {loop}: {len(df_filtered)}")
    return df_filtered

def compute_L_from_T(freq, gain_db, phase_deg):
    T_mod = 10**(gain_db / 20)
    T_phase_rad = np.deg2rad(phase_deg)
    T_jw = T_mod * np.exp(1j * T_phase_rad)
    L_jw = T_jw / (1 - T_jw)
    T_calc = L_jw / (1 + L_jw)
    return L_jw, T_jw, T_calc
```

Figura 50-Filtro e ricostruzione numerica della funzione di trasferimento e dell'anello aperto

L'analisi delle prestazioni del controllore PI è implementata nella funzione diagnostic_PI. Questa procedura identifica automaticamente la frequenza di attraversamento in ampiezza unitaria dell'anello aperto e calcola i principali indicatori di stabilità, tra cui margine di fase e margine di guadagno.

Oltre alla sola valutazione diagnostica, la funzione fornisce anche una stima preliminare di nuovi parametri del regolatore, ottenuta tramite una semplice regola di scalatura del guadagno proporzionale. Lo scopo non è realizzare un algoritmo di tuning automatico completo, ma offrire uno strumento rapido di interpretazione dei risultati sperimentali e di verifica immediata della bontà della taratura attuale.

Le operazioni di gestione dei risultati vengono invece affidate a funzioni dedicate al salvataggio automatico dei dati e delle figure. La funzione save_dataframe crea, se necessario, la directory di output e salva i risultati numerici in formato Excel, garantendo la tracciabilità delle elaborazioni effettuate. Analogamente, plot_and_save consente di esportare automaticamente i grafici generati durante l'analisi, mantenendo una struttura ordinata delle cartelle (Figura 51).

```python
def save_dataframe(df, filename, output_dir="output_fdt"):
    os.makedirs(output_dir, exist_ok=True)
    path = os.path.join(output_dir, filename)
    df.to_excel(path, index=False)
    print(f"[SALVATO] {path}")

def plot_and_save(fig, filename, output_dir="output_fdt"):
    os.makedirs(output_dir, exist_ok=True)
    path = os.path.join(output_dir, filename)
    fig.savefig(path, dpi=150)
    print(f"[PLOT SALVATO] {path}")
```

Figura 51-Funzioni di salvataggio automatico dei risultati numerici e dei grafici

Una delle funzionalità più rilevanti del software è l'identificazione di un modello matematico equivalente del sistema reale. Questa operazione è realizzata dalla funzione fit_rational_model, che esegue un fitting razionale della risposta in frequenza misurata. La funzione costruisce un sistema lineare complesso basato sui valori della variabile di Laplace $s = j\omega$ e determina i coefficienti del numeratore e del denominatore della funzione di trasferimento mediante una soluzione ai minimi quadrati complessi. Il risultato è una funzione di trasferimento continua capace di approssimare il comportamento dinamico osservato sperimentalmente, permettendo il passaggio da una descrizione puramente numerica a un modello analitico utilizzabile in successive analisi di controllo.

La validazione del modello identificato è affidata alla funzione eval_rational_bode, che valuta la funzione di trasferimento stimata lungo lo stesso vettore di frequenze sperimentali (Figura 52). In questo modo è possibile confrontare direttamente diagrammi di Bode misurati e diagrammi ottenuti dal modello, verificando la qualità dell'approssimazione e individuando eventuali discrepanze dinamiche.

```python
def fit_rational_model(freq, H_jw, m, n):
    w = 2 * np.pi * freq
    s = 1j * w
    A = np.zeros((len(s), m+n+1), dtype=np.complex128)
    for i in range(n+1):
        A[:, i] = s**i
    for i in range(1, m+1):
        A[:, n+i] = -H_jw * s**i
    x, _, _, _ = np.linalg.lstsq(A, H_jw, rcond=None)
    b_coeff = x[:n+1]
    a_coeff = np.hstack(([1.0], x[n+1:]))
    return b_coeff, a_coeff

def eval_rational_bode(b_coeff, a_coeff, freq):
    w = 2*np.pi*freq
    s = 1j*w
    num = np.polyval(b_coeff[::-1], s)
    den = np.polyval(a_coeff[::-1], s)
    return num / den
```

Figura 52-Valutazione del modello razionale e confronto con i dati sperimentali

Nel loro insieme, le funzioni implementate costituiscono una pipeline coerente di elaborazione che parte dal dataset sperimentale grezzo e conduce alla determinazione della funzione di trasferimento del sistema controllato. La suddivisione in moduli indipendenti consente di mantenere il codice facilmente estendibile e garantisce continuità metodologica con gli strumenti di analisi sviluppati nei capitoli precedenti, preservando allo stesso tempo la flessibilità necessaria per l'analisi di dati provenienti da un sistema reale.

5. Valutazione delle prestazioni del sistema

Nel presente capitolo viene affrontata la valutazione complessiva delle prestazioni del sistema di azionamento, con l'obiettivo di verificare la coerenza tra il comportamento teoricamente previsto, quello simulato e quello effettivamente osservato in laboratorio. Nei capitoli precedenti sono stati infatti sviluppati e descritti gli strumenti software necessari all'intero processo di analisi: sono stati presentati i programmi Python realizzati per l'automatizzazione delle prove e per la generazione degli anelli di corrente, ottenuti attraverso la comunicazione inizialmente con l'ambiente di simulazione PLECS e successivamente con il microcontrollore impiegato per il controllo del motore elettrico stepper reale. È stato inoltre illustrato il programma di post-process dedicato all'elaborazione dei dati acquisiti, mediante il quale è possibile ricostruire le funzioni di trasferimento del sistema in anello aperto e in anello chiuso, determinare la corrispondente funzione $G(s)$ e calcolare poli e zeri del modello dinamico identificato. Alla luce di tali strumenti, risulta quindi necessario analizzare in modo sistematico i risultati ottenuti, così da validare il modello sviluppato e interpretare eventuali differenze tra simulazioni e comportamento reale del sistema.

A tal fine, il capitolo è organizzato in più sottosezioni dedicate alla presentazione separata delle diverse fonti di informazione disponibili: inizialmente vengono riportati i risultati numerici ricavati dai datasheet e dalle specifiche nominali del sistema, successivamente i risultati ottenuti mediante simulazioni in ambiente PLECS e infine i risultati sperimentali derivanti dalle prove reali sul banco di prova. Nell'ultima parte del capitolo tali risultati verranno messi direttamente a confronto, permettendo una valutazione critica delle differenze osservate e l'individuazione delle principali cause fisiche e modellistiche responsabili degli scostamenti.

Nel contesto degli azionamenti elettrici, la valutazione delle prestazioni si basa principalmente sull'analisi dell'anello di corrente, che rappresenta il livello di controllo più interno e più rapido del sistema. L'anello di corrente ha il compito di regolare in modo preciso la corrente di statore, e di conseguenza direttamente la coppia elettromagnetica prodotta dal motore, garantendo rapidità di risposta, stabilità e robustezza rispetto a disturbi e variazioni parametriche. Poiché le prestazioni dinamiche complessive dell'azionamento dipendono fortemente da questo anello interno, la sua caratterizzazione costituisce un passaggio fondamentale per la validazione del controllo.

Per analizzare quantitativamente il comportamento dell'anello di corrente vengono utilizzati strumenti tipici dell'analisi nei sistemi di controllo in frequenza. In particolare, attraverso i diagrammi di Bode è possibile osservare come il sistema amplifichi o attenui segnali sinusoidali alle diverse frequenze, ricavando informazioni essenziali quali banda passante, margini di stabilità, rapidità di risposta e capacità di reiezione dei disturbi. I diagrammi sperimentali acquisiti mediante le procedure automatiche sviluppate vengono quindi elaborati in fase di post-process per ricostruire le funzioni di trasferimento del sistema, sia in anello aperto sia in anello chiuso. Il calcolo delle funzioni di trasferimento e della corrispondente funzione $G(s)$ consente infatti di passare dalla semplice osservazione dei dati misurati a una rappresentazione matematica compatta e confrontabile della dinamica del motore controllato.

Questa rappresentazione risulta particolarmente utile perché permette di confrontare direttamente risultati provenienti da fonti differenti: modelli teorici, simulazioni numeriche e misure reali; utilizzando un linguaggio comune basato su poli, zeri e risposta in frequenza. Inoltre, la conoscenza della funzione di trasferimento rende possibile valutare in modo oggettivo le prestazioni del regolatore PI progettato, verificando se i requisiti di progetto relativi a stabilità, velocità di risposta e robustezza siano effettivamente soddisfatti dal sistema reale. L'analisi svolta nel capitolo costituisce quindi il collegamento finale tra modellazione teorica, sviluppo degli strumenti software, identificazione sperimentale e validazione delle prestazioni dell'azionamento, permettendo di dimostrare l'efficacia del metodo adottato e di interpretare in maniera critica i risultati ottenuti.

Prima di procedere con l'analisi dettagliata dei risultati presentati nei sottocapitoli successivi, è opportuno definire le condizioni sperimentali comuni adottate durante tutte le attività di prova e simulazione. Gli esperimenti sono stati infatti condotti utilizzando tre diverse taglie di motori stepper, selezionate con l'obiettivo di valutare il comportamento del metodo di identificazione e del sistema di controllo su attuatori caratterizzati da differenti parametri elettrici e dinamici. Le principali caratteristiche nominali dei motori impiegati; quali resistenza di fase, induttanza, corrente nominale e coppia dichiarata, sono riassunte nella Tabella 1, che costituisce il riferimento per l'interpretazione dei risultati presentati nel seguito.

PARAMETRI MOTORI				
PARAMETRI		GRANDE	REF.	PICCOLO
Corrente nominale di fase	$I_N[A_{pk}]$	10	9	9
Denti del rotore	N_r	50	50	50
Resistenza dello statore	$R_S[\Omega]$	0,23	0,16	0,12
Induttanza dello statore	$L_0[\mathrm{mH}]$	2,3	1,5	0,85
Costante di coppia	$k_M[\mathrm{Nm/A}]$	0,8	0,51	0,25
Coppia nominale	$T_L[\mathrm{Nm}]$	7,2	4,2	2,1
Velocità nominale (Vdc = 70V)	$\omega_N[\mathrm{rad/s}]$	30	50	100
Velocità massima	$\omega_{max}[\mathrm{rad/s}]$	314	314	314
Peso	M[Kg]	5,2	3	2

Tabella 1-Caratteristiche motori utilizzati

Tutte le prove, sia in simulazione sia in laboratorio, sono state eseguite mantenendo condizioni operative il più possibile uniformi, così da garantire la confrontabilità diretta dei risultati ottenuti. In particolare, l'alimentazione dei motori è stata fissata a 24 V con una corrente massima limitata a 1 A. Tale scelta non rappresenta il punto di funzionamento nominale massimo dei motori utilizzati, ma è stata imposta dai limiti degli alimentatori disponibili in laboratorio e dalla necessità di operare in condizioni sicure e ripetibili durante l'intera campagna sperimentale.

Le prove di identificazione in frequenza sono state realizzate imponendo riferimenti sinusoidali di corrente a diverse frequenze, variate nell'intervallo compreso tra 100 Hz e 1500 Hz. Tuttavia, ai fini dell'analisi finale, sono stati considerati affidabili soltanto i risultati fino a 1000 Hz. Per frequenze superiori, infatti, nelle prove effettuate sui motori reali si è osservata la presenza di rumore sperimentale significativo, attribuibile principalmente a limitazioni hardware, fenomeni di quantizzazione delle misure e riduzione del rapporto segnale-rumore alle alte frequenze. Di conseguenza, tali dati non risultavano sufficientemente rappresentativi della dinamica effettiva del sistema e sono stati esclusi dalla fase di valutazione delle prestazioni.

L’ampiezza delle sinusoidi di riferimento impiegate durante i test è stata fissata a 0.3 A. Anche questa scelta deriva direttamente dai limiti di alimentazione disponibili: un’ampiezza maggiore avrebbe comportato richieste di corrente incompatibili con le restrizioni imposte dagli alimentatori utilizzati. Per garantire la massima coerenza metodologica, le stesse condizioni operative di tensione di alimentazione, limitazione di corrente e ampiezza del segnale sinusoidale sono state replicate anche nelle simulazioni eseguite in ambiente PLECS, permettendo così un confronto diretto tra risultati simulati e sperimentali senza introdurre differenze dovute al punto di funzionamento del sistema.

È importante osservare che l’utilizzo di un’alimentazione fortemente limitata rispetto alle reali capacità operative dei motori stepper comporta inevitabilmente una significativa attenuazione delle risposte dinamiche misurate. Le ampiezze delle correnti e delle grandezze osservate risultano pertanto inferiori rispetto a quelle ottenibili in condizioni nominali di funzionamento. Tale scelta operativa, tuttavia, non rappresenta una limitazione per gli obiettivi del lavoro di tesi. L’interesse principale non era infatti la determinazione delle prestazioni massime del sistema né l’ottimizzazione fine di guadagni e sfasamenti del regolatore, bensì la validazione della metodologia sviluppata e dei programmi Python realizzati per l’estrazione automatica degli anelli di corrente e dei diagrammi di Bode a partire da simulazioni e misure sperimentali.

Operare con livelli di eccitazione ridotti ha consentito di mantenere il sistema in un regime sostanzialmente lineare, condizione particolarmente favorevole per verificare la correttezza delle procedure automatiche di acquisizione, elaborazione e post-process dei dati. In questo contesto, la coerenza tra le funzioni di trasferimento ricavate, le rappresentazioni in frequenza ottenute e il comportamento atteso del sistema assume un ruolo più rilevante rispetto ai valori assoluti delle grandezze fisiche coinvolte. L’attenzione è quindi rivolta principalmente alla capacità degli strumenti software sviluppati di ricostruire in modo affidabile la dinamica del sistema controllato, garantendo ripetibilità delle analisi e confrontabilità diretta tra risultati provenienti da datasheet, simulazioni PLECS ed esperimenti sui motori reali.

5.1. Risultati numerici da datasheet

Nel primo passo della valutazione delle prestazioni vengono analizzati i risultati teorici ricavati direttamente dai dati di targa dei motori stepper utilizzati nelle prove sperimentali. Questa fase rappresenta il riferimento iniziale dell'intero confronto, poiché consente di stimare il comportamento dinamico atteso del sistema sulla base delle sole caratteristiche elettriche fornite dal costruttore, prima di introdurre gli effetti del controllo, delle simulazioni e delle misure reali.

A partire dai valori nominali di resistenza e induttanza di fase riportati nei datasheet, ciascun motore può essere modellato, dal punto di vista elettrico, mediante il classico modello RL equivalente dell'avvolgimento di statore. In tale approssimazione la dinamica della corrente è descritta da una funzione di trasferimento del primo ordine tra tensione applicata e corrente di fase, esprimibile nella forma generale mostrata dall'Equazione 23:

$$G(s) = \frac{1}{Ls + R}$$

Equazione 23-Funzione di trasferimento RL di una fase del motore

dove L rappresenta l'induttanza di fase e R la resistenza dell'avvolgimento. Questo modello descrive esclusivamente la dinamica elettrica interna del motore, trascurando gli effetti del regolatore di corrente, della discretizzazione digitale e delle non idealità dell'azionamento, che verranno invece considerati nelle successive analisi tramite simulazioni e prove sperimentali.

Applicando tale modello ai tre motori stepper di diversa taglia, indicati come Grande, Riferimento e Piccolo, sono stati calcolati i principali parametri dinamici teorici riportati in Tabella 2. In particolare, per ciascun motore sono stati determinati:

- la funzione di trasferimento elettrica;
- la costante di tempo elettrica $\tau_e = L/R$;
- la posizione del polo elettrico $\omega_p = R/L$;

- la corrispondente frequenza di polo $f_p = \omega_p/2\pi$;
- il guadagno statico $G(0) = 1/R$;
- il livello del diagramma di Bode a bassa frequenza.

I risultati mostrano come i tre motori presentino dinamiche elettriche simili ma non identiche: all'aumentare della taglia del motore cresce generalmente l'induttanza di fase, con conseguente variazione della costante di tempo elettrica e dello smorzamento della risposta in corrente. I poli elettrici risultano compresi indicativamente tra 100 e 141 rad/s (circa 16–22 Hz), evidenziando che la dinamica elettrica dell'avvolgimento è relativamente lenta rispetto alle frequenze di eccitazione investigate durante i test in frequenza.

Il guadagno statico teorico, inversamente proporzionale alla resistenza di fase, aumenta passando dal motore di taglia maggiore a quello più piccolo, indicando una maggiore sensibilità della corrente rispetto alla tensione applicata. Analogamente, il calcolo della corrente a regime evidenzia valori teorici molto elevati se si considera l'applicazione diretta della tensione nominale di 24 V al modello puramente resistivo-induttivo. Tali valori non rappresentano condizioni operative reali dell'azionamento, poiché nella pratica la corrente è limitata dall'elettronica di controllo e dal regolatore PI; essi vengono quindi utilizzati esclusivamente come riferimento teorico utile alla caratterizzazione del modello elettrico ideale.

È importante sottolineare che, in questa fase, non vengono calcolati anelli di corrente, diagrammi di Bode completi né funzioni di trasferimento in anello aperto o chiuso. I dati di datasheet consentono infatti di descrivere soltanto la dinamica intrinseca dell'avvolgimento del motore, mentre la ricostruzione dell'anello di controllo richiede necessariamente l'inclusione del regolatore, dell'inverter e della catena di misura, elementi accessibili esclusivamente tramite simulazioni in ambiente PLECS e attraverso le prove sperimentali reali.

L'analisi basata sui datasheet costituisce quindi il livello più teorico del confronto: essa fornisce una stima preliminare dei parametri dinamici fondamentali del sistema e rappresenta il punto di partenza rispetto al quale verranno successivamente confrontati i risultati ottenuti dalle simulazioni e dalle misure di laboratorio, permettendo di evidenziare l'influenza del controllo e delle non idealità dell'azionamento reale.

	DATI DI TARGA			
	GRANDE		RIFERIMENTO	PICCOLO
funzione di trasferimento	G(s) = 1/(Ls+R)	G(s) = 1/((3,3*10^-3s)+0,23)	G(s) = 1/((1,5*10^-3s)+0,16)	G(s) = 1/((8,5*10^-4s)+0,12)
costante di tempo elettrica	τe = L/R	τe = (2,3*10^-3)/0,23 = 0,01s = 10ms	τe = (1,5*10^-3)/0,16 = 9,375*10^-3s = 9,4ms	τe = (8,5*10^-4)/0,12 = 7,08*10^-3s = 7,1ms
polo elettrico	ωp = R/L	ωp = 0,23 / (2,3*10^-3) = 100rad/s	ωp = 0,16 / (1,5*10^-3) = 106,7rad/s	ωp = 0,12 / (8,5*10^-4) = 141rad/s
		fp = ωp/2π = 15,9Hz	fp = ωp/2π = 17Hz	fp = ωp/2π = 22,5Hz
guadagno statico	G(0) = 1/R	G(0) = 1/0,23 =4,35A/V	G(0) = 1/0,16 =6,25A/V	G(0) = 1/0,12 =8,33A/V
corrente a regime	iss = V/R	iss = 24/0,23 = 104A	iss = 24/0,16 = 150A	iss = 24/0,12 = 200A
bode a bassa frequenza		20 log (4,35) = 12,8dB	20 log (6,25) = 15,9dB	20 log (8,33) = 18,4dB
funzione di trasferimento anello chiuso		$\frac{i_d(s)}{i_{ref}(s)} = \frac{k_p s + k_i}{Ls^2 + (R + k_p)s + k_i}$		

Tabella 2-Parametri dinamici teorici ricavati dai datasheet dei tre motori stepper

5.2. Risultati da simulazioni PLECS

In questo capitolo vengono presentati e analizzati i risultati ottenuti dalle simulazioni condotte in ambiente PLECS, eseguite e automatizzate mediante lo script Python descritto precedentemente nell'apposito capitolo. L'obiettivo principale delle prove è la caratterizzazione dinamica degli anelli di corrente del sistema di controllo, con particolare riferimento all'anello i_d. L'andamento dell'anello i_q risulta infatti qualitativamente simile a quello della componente i_d, differenziandosi principalmente per uno sfasamento della risposta; per tale motivo l'analisi è stata focalizzata sulla sola componente i_d, ritenuta rappresentativa del comportamento complessivo del controllo di corrente.

Attraverso l'applicazione di eccitazioni sinusoidali a diverse frequenze è stato possibile estrarre i diagrammi di Bode degli anelli di corrente, strumenti fondamentali per la valutazione delle prestazioni dinamiche del sistema di controllo sviluppato. I risultati ottenuti sono successivamente elaborati mediante il software di post-processing, che consente di ricavare le corrispondenti funzioni di trasferimento $G(s)$, insieme alla posizione di poli e zeri, permettendo una descrizione completa del comportamento dinamico del sistema.

Le simulazioni sono state organizzate considerando le tre differenti taglie di motore stepper. Per ciascun modello sono state riprodotte condizioni operative il più possibile aderenti a quelle previste nelle prove sperimentali reali, mantenendo identiche condizioni di alimentazione, un ritardo dell'inverter coerente con l'elettronica hardware e i parametri elettrici specifici dei motori analizzati, quali resistenza e induttanza di statore e costante di coppia. Ogni motore è stato inoltre testato per diversi valori dei parametri del regolatore PI, K_p e K_i, con l'obiettivo di valutare l'influenza della taratura del controllore sulla dinamica dell'anello di corrente e sulla banda di regolazione ottenibile. L'analisi che segue presenta quindi un confronto sistematico tra le diverse configurazioni considerate.

Per comprendere meglio i risultati, è opportuno richiamare brevemente il significato dei parametri di guadagno e fase che caratterizzano i diagrammi di Bode. Quando un sistema dinamico viene eccitato mediante un segnale sinusoidale di ingresso, l'uscita assume anch'essa un andamento sinusoidale alla stessa frequenza, ma generalmente con ampiezza differente e con uno sfasamento temporale rispetto al segnale applicato. La Figura 53 mostra in maniera qualitativa tale comportamento, evidenziando la sinusoide di ingresso e quella di uscita.

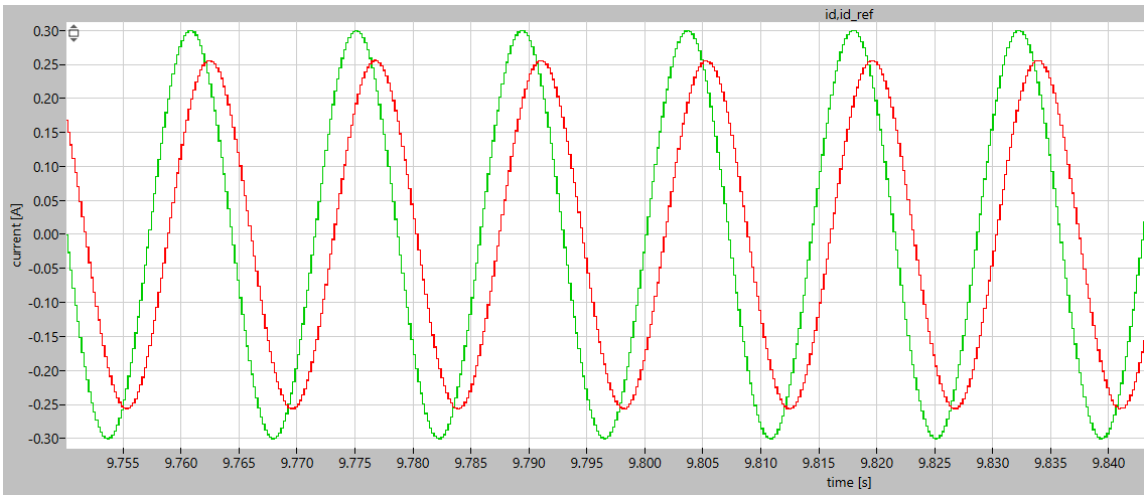

Figura 53-Esempio di sinusoide di ingresso e di uscita (riferimento in verde e misura in rosso)

Il guadagno rappresenta il rapporto tra l'ampiezza del segnale di uscita e quella del segnale di ingresso ed è comunemente espresso in decibel; esso fornisce un'indicazione diretta della capacità dell'anello di controllo di amplificare o attenuare una perturbazione a una determinata frequenza. La fase, invece, descrive il ritardo o anticipo temporale dell'uscita rispetto all'ingresso ed è espressa in gradi, essendo strettamente legata alla stabilità del sistema e alla rapidità della risposta dinamica. Variando progressivamente la frequenza della sinusoide applicata e misurando guadagno e fase corrispondenti, è possibile costruire il diagramma di Bode dell'anello di controllo, che costituisce lo strumento principale per l'analisi in frequenza adottata nel presente lavoro.

L'identificazione degli anelli di corrente è stata effettuata applicando al riferimento di corrente una perturbazione sinusoidale di ampiezza costante e frequenza variabile. Per ciascun valore di frequenza sono stati acquisiti i segnali di riferimento e di corrente misurata, successivamente elaborati tramite lo script Python dedicato. Il processo di elaborazione consente di estrarre automaticamente il guadagno dell'anello di corrente, lo sfasamento tra ingresso e uscita, l'indice di coerenza della misura e la funzione di trasferimento equivalente del sistema. I dati ottenuti vengono quindi organizzati sotto forma di diagrammi di Bode e tabelle riassuntive contenenti i principali parametri dinamici, che costituiscono la base per il confronto tra i diversi motori e le differenti tarature del controllore.

L’attività di simulazione è stata sviluppata considerando tre diverse taglie di motore stepper, ciascuna contraddistinta da specifici parametri elettrici e dinamici. Al fine di garantire un confronto omogeneo e metodologicamente corretto, per tutti i motori sono state mantenute invariate la struttura dell’anello di controllo, la strategia di identificazione e le condizioni operative, inclusi i limiti di tensione e corrente di alimentazione. Tale scelta ha permesso di analizzare in modo sistematico, da un lato, l’influenza della taglia del motore sulla dinamica dell’anello di corrente e, dall’altro, l’effetto della variazione dei parametri del regolatore PI (K_p e K_i) sulle prestazioni dell’anello i_d, isolando le variabili di interesse e rendendo i risultati direttamente confrontabili. Il medesimo approccio verrà successivamente adottato anche nelle prove sperimentali sul banco di laboratorio, così da mantenere piena coerenza tra simulazioni e misure reali.

Nei paragrafi seguenti vengono quindi presentati i risultati relativi all’anello di corrente i_d per ciascuna delle tre taglie di motore analizzate. Per ogni configurazione sono riportati il diagramma di Bode e i principali valori numerici estratti dalle simulazioni, utili alla valutazione della banda passante, del comportamento in frequenza e delle caratteristiche dinamiche complessive del sistema. In particolare vengono illustrati due distinti set di parametri del regolatore in retroazione, tra tutti quelli analizzati: un primo caso con K_i = 100 e K_p = 0,19, rappresentativo di una delle tarature più basse utilizzate durante le prove, e un secondo caso con K_i = 150 e K_p = 0,54, corrispondente invece alla condizione con i valori più elevati adottati. La scelta di questi due estremi consente di evidenziare i limiti operativi del regolatore e di osservare come la variazione dei guadagni influenzi la risposta dinamica dell’anello di corrente. Gli stessi due casi verranno riproposti anche nell’analisi dei risultati sperimentali, così da garantire una piena coerenza metodologica e rendere immediato il confronto tra simulazioni e prove su banco, permettendo di valutare in modo chiaro l’effetto della taglia del motore sia sugli anelli di corrente, sia sui relativi diagrammi di Bode e sulle funzioni di trasferimento ottenute.

K_i = 100, K_p = 0,19

L'analisi dei risultati di simulazione è stata condotta considerando inizialmente il caso in cui il regolatore PI dell'anello di corrente assume i valori $K_i = 100$e $K_p = 0{,}19$. Questa configurazione rappresenta una delle tarature con guadagni più bassi tra quelle utilizzate durante le prove ed è stata scelta per osservare il comportamento dinamico dell'anello di corrente in condizioni di regolazione relativamente moderata. Le simulazioni sono state eseguite per le tre diverse taglie di motore considerate nello studio, mantenendo invariate le condizioni operative e la struttura del controllo, al fine di evidenziare esclusivamente l'influenza dei parametri elettrici del motore sulla risposta dinamica del sistema.

Il confronto tra le tre configurazioni di motore è riportato nel diagramma di Bode complessivo mostrato in Figura 54 e Figura 55, nelle quali sono rappresentate simultaneamente le risposte in frequenza dei tre motori analizzati.

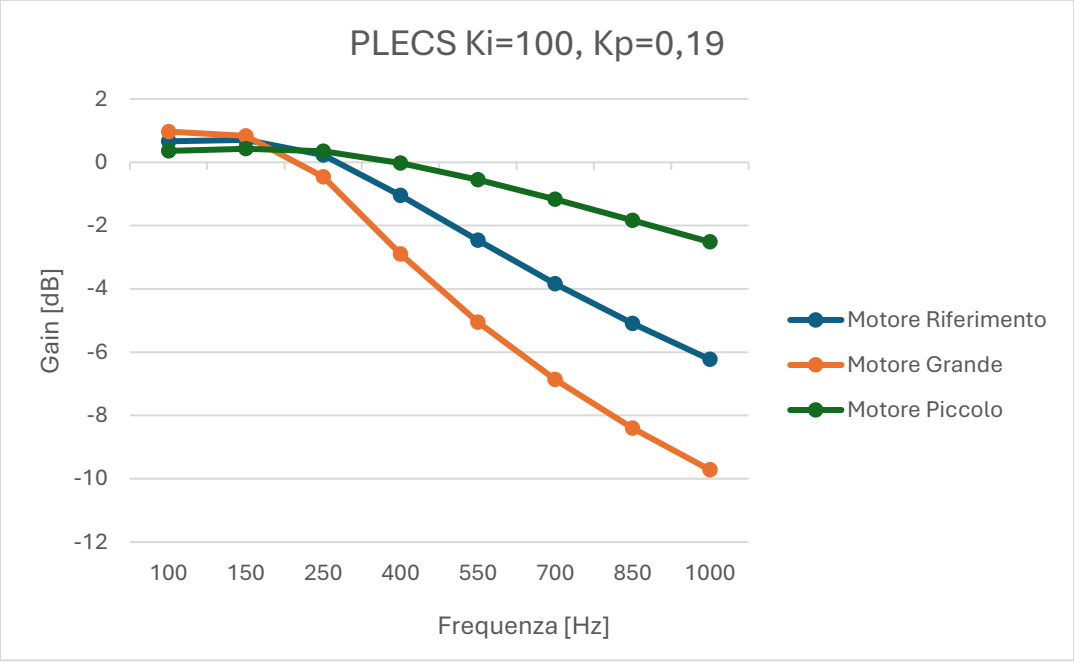

Figura 54-Grafico guadagno del diagramma di Bode (Ki=100, Kp=0.19 PLECS)

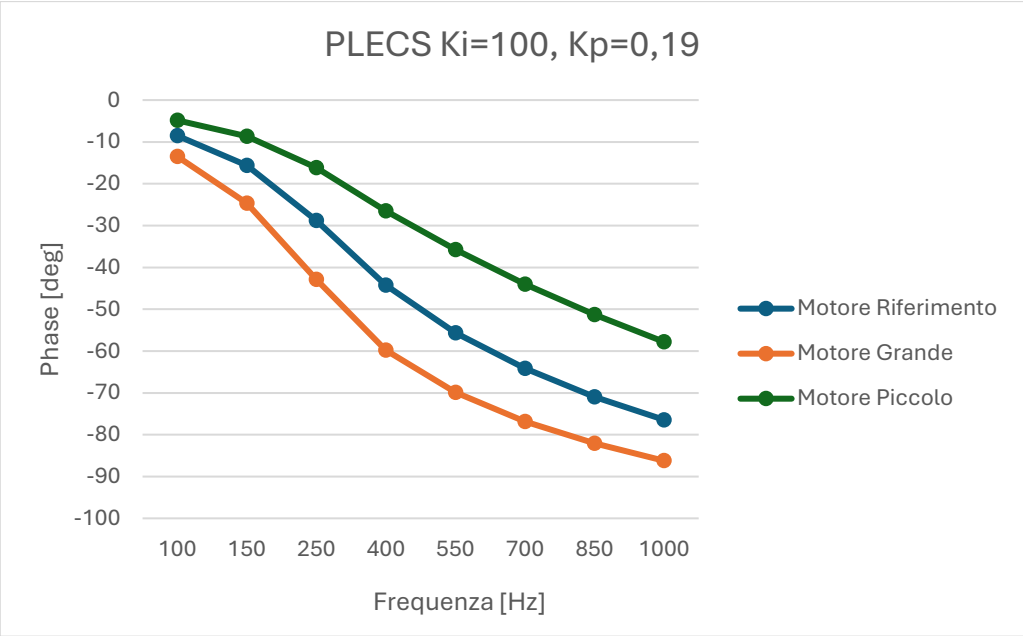

Figura 55-Grafico fase del diagramma di Bode (Ki=100, Kp=0.19 PLECS)

I valori numerici di guadagno e fase relativi alle diverse frequenze di eccitazione sono invece riportati in forma riassuntiva nella Tabella 3.

MOTORE GRANDE			**MOTORE RIFERIMENTO**			**MOTORE PICCOLO**		
f [Hz]	gain_db	phase_deg	f [Hz]	gain_db	phase_deg	f [Hz]	gain_db	phase_deg
100	0,9696853	-13,49099	100	0,662771	-8,521638	100	0,3571168	-4,840904
150	0,8362326	-24,66547	150	0,7081039	-15,6215	150	0,4267312	-8,670238
250	-0,460594	-42,88593	250	0,2295289	-28,77011	250	0,3470066	-16,19005
400	-2,893092	-59,76591	400	-1,044285	-44,23699	400	-0,02396	-26,52161
550	-5,048589	-69,90475	550	-2,466163	-55,61529	550	-0,551529	-35,75281
700	-6,861172	-76,83798	700	-3,835878	-64,18204	700	-1,170665	-43,99157
850	-8,395308	-82,04902	850	-5,090002	-70,92584	850	-1,836108	-51,29691
1000	-9,716599	-86,2224	1000	-6,229994	-76,4348	1000	-2,51757	-57,79899

Tabella 3-Tabella riassuntiva risultati Ki=100, Kp=0.19 in PLECS

Analizzando le tre curve del diagramma di Bode si osserva innanzitutto il comportamento del motore di riferimento, che rappresenta la configurazione intermedia tra le tre taglie considerate. Il diagramma di Bode dell'anello di corrente I_d mostra un comportamento tipico di un sistema a primo ordine dominato dalla dinamica elettrica dell'avvolgimento.

Alle basse frequenze il guadagno risulta prossimo a 0 dB, con un valore pari a circa 0,66 dB a 100 Hz e un lieve incremento fino a circa 0,7 dB a 150 Hz. All'aumentare della frequenza si osserva una progressiva attenuazione del guadagno, che diventa negativo oltre i 250 Hz e raggiunge circa −6,2 dB a 1000 Hz. Parallelamente, la fase mostra una progressiva diminuzione, passando da circa −8,5° a 100 Hz fino a circa −76° a 1000 Hz. Questo andamento evidenzia la presenza di una dinamica dominante che introduce un crescente ritardo di fase all'aumentare della frequenza di eccitazione.

Il motore di taglia piccola presenta invece una dinamica leggermente più rapida rispetto al motore di riferimento. Il guadagno alle basse frequenze risulta leggermente inferiore, con valori compresi tra circa 0,36 dB e 0,43 dB nel range tra 100 Hz e 150 Hz. L'attenuazione con l'aumentare della frequenza risulta tuttavia più contenuta rispetto al caso precedente: a 1000 Hz il guadagno si attesta intorno a −2,5 dB, risultando quindi significativamente meno attenuato rispetto al motore di riferimento. Anche l'andamento della fase risulta meno pronunciato, passando da circa −4,8° a 100 Hz fino a circa −58° a 1000 Hz. Questo comportamento indica una dinamica complessivamente più veloce, coerente con i parametri elettrici del motore di dimensioni ridotte, caratterizzato da una minore induttanza e quindi da una costante di tempo elettrica inferiore.

Infine, il motore di taglia grande evidenzia una dinamica più lenta rispetto agli altri due motori. Il guadagno alle basse frequenze risulta inizialmente più elevato, con un valore di circa 0,97 dB a 100 Hz, ma decresce rapidamente con l'aumentare della frequenza. Già a 250 Hz il guadagno diventa negativo e raggiunge circa −9,7 dB a 1000 Hz, evidenziando un'attenuazione significativamente maggiore rispetto agli altri casi analizzati. Anche la fase presenta una variazione più marcata, passando da circa −13,5° a 100 Hz fino a circa −86° a 1000 Hz. Questo comportamento riflette la maggiore costante di tempo elettrica del motore di dimensioni più grandi, che introduce una dinamica più lenta e quindi una maggiore attenuazione e ritardo di fase alle frequenze più elevate.

Il confronto complessivo tra i tre casi evidenzia chiaramente come la taglia del motore influenzi in modo significativo la dinamica dell'anello di corrente. Il motore di dimensioni ridotte mostra la risposta più rapida, con una minore attenuazione del guadagno e una variazione di fase più contenuta all'aumentare della frequenza.

Al contrario, il motore di dimensioni maggiori presenta una dinamica più lenta, caratterizzata da una maggiore attenuazione del guadagno e da un ritardo di fase più marcato. Il motore di riferimento si colloca in una posizione intermedia tra i due casi estremi, mostrando un comportamento dinamico coerente con i suoi parametri elettrici.

K_i = 150, K_p = 0,54

Nel secondo caso di simulazione sono stati analizzati i risultati ottenuti utilizzando nel regolatore PI valori pari a $K_i = 150$ e $K_p = 0{,}54$. L'obiettivo di questa configurazione è valutare il comportamento dell'anello di corrente in presenza di parametri del controllore più elevati rispetto al caso precedente, osservando come tali valori influenzino la risposta dinamica del sistema in termini di guadagno e sfasamento lungo l'intervallo di frequenze considerato. Analogamente a quanto fatto in precedenza, le simulazioni sono state condotte sulle tre diverse taglie di motore stepper mantenendo invariate tutte le altre condizioni operative, così da rendere direttamente confrontabili i risultati.

Il confronto tra le tre configurazioni di motore è riportato nel diagramma di Bode complessivo mostrato in Figura 56 e Figura 57, nelle quali sono rappresentate simultaneamente le risposte in frequenza dei tre motori analizzati.

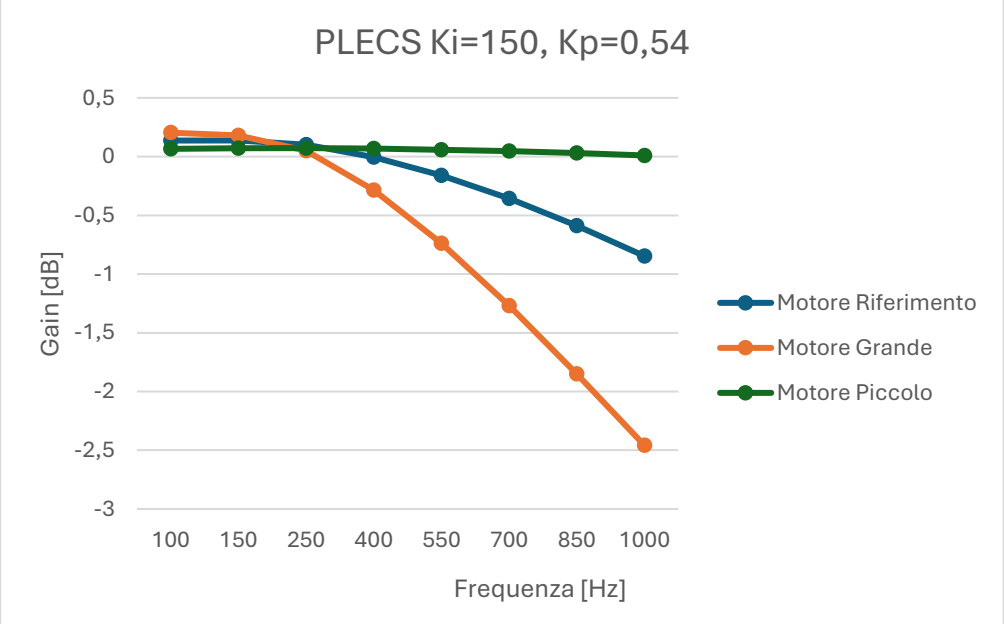

Figura 56-Grafico guadagno del diagramma di Bode (Ki=150, Kp=0.54 PLECS)

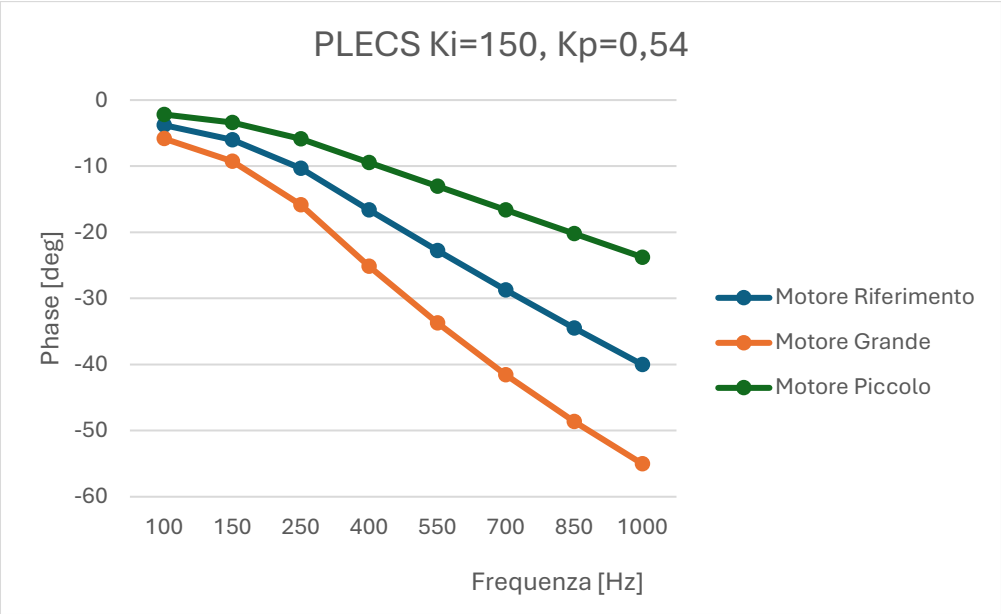

Figura 57-Grafico fase del diagramma di Bode (Ki=150, Kp=0.54 PLECS)

I valori numerici corrispondenti di modulo e fase per tutte le frequenze considerate sono invece riportati in forma riassuntiva nella Tabella 4.

MOTORE GRANDE			**MOTORE RIFERIMENTO**			**MOTORE PICCOLO**		
f [Hz]	gain_db	phase_deg	f [Hz]	gain_db	phase_deg	f [Hz]	gain_db	phase_deg
100	0,2039217	-5,836183	100	0,1368177	-3,791613	100	0,0653216	-2,179964
150	0,1815144	-9,272728	150	0,137511	-6,016481	150	0,0710348	-3,416989
250	0,0489453	-15,85151	250	0,1007795	-10,33242	250	0,0723607	-5,843724
400	-0,285821	-25,13255	400	-0,005373	-16,63333	400	0,0673446	-9,437043
550	-0,737418	-33,70173	550	-0,160402	-22,76812	550	0,0585743	-13,01707
700	-1,270158	-41,52873	700	-0,356861	-28,71696	700	0,0469116	-16,59165
850	-1,850992	-48,62141	850	-0,58812	-34,48104	850	0,0299452	-20,18083
1000	-2,458614	-55,02246	1000	-0,846405	-40,0186	1000	0,0099047	-23,76047

Tabella 4-Tabella riassuntiva risultati Ki=150, Kp=0.54 in PLECS

Analizzando le tre curve del diagramma di Bode si osserva innanzitutto il comportamento del motore di riferimento. Il guadagno si mantiene inizialmente molto vicino a 0 dB alle basse frequenze, con valori leggermente positivi attorno ai 100–150 Hz, per poi diminuire progressivamente con l'aumentare della frequenza, raggiungendo valori leggermente negativi in prossimità dei 1000 Hz. L'andamento della fase evidenzia invece un incremento graduale dello sfasamento negativo, che passa da pochi gradi alle basse frequenze fino a circa −40° nella parte alta dello spettro. Questo comportamento è tipico di un sistema con dinamica dominante del primo ordine, nel quale l'aumento della frequenza porta a una progressiva attenuazione del segnale e a un incremento dello sfasamento.

Il motore di taglia piccola mostra una risposta complessivamente più piatta in termini di guadagno lungo l'intero intervallo di frequenze analizzato. I valori rimangono infatti molto prossimi a 0 dB anche alle frequenze più elevate, con variazioni estremamente contenute. Questo comportamento indica che il sistema presenta una dinamica relativamente rapida e una minore attenuazione della risposta rispetto agli altri motori. Anche lo sfasamento cresce progressivamente con la frequenza, ma con valori generalmente inferiori rispetto a quelli osservati nel motore di riferimento. Alla frequenza massima analizzata lo sfasamento si mantiene infatti intorno a −24°, evidenziando una dinamica complessivamente più veloce e una minore inerzia elettrica.

Infine, il motore di taglia grande evidenzia una dinamica leggermente più lenta rispetto agli altri due casi. Il guadagno risulta inizialmente positivo alle frequenze più basse, ma decresce in maniera più marcata con l'aumentare della frequenza, raggiungendo valori più negativi rispetto agli altri motori alle frequenze più elevate. Questo comportamento evidenzia una maggiore attenuazione della risposta alle alte frequenze, coerente con una maggiore induttanza e quindi con una costante di tempo elettrica più elevata. Anche lo sfasamento cresce progressivamente lungo lo spettro di frequenza e raggiunge valori prossimi a −55° intorno ai 1000 Hz, risultando quindi più pronunciato rispetto agli altri motori analizzati.

Confrontando complessivamente i tre casi, emerge come la taglia del motore influenzi direttamente la dinamica dell'anello di corrente. Il motore di dimensioni più ridotte mostra infatti una risposta più rapida, con attenuazioni minori e uno sfasamento più contenuto alle frequenze elevate.

Al contrario, il motore di taglia maggiore presenta una risposta più smorzata e uno sfasamento più pronunciato, indice di una dinamica elettrica più lenta. Il motore di riferimento si colloca in una posizione intermedia tra i due comportamenti.

Questi risultati risultano coerenti con le differenze nei parametri elettrici dei motori analizzati, in particolare con i valori di resistenza e induttanza di fase, che determinano la costante di tempo elettrica del sistema. L'analisi conferma quindi che, anche mantenendo invariata la struttura del controllo e i parametri del regolatore PI, la dinamica dell'anello di corrente risulta comunque influenzata dalle caratteristiche intrinseche del motore considerato.

Funzioni di trasferimento

A partire dai dati sperimentali acquisiti durante le prove sugli anelli di corrente i_d, già analizzati mediante diagrammi di Bode, è stata condotta un'ulteriore fase di post–processing finalizzata alla caratterizzazione dinamica dei sistemi in esame. In particolare, per ciascuna configurazione di controllo considerata, sono state stimate le funzioni di trasferimento in anello chiuso $G(s)$ e, successivamente, sono stati determinati poli e zeri delle relative rappresentazioni.

L'analisi preliminare dei diagrammi di Bode, riportata nei paragrafi precedenti, ha evidenziato un comportamento dinamico riconducibile, in prima approssimazione, a un sistema dominato da un singolo polo. Per tale ragione, nella fase di identificazione della funzione di trasferimento è stato adottato un modello del primo ordine, ritenuto adeguato a descrivere la dinamica dominante dell'anello di corrente nelle diverse condizioni analizzate. Tale scelta consente di ridurre la complessità del modello mantenendo, al contempo, una rappresentazione sufficientemente accurata del comportamento osservato sperimentalmente.

È tuttavia opportuno sottolineare che il processo di identificazione è stato eseguito direttamente sulla funzione di trasferimento in anello chiuso. In tale contesto, anche imponendo una struttura del primo ordine, la risposta in frequenza risultante non necessariamente presenta un andamento strettamente monotono del modulo, come invece accade per un sistema ideale di primo ordine. Infatti, la dinamica complessiva dell'anello chiuso dipende dall'interazione tra impianto e regolatore (nel caso specifico un controllore PI), e può pertanto includere effetti secondari non pienamente riconducibili a un unico polo dominante.

In particolare, il lieve rigonfiamento osservabile nel modulo del diagramma di Bode non deve essere interpretato come la presenza effettiva di uno zero nel modello identificato, bensì come una conseguenza del procedimento di stima. L'algoritmo utilizzato, basato su una minimizzazione ai minimi quadrati nel dominio complesso, determina infatti il miglior compromesso globale tra modulo e fase rispetto ai dati sperimentali. Di conseguenza, in presenza di dinamiche reali leggermente più articolate rispetto a quelle di un sistema del primo ordine, il modello identificato può presentare deviazioni locali, pur mantenendo una struttura caratterizzata da un solo polo.

Al fine di verificare la coerenza del modello adottato tra i diversi attuatori analizzati, è stata effettuata una comparazione della funzione di trasferimento in anello chiuso nel caso $K_i = 150$ e $K_p = 0.54$ per i tre motori considerati (motore piccolo in Figura 60, Figura 61, motore di riferimento in Figura 58, Figura 59 e motore grande in Figura 62, Figura 63). Per ragioni di chiarezza espositiva e organizzazione dei contenuti, nel presente elaborato viene riportato unicamente tale caso, che coincide inoltre con la configurazione successivamente impiegata anche nell'analisi relativa al motore reale. È comunque opportuno evidenziare che un comportamento qualitativamente analogo è stato riscontrato per tutte le altre combinazioni di parametri analizzate, confermando la coerenza del modello adottato nei diversi scenari di studio.

Nei grafici riportati nelle figure successive sono stati confrontati i diagrammi di Bode ottenuti sperimentalmente con quelli ricavati a partire dalle funzioni di trasferimento $G(s)$ identificate. Tale confronto consente di valutare l'adeguatezza del modello del primo ordine nel descrivere il comportamento dinamico del sistema. In particolare, si osserva una sostanziale coincidenza tra le fasi nei due casi, evidenziando una sovrapposizione pressoché perfetta tra dati sperimentali e modello. Per quanto riguarda il modulo del guadagno, si riscontra un'elevata concordanza tra le due curve, sebbene siano presenti lievi discrepanze che impediscono una perfetta sovrapposizione. Tali scostamenti risultano coerenti con le considerazioni precedenti e sono attribuibili sia all'approssimazione intrinseca del modello adottato sia alla presenza di dinamiche non modellate nel sistema reale.

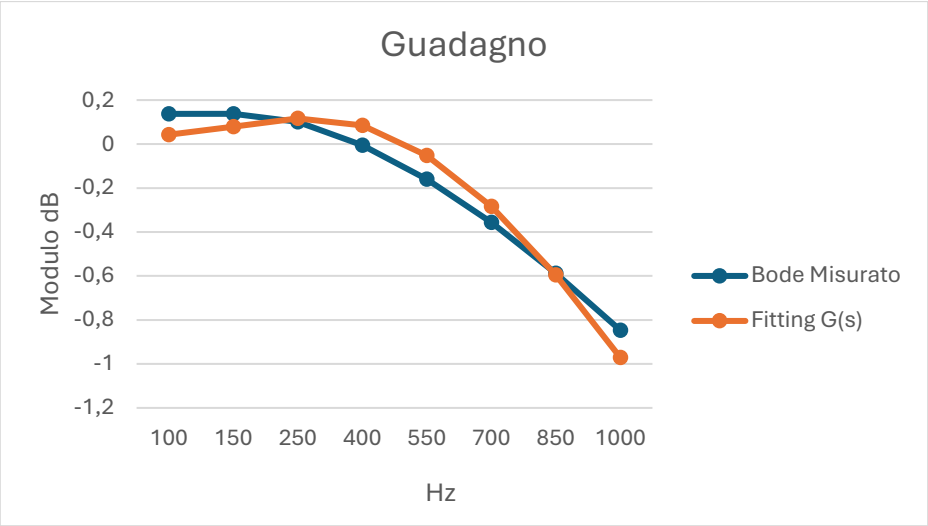

Figura 58-Confronto guadagno per Ki =150 e Kp =0.54 Motore Riferimento(PLECS)

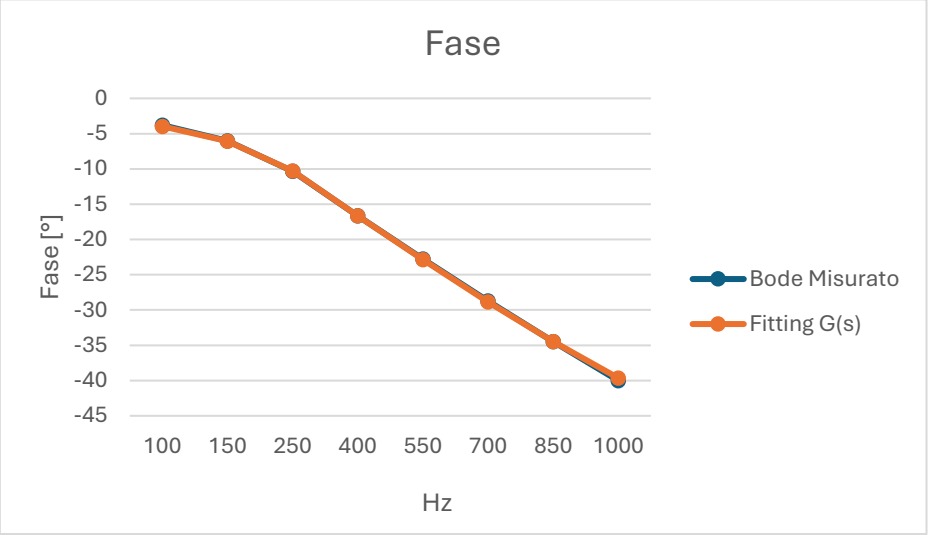

Figura 59-Confronto fase per Ki =150 e Kp =0.54 Motore Riferimento(PLECS)

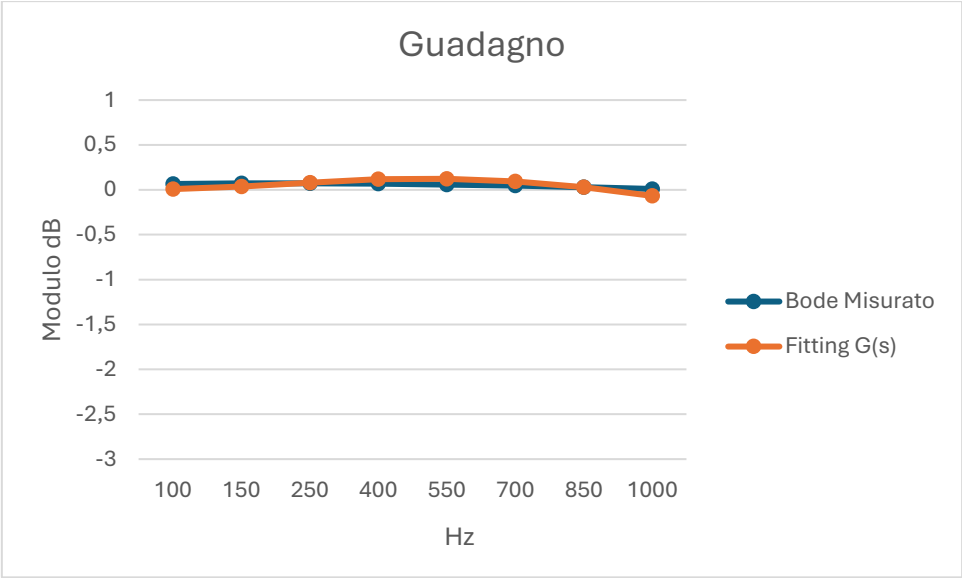

Figura 60-Confronto guadagno per Ki =150 e Kp =0.54 Motore Piccolo(PLECS)

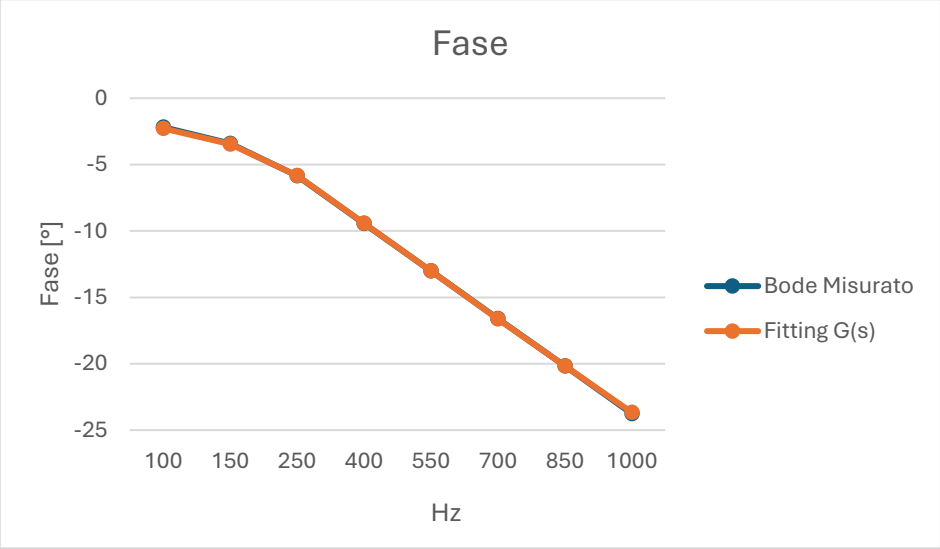

Figura 61-Confronto fase per Ki =150 e Kp =0.54 Motore Piccolo(PLECS)

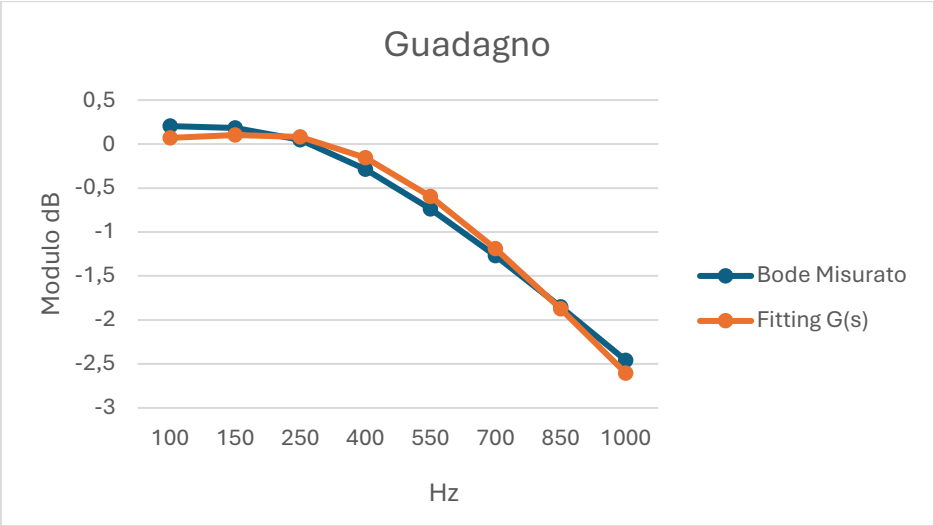

Figura 62-Confronto guadagno per Ki =150 e Kp =0.54 Motore Grande (PLECS)

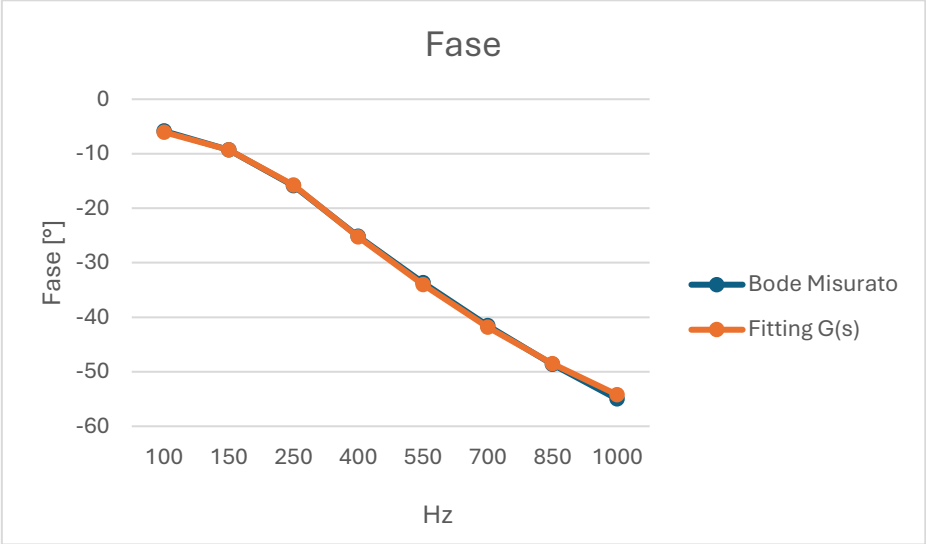

Figura 63-Confronto fase per Ki =150 e Kp =0.54 Motore Grande (PLECS)

Come si può osservare, l'andamento delle funzioni di trasferimento risulta molto simile per tutti e tre i motori, confermando che la struttura dinamica del sistema è sostanzialmente la stessa; una parte iniziale maggiore di zero che aumenta leggermente a frequenze basse, per poi calare in modo repentino all'aumentare della frequenza di eccitazione. Le differenze tra i tre casi sono principalmente riconducibili a variazioni di guadagno e di fase, già evidenziate nell'analisi dei diagrammi di Bode degli anelli di corrente. Tali differenze sono legate alle diverse caratteristiche elettromeccaniche dei motori considerati, ma non modificano in maniera significativa la forma complessiva della risposta dinamica.

Una volta ottenute le funzioni di trasferimento, sono stati calcolati poli e zeri per ciascuna configurazione di parametri del controllore. Ai fini della stabilità del sistema, l'attenzione è stata posta in particolare sulla parte reale dei poli, poiché la condizione di stabilità richiede che essa sia strettamente negativa.

Dai risultati ottenuti emerge che tutti i poli presentano parte reale negativa, indicando che il sistema risulta stabile per tutte le combinazioni di parametri considerate. Inoltre, l'analisi mostra che i poli assumono valori complessi coniugati, caratterizzati da una parte reale negativa e da una componente immaginaria variabile in funzione dei parametri del controllore.

Per evidenziare l'andamento della parte reale dei poli al variare del guadagno proporzionale K_p, sono stati realizzati i grafici riportati nelle Figura 64 e Figura 65, relativi rispettivamente ai casi $K_i = 100$ e $K_i = 150$.

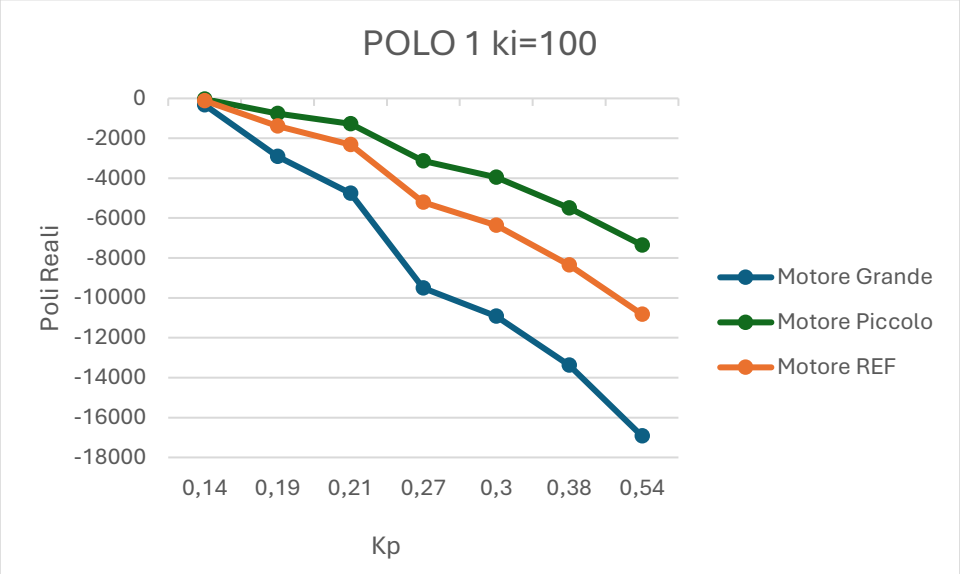

Figura 64-Andamento della parte reale dei poli per Ki = 100

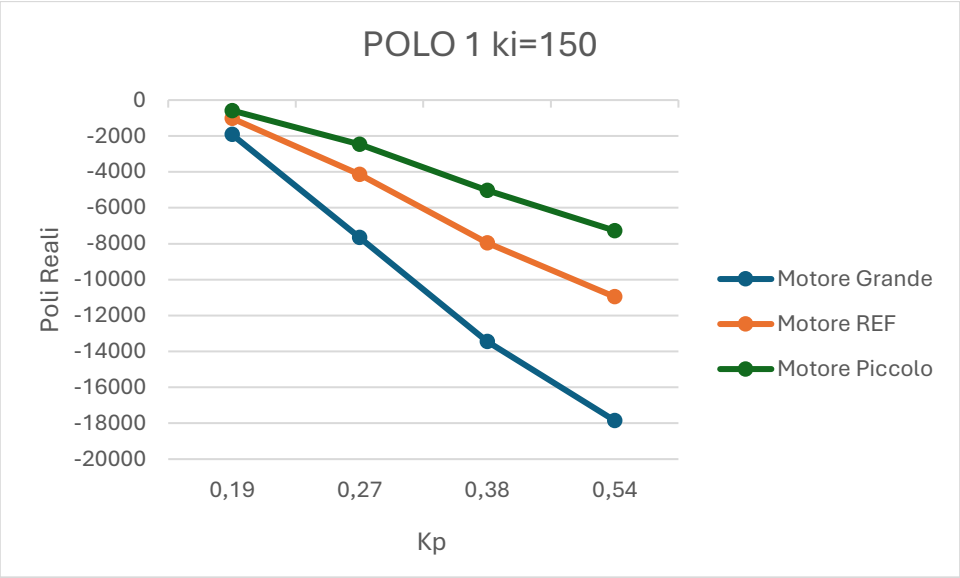

Figura 65-Andamento della parte reale dei poli per Ki = 150

Dall'osservazione dei grafici è possibile individuare alcune tendenze comuni ai tre motori. In particolare, all'aumentare del guadagno proporzionale K_p la parte reale dei poli tende a spostarsi verso valori più negativi, indicando una dinamica progressivamente più rapida del sistema. Questo comportamento è coerente con l'effetto atteso dell'incremento dell'azione proporzionale nel regolatore PI, che tende ad aumentare la banda passante dell'anello di controllo.

Inoltre, si osserva che per ciascun valore di K_p il motore di riferimento presenta valori intermedi della parte reale dei poli rispetto al motore piccolo e al motore grande. Il motore piccolo tende infatti a mostrare poli con modulo maggiore, associati quindi a dinamiche più rapide, mentre il motore grande presenta valori più vicini all'origine, indicativi di una risposta leggermente più lenta. Questo comportamento è coerente con le diverse costanti elettriche e parametri fisici dei tre motori analizzati. I valori numerici esatti dell'andamento dei poli misurati sono riportati nella Tabella 5.

POLI	1 POLO					
KI/KP	MOTORE PICCOLO		MOTORE RIFERIMENTO		MOTORE GRANDE	
	Re	Im	Re	Im	Re	Im
Ki=100 Kp=0,14	-327,7241	-403,5812	-116,7011	160,63907	-37,53989	319,27823
Ki=100 Kp=0,19	-2913,508	-3135,942	-1391,582	-1519,189	-761,3948	-804,6471
Ki=100 Kp=0,21	-4751,379	-3714,372	-2311,199	-1990,419	-1280,962	-1184,578
Ki=100 Kp=0,27	-9511,18	-2587,916	-5212,127	-2129,261	-3136,773	-1644,829
Ki=100 Kp=0,3	-10919,03	-1543,038	-6365,743	-1731,019	-3950,206	-1493,847
Ki=100 Kp=0,38	-13383,12	701,48707	-8352,459	-404,0111	-5502,123	-749,6599
Ki=100 Kp=0,54	-16929,09	3258,0451	-10824,84	1395,4986	-7352,334	522,72411
Ki=150 Kp=0,19	-1918,353	-2078,346	-1016,536	-849,5429	-581,0507	-307,8789
Ki=150 Kp=0,27	-7655,141	-3603,653	-4141,676	-2149,05	-2470,519	-1399,861
Ki=150 Kp=0,38	-13446,8	-952,8935	-7950,545	-1086,908	-5022,532	-989,8802
Ki=150 Kp=0,54	-17854,33	2382,6383	-10958,8	920,0421	-7271,273	247,10746

Tabella 5-Valori dei poli PLECS

Infine, confrontando i risultati ottenuti per $K_i = 100$ e $K_i = 150$, si nota come l'aumento del guadagno integrale influenzi principalmente la posizione dei poli nel piano complesso, modificando sia la componente reale sia quella immaginaria, pur mantenendo invariata la stabilità del sistema. L'effetto complessivo è una variazione della rapidità e del grado di oscillazione della risposta dinamica, pur senza alterare la struttura dominante del sistema, che rimane ben descritta da un modello del primo ordine.

5.3. Risultati sperimentali su motore reale

Nel presente capitolo vengono illustrati i risultati sperimentali ottenuti durante l'attività di laboratorio, finalizzata alla verifica pratica del comportamento dinamico del sistema e alla validazione delle procedure di acquisizione e analisi sviluppate nei capitoli precedenti.

Dopo aver definito il modello teorico del motore, progettato il regolatore e implementato gli strumenti software necessari all'estrazione automatica degli anelli di corrente e dei diagrammi di Bode, risulta infatti fondamentale confrontare tali strumenti con il funzionamento reale del sistema fisico.

L'obiettivo principale delle prove sperimentali non è esclusivamente l'osservazione delle prestazioni del motore elettrico, ma soprattutto la verifica dell'intero flusso di lavoro sviluppato, comprendente la comunicazione tra PC e microcontrollore, l'acquisizione delle grandezze elettriche, la ricostruzione degli anelli di corrente e il successivo post-process dei dati per l'analisi in frequenza. Le misure sperimentali rappresentano quindi il passaggio conclusivo necessario per confermare la coerenza tra modello teorico, simulazioni e comportamento reale del sistema controllato.

Il capitolo è organizzato in due sottocapitoli principali. Nel primo viene descritto il setup sperimentale di laboratorio, illustrando l'architettura hardware utilizzata, le modalità di alimentazione, i dispositivi impiegati e la configurazione di misura adottata durante le prove. Nel secondo sottocapitolo vengono invece presentati e analizzati i risultati sperimentali ottenuti, mostrando le risposte misurate, gli anelli di corrente ricavati e le elaborazioni effettuate tramite le procedure di post-process, con l'obiettivo di valutare la dinamica del sistema e la correttezza delle metodologie sviluppate.

5.3.1. Setup sperimentale

Il setup sperimentale utilizzato per l'esecuzione delle prove in laboratorio è stato progettato con l'obiettivo di garantire condizioni operative controllate, ripetibili e coerenti con le simulazioni sviluppate e con la procedura automatizzata di acquisizione dati. Nelle figure in seguito saranno riportate le principali componenti fisiche del banco prova.

Il sistema è costituito da due alimentatori separati, scelta progettuale adottata per isolare elettricamente la sezione di potenza dalla sezione di controllo digitale, riducendo l'influenza di disturbi elettromagnetici e transitori di corrente sulle misure effettuate dal microcontrollore.

Il primo alimentatore è dedicato all'elettronica di potenza del motore elettrico ed è impostato a 24 V, corrispondenti alla V_{bus} dei ponti H, con limitazione di corrente a 1 A. Tale valore consente di operare in condizioni sicure e controllate durante le prove sperimentali e fornisce l'energia necessaria allo stadio di potenza che genera le correnti di fase del motore stepper. (Figura 66).

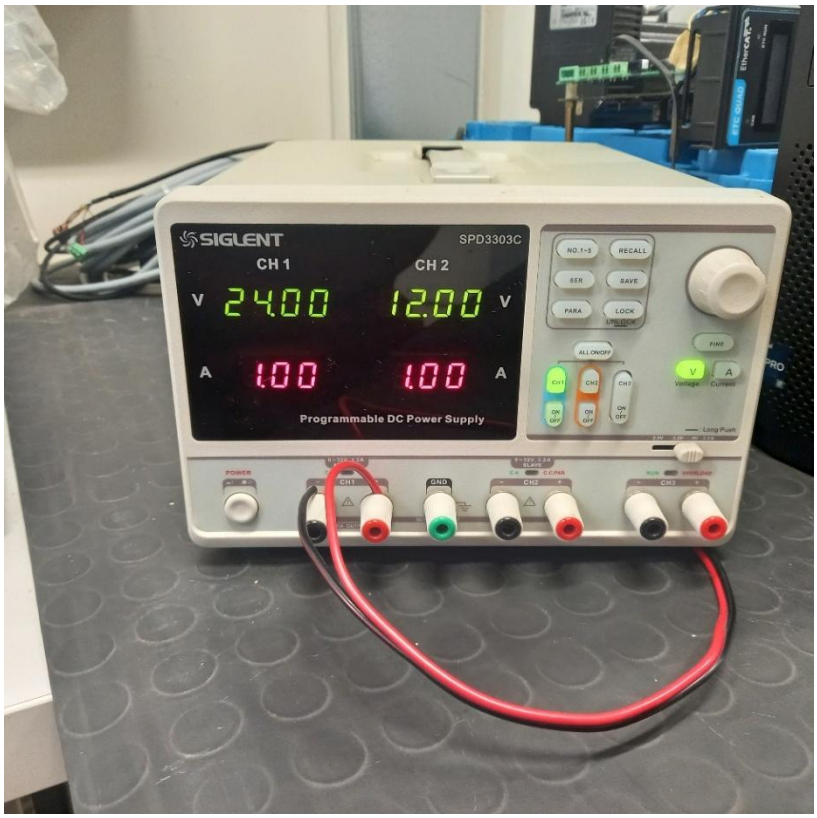

Figura 66-Alimentatore elettronica di potenza

Un secondo alimentatore indipendente viene invece utilizzato per alimentare la scheda di controllo basata su microcontrollore ST (Figura 67). Questo alimentatore è configurato a 12 V e 1 A ed è dedicato esclusivamente alla sezione di controllo digitale, al fine di garantire stabilità di funzionamento e immunità ai disturbi provenienti dallo stadio di potenza.

Figura 67-Alimentatore microcontrollore

La scheda a microcontrollore è fisicamente installata sopra l'elettronica di potenza e rappresenta l'unità centrale di gestione del sistema (Figura 68). Essa esegue gli algoritmi di controllo delle correnti, gestisce la generazione dei riferimenti e supervisiona il funzionamento complessivo del motore. Attraverso i sensori di corrente integrati nello stadio di potenza, il microcontrollore acquisisce le correnti di fase e calcola le componenti nel riferimento sincrono i_d, i_q, necessarie per l'analisi degli anelli di corrente.

La comunicazione tra microcontrollore e computer di supervisione avviene tramite collegamento Ethernet, soluzione che consente uno scambio dati affidabile e ad alta velocità durante l'esecuzione dei test automatici.

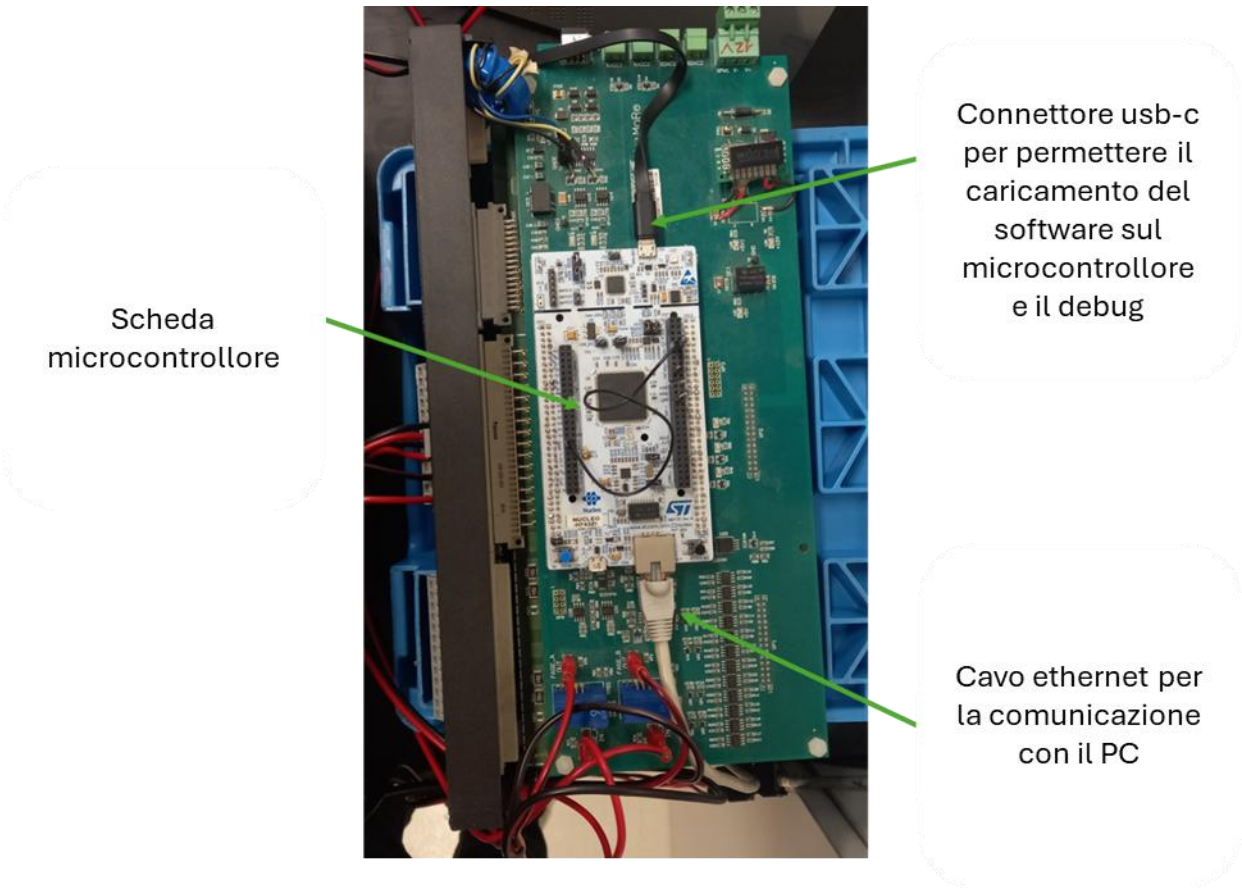

Figura 68-Microcontrollore installato sull'elettronica di potenza

Il motore oggetto delle prove è collegato direttamente allo stadio di potenza tramite le due fasi di avvolgimento (Figura 69). L'elettronica di potenza provvede alla modulazione delle correnti secondo i riferimenti generati dal controllore digitale, permettendo l'eccitazione controllata degli anelli di corrente necessari all'identificazione dinamica del sistema.

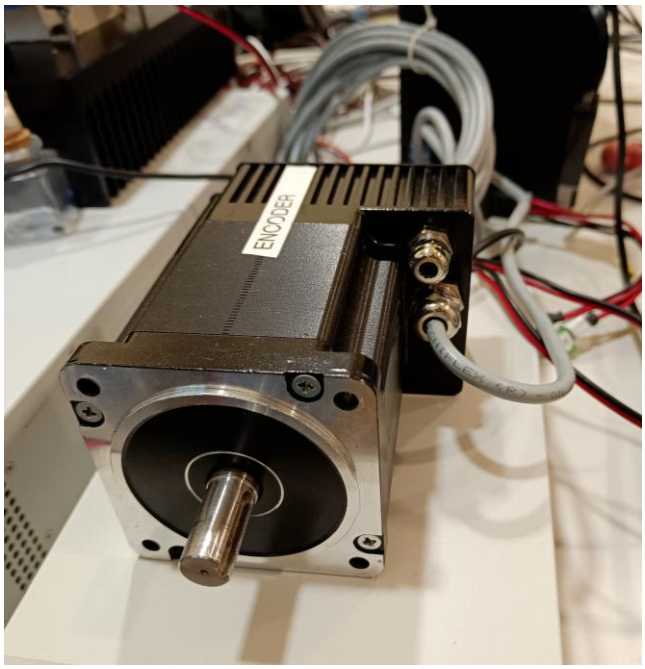

Figura 69-Motore elettrico stepper

Dal punto di vista funzionale, il PC svolge il ruolo di stazione di supervisione degli esperimenti. Per ogni prova viene inviato al microcontrollore un pacchetto contenente i parametri di test, in particolare:

- ampiezza del segnale di eccitazione;
- frequenza di prova;
- coefficienti del regolatore di corrente K_p e K_i.

Una volta ricevuti i parametri, il microcontrollore esegue autonomamente il test applicando il riferimento sinusoidale previsto. Durante il funzionamento vengono trasmesse ciclicamente al PC le correnti misurate i_d e i_q. Sebbene i test vengano eseguiti eccitando un solo anello alla volta (nel presente lavoro esclusivamente l'anello i_d) entrambe le componenti vengono sempre inviate e registrate, consentendo eventuali verifiche incrociate e analisi successive.

I dati ricevuti vengono acquisiti automaticamente tramite programmi Python sviluppati appositamente per:

- la validazione delle acquisizioni sperimentali;
- l'estrazione automatica degli anelli di corrente;
- la successiva generazione dei diagrammi di Bode.

L’intero banco prova risulta quindi composto da tre sottosistemi principali strettamente integrati:

1. sistema di potenza, responsabile dell’alimentazione e dell’attuazione;
2. unità di controllo a microcontrollore, dedicata a misura e regolazione;
3. PC di supervisione, utilizzato per configurazione test ed elaborazione automatica dei dati.

Questa architettura consente l’esecuzione di prove completamente automatizzate, ripetibili e direttamente confrontabili con i risultati di simulazione, riducendo l’intervento manuale dell’operatore e aumentando l’affidabilità dell’analisi sperimentale.

5.3.2. Dati acquisiti

Come già fatto nel capitolo dedicato alle simulazioni con PLECS, anche in questa sezione verranno presentati i risultati relativi agli stessi due casi specifici: $K_i = 100$, $K_p = 0{,}19$ e $K_i = 150$, $K_p = 0{,}54$.

K_i = 100, K_p = 0,19

In questa sezione vengono presentati i risultati sperimentali ottenuti dal banco prova utilizzando i parametri del regolatore pari a $K_i = 100$e $K_p = 0{,}19$. Le misure sono state effettuate applicando al sistema una serie di eccitazioni sinusoidali a frequenze comprese tra 100 Hz e 1000 Hz, mantenendo costanti le condizioni operative descritte nel capitolo precedente, dove sono stati illustrati i risultati ottenuti tramite le simulazioni in ambiente PLECS. I dati acquisiti dal microcontrollore durante il funzionamento del sistema sono stati successivamente elaborati mediante il programma di post-processing sviluppato in Python, che ha permesso di ricostruire i diagrammi di Bode dell’anello di corrente ed estrarre i valori numerici di guadagno e fase per ciascun punto di frequenza analizzato.

Il confronto tra le tre configurazioni di motore è riportato nel diagramma di Bode complessivo mostrato in Figura 70 e Figura 71, nelle quali sono rappresentate simultaneamente le risposte in frequenza dei tre motori analizzati.

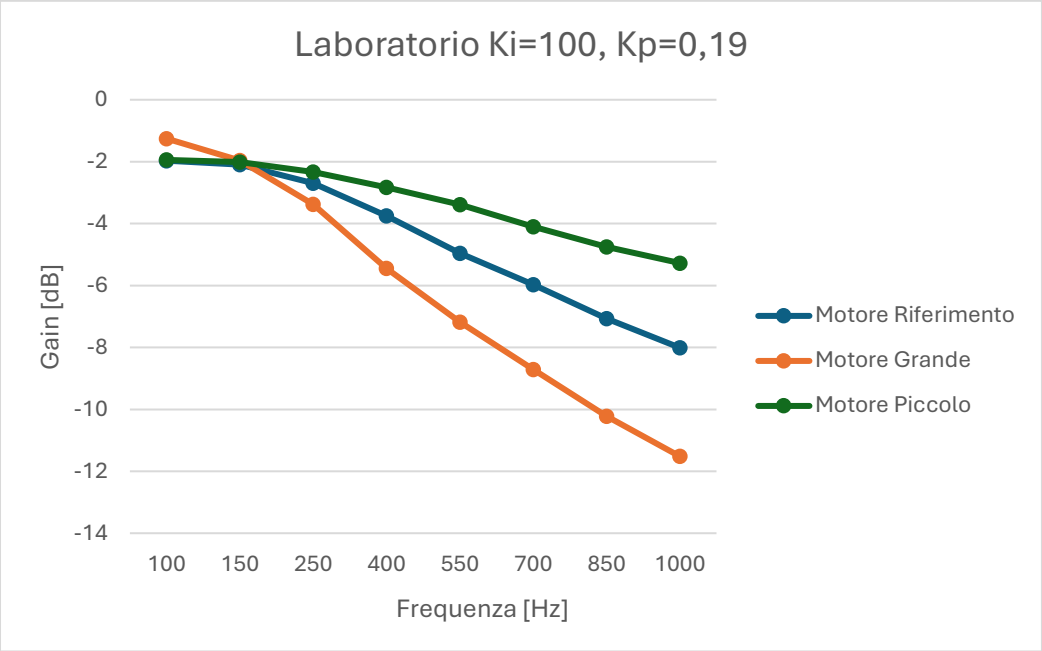

Figura 70-Grafico guadagno del diagramma di Bode (Ki=100, Kp=0.19 Laboratorio)

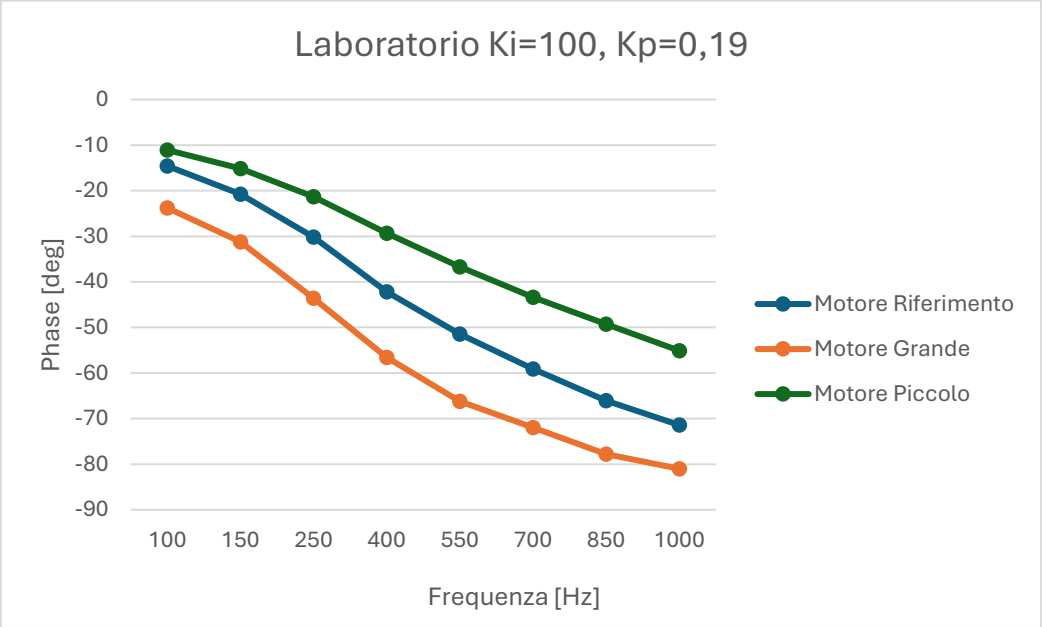

Figura 71-Grafico fase del diagramma di Bode (Ki=100, Kp=0.19 Laboratorio)

I valori numerici corrispondenti di modulo e fase per tutte le frequenze considerate sono invece riportati in forma riassuntiva nella Tabella 6.

MOTORE GRANDE			MOTORE RIFERIMENTO			MOTORE PICCOLO		
f [Hz]	gain_db	phase_deg	f [Hz]	gain_db	phase_deg	f [Hz]	gain_db	phase_deg
100	-1,254952	-23,78031	100	-1,968583	-14,58038	100	-1,943689	-11,06839
150	-1,960875	-31,25962	150	-2,092599	-20,76732	150	-2,014953	-15,10982
250	-3,375994	-43,55076	250	-2,691022	-30,14097	250	-2,332611	-21,32057
400	-5,447666	-56,61145	400	-3,754297	-42,17889	400	-2,828695	-29,30607
550	-7,184939	-66,18187	550	-4,959331	-51,45261	550	-3,38983	-36,70784
700	-8,711758	-71,95498	700	-5,971036	-59,10686	700	-4,105896	-43,3873
850	-10,21936	-77,76892	850	-7,072776	-66,08261	850	-4,751266	-49,28476
1000	-11,51729	-80,99755	1000	-8,008951	-71,4049	1000	-5,274775	-55,09001

Tabella 6-Tabella riassuntiva risultati Ki=100, Kp=0.19 in Laboratorio

Analizzando nel dettaglio le tre curve del diagramma di Bode, si osserva innanzitutto il comportamento del motore di riferimento. Il guadagno dell'anello di corrente assume valori inferiori a 0 dB già alle frequenze più basse considerate. In particolare, a 100 Hz il guadagno risulta pari a circa −1,97 dB, diminuendo progressivamente con l'aumentare della frequenza fino a raggiungere circa −8 dB a 1000 Hz. Questo andamento evidenzia un comportamento tipico di sistema a dinamica del primo ordine, caratterizzato da una progressiva attenuazione della risposta alle frequenze più elevate. Per quanto riguarda la fase, si osserva uno sfasamento che cresce gradualmente con la frequenza, passando da circa −14,6° a 100 Hz fino a circa −71° a 1000 Hz. L'andamento della fase conferma la presenza di una dinamica dominante di tipo induttivo, coerente con il modello elettrico del motore e con il comportamento atteso dell'anello di corrente.

Il motore di taglia piccola mostra invece un'attenuazione del guadagno meno marcata lungo l'intero intervallo di frequenze analizzato. A 100 Hz il guadagno è pari a circa −1,94 dB, valore molto simile a quello del motore di riferimento, ma la diminuzione con la frequenza avviene in modo più graduale. Alla frequenza di 1000 Hz il guadagno si attesta infatti intorno a −5,27 dB, risultando quindi meno attenuato rispetto al caso precedente. Anche l'andamento della fase presenta uno sfasamento inferiore rispetto al motore di riferimento: il valore passa infatti da circa −11° a 100 Hz fino a circa −55° a 1000 Hz.

Questo comportamento suggerisce una dinamica leggermente più rapida dell'anello di corrente, coerente con i parametri elettrici del motore di taglia minore, che generalmente presenta valori di induttanza inferiori e quindi una costante di tempo elettrica più ridotta.

Infine, il motore di taglia grande evidenzia la dinamica più lenta tra i tre casi analizzati. Il guadagno iniziale a 100 Hz risulta pari a circa −1,25 dB, ma decresce rapidamente con l'aumentare della frequenza fino a raggiungere circa −11,5 dB a 1000 Hz. La pendenza più accentuata del diagramma indica una dinamica complessivamente più lenta del sistema. Anche l'andamento della fase conferma questa tendenza: lo sfasamento iniziale è già relativamente elevato (circa −23,8° a 100 Hz) e cresce rapidamente fino a raggiungere circa −81° a 1000 Hz. Questo comportamento è coerente con le caratteristiche elettriche del motore di maggiore taglia, che presenta valori di induttanza più elevati e quindi una costante di tempo elettrica più grande.

Nel complesso, il confronto tra le tre risposte sperimentali evidenzia chiaramente come la taglia del motore influenzi in modo significativo il comportamento dinamico dell'anello di corrente. In particolare, il motore di dimensioni ridotte mostra una risposta più rapida, caratterizzata da una minore attenuazione del guadagno e da uno sfasamento più contenuto lungo l'intero intervallo di frequenze analizzato. Al contrario, il motore di taglia maggiore presenta una dinamica più lenta, con una riduzione del guadagno più pronunciata e un incremento più rapido dello sfasamento. Il motore di riferimento si colloca in una posizione intermedia tra i due casi estremi, confermando la relazione diretta tra i parametri elettrici del motore, in particolare induttanza e resistenza di fase, e le prestazioni dinamiche dell'anello di corrente.

K_i = 150, K_p = 0,54

Nel presente paragrafo vengono analizzati i risultati sperimentali ottenuti sul banco di prova considerando la configurazione del regolatore con parametri $K_i = 150$e $K_p = 0{,}54$. Come nel caso precedente, l'obiettivo dell'analisi è osservare l'andamento della risposta in frequenza dell'anello di corrente $I_d$per le tre diverse taglie di motore stepper utilizzate durante le prove, valutando in particolare l'attenuazione del guadagno e l'andamento dello sfasamento al variare della frequenza di eccitazione.

Il confronto tra le tre configurazioni di motore è riportato nel diagramma di Bode complessivo mostrato in Figura 72 e Figura 73, nelle quali sono rappresentate simultaneamente le risposte in frequenza dei tre motori analizzati.

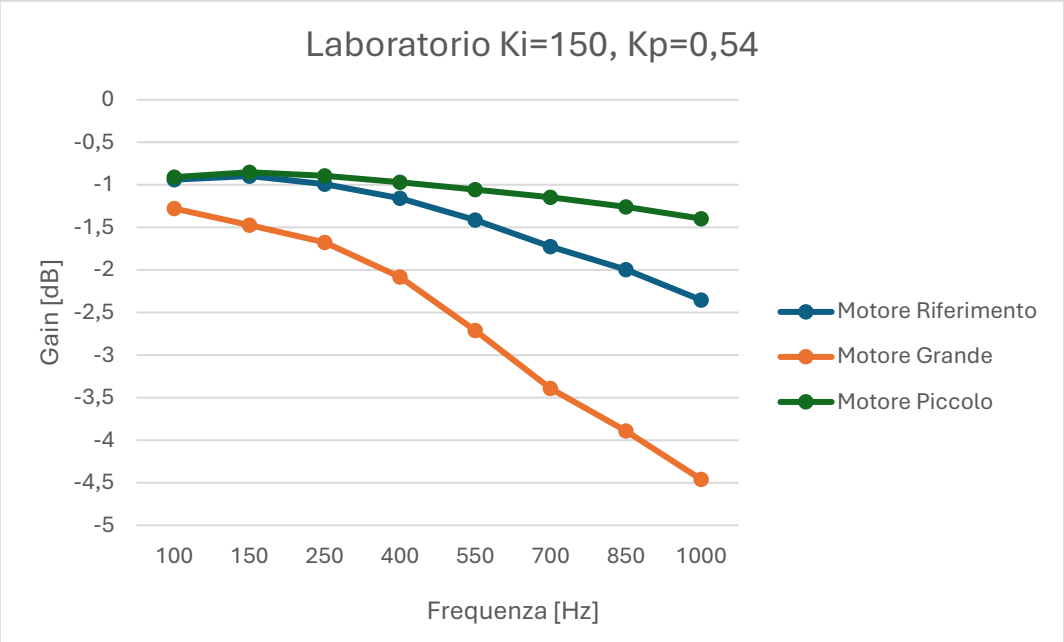

Figura 72-Grafico guadagno del diagramma di Bode (Ki=150, Kp=0.54 Laboratorio)

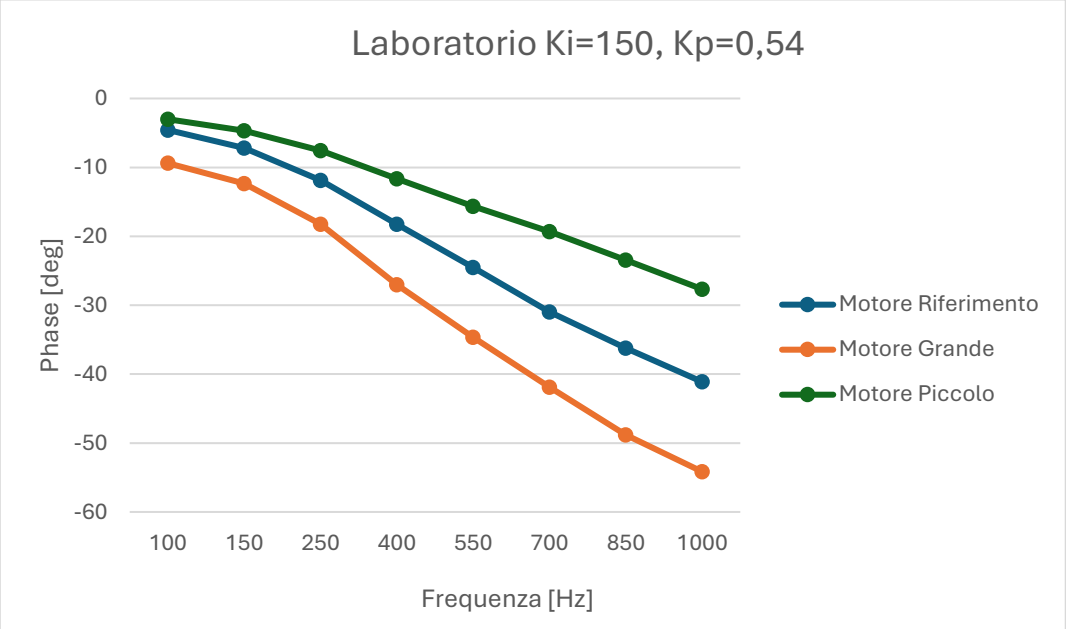

Figura 73-Grafico fase del diagramma di Bode (Ki=150, Kp=0.54 Laboratorio)

I valori numerici di guadagno e fase ottenuti dalle misure sperimentali alle diverse frequenze sono invece riportati in forma riassuntiva nella Tabella 7.

MOTORE GRANDE			MOTORE RIFERIMENTO			MOTORE PICCOLO		
f [Hz]	gain_db	phase_deg	f [Hz]	gain_db	phase_deg	f [Hz]	gain_db	phase_deg
100	-1,278246	-9,388117	100	-0,938322	-4,566133	100	-0,907262	-3,006903
150	-1,473061	-12,34058	150	-0,896277	-7,202207	150	-0,851838	-4,657004
250	-1,675595	-18,23254	250	-0,992586	-11,85635	250	-0,89564	-7,54716
400	-2,083755	-27,02532	400	-1,157881	-18,26277	400	-0,968048	-11,63951
550	-2,71281	-34,64112	550	-1,412124	-24,50462	550	-1,054978	-15,6445
700	-3,393782	-41,90668	700	-1,723833	-30,95554	700	-1,146149	-19,32794
850	-3,892455	-48,81652	850	-1,996859	-36,21435	850	-1,259	-23,43398
1000	-4,461517	-54,15272	1000	-2,352942	-41,11375	1000	-1,396691	-27,66982

Tabella 7-Tabella riassuntiva risultati Ki=150, Kp=0.54 in Laboratorio

Analizzando le tre curve del diagramma di Bode si osserva innanzitutto il comportamento del motore di riferimento. Il guadagno dell'anello di corrente assume valori prossimi a −1 dB già alle frequenze più basse analizzate, con un valore pari a circa −0,94 dB a 100 Hz. All'aumentare della frequenza si osserva una progressiva attenuazione della risposta, che raggiunge circa −2,35 dB a 1000 Hz. L'andamento risulta quindi relativamente regolare e monotono, caratterizzato da una riduzione graduale del guadagno all'aumentare della frequenza. Anche lo sfasamento segue un andamento tipico di un sistema a dinamica di primo ordine: il valore passa da circa −4,6° a 100 Hz fino a circa −41° a 1000 Hz, evidenziando una crescita progressiva dello sfasamento con la frequenza. Questo comportamento è coerente con la dinamica dell'anello di corrente, nella quale l'induttanza dell'avvolgimento introduce un ritardo crescente alle frequenze più elevate.

Il motore di taglia piccola mostra un comportamento qualitativamente simile ma caratterizzato da una dinamica leggermente più rapida. Il guadagno rimane infatti leggermente più elevato rispetto al caso precedente alle basse frequenze, con un valore di circa −0,91 dB a 100 Hz, e diminuisce gradualmente fino a circa −1,40 dB a 1000 Hz. L'attenuazione complessiva risulta quindi più contenuta rispetto al motore di riferimento, suggerendo una risposta dinamica leggermente più veloce. Anche lo sfasamento cresce con la frequenza ma con valori inferiori rispetto al motore precedente: si passa infatti da circa −3° a 100 Hz fino a circa −27,7° a 1000 Hz.

Questo comportamento è coerente con il fatto che il motore di dimensioni più ridotte presenta generalmente valori di induttanza inferiori, che comportano una costante di tempo elettrica più piccola e quindi una dinamica più rapida dell'anello di corrente.

Infine viene analizzato il comportamento del motore di taglia grande, che evidenzia una risposta più attenuata e uno sfasamento più pronunciato rispetto agli altri due motori. Il guadagno parte infatti da circa −1,28 dB a 100 Hz e decresce progressivamente fino a circa −4,46 dB a 1000 Hz, mostrando una riduzione più marcata rispetto ai casi precedenti. Anche l'andamento della fase evidenzia una dinamica più lenta: lo sfasamento passa da circa −9,4° a 100 Hz fino a oltre −54° a 1000 Hz. Questo comportamento è coerente con le caratteristiche elettriche di un motore di dimensioni maggiori, che tende a presentare induttanze più elevate e quindi costanti di tempo elettriche maggiori. Di conseguenza, l'anello di corrente risulta più lento e mostra una maggiore attenuazione alle frequenze più alte.

Confrontando complessivamente i tre motori emerge come la taglia della macchina influenzi in modo significativo la dinamica dell'anello di corrente. Il motore di dimensioni più ridotte mostra infatti una risposta più rapida, con minore attenuazione del guadagno e valori di sfasamento più contenuti sull'intero intervallo di frequenze analizzato. Il motore di riferimento presenta un comportamento intermedio, mentre il motore di taglia maggiore evidenzia una dinamica più lenta, caratterizzata da una maggiore attenuazione del guadagno e da uno sfasamento più pronunciato alle alte frequenze. Nonostante tali differenze, l'andamento generale delle risposte risulta coerente tra i tre casi, confermando la correttezza della procedura di misura e la capacità del sistema di acquisizione di rappresentare in modo affidabile la dinamica dell'anello di corrente anche nel caso di prove sperimentali su motori reali.

Funzioni di trasferimento

Analogamente a quanto effettuato per il modello PLECS analizzato in precedenza, anche per i motori reali testati sul banco prova è stata eseguita una fase di analisi post–process dei dati sperimentali acquisiti sugli anelli di corrente i_d. A partire dai diagrammi di Bode ottenuti sperimentalmente sono state identificate le funzioni di trasferimento in anello chiuso $G(s)$ e sono stati successivamente calcolati i poli delle funzioni risultanti.

L'analisi dell'andamento dei diagrammi di Bode relativi ai motori reali ha mostrato un comportamento dinamico che, analogamente al caso precedentemente analizzato, può essere ricondotto a una dinamica dominante di primo ordine.

Per questo motivo, anche per i dati sperimentali ottenuti dal banco prova è stato adottato un modello con un singolo polo dominante, ritenuto sufficiente a descrivere il comportamento principale dell'anello di corrente nei diversi casi di studio.

Per verificare la coerenza del comportamento tra i diversi attuatori analizzati, è stata confrontata la funzione di trasferimento in anello chiuso nel caso $K_i = 150$ e $K_p = 0.54$ per i tre motori considerati (motore piccolo Figura 76, Figura 77, motore di riferimento Figura 74, Figura 75 e motore grande Figura 78, Figura 79). Come già fatto nella sezione precedente, per motivi di ordine e chiarezza espositiva viene riportato unicamente questo caso rappresentativo. È comunque opportuno sottolineare che un comportamento qualitativamente analogo è stato osservato anche per tutte le altre combinazioni di parametri considerate.

Nei grafici riportati nelle figure sottostanti sono stati sovrapposti i diagrammi di Bode ottenuti sperimentalmente e quelli ricavati a partire dalle funzioni di trasferimento $G(s)$ identificate. Anche in questo caso il confronto consente di valutare l'accuratezza del modello adottato nel descrivere la dinamica osservata. Si può osservare come l'andamento della fase risulti sostanzialmente coincidente tra modello e dati sperimentali, mostrando una sovrapposizione praticamente perfetta. Per quanto riguarda invece il modulo del guadagno, i due andamenti risultano molto simili, pur non essendo completamente sovrapponibili, evidenziando alcune lievi differenze tra modello e risposta reale del sistema.

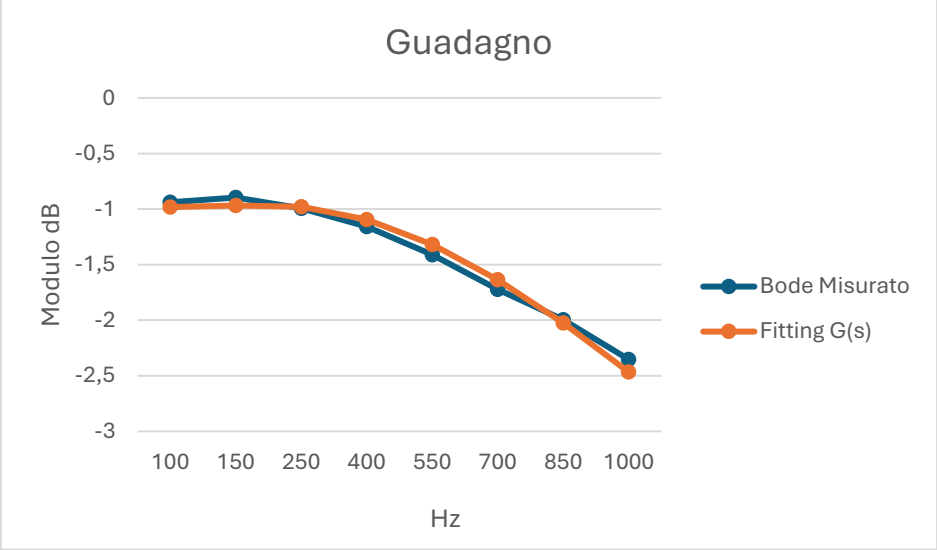

Figura 74-Confronto guadagno per Ki =150 e Kp =0.54 Motore Riferimento

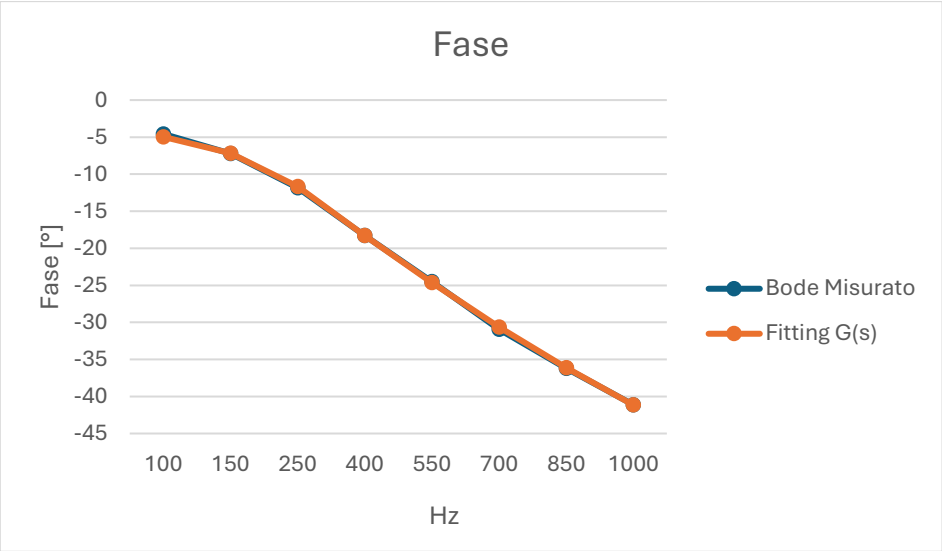

Figura 75-Confronto fase per Ki =150 e Kp =0.54 Motore Riferimento

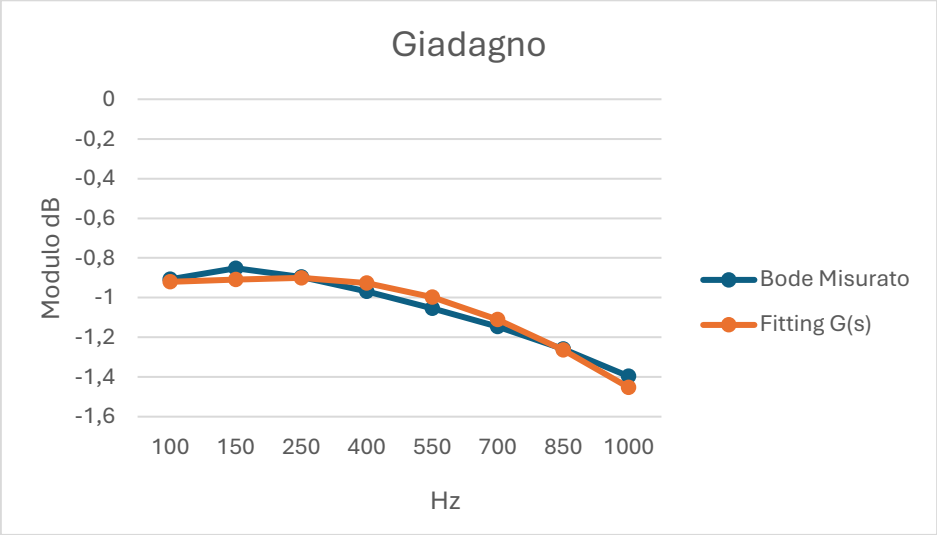

Figura 76-Confronto guadagno G(s) per Ki =150 e Kp =0.54 Motore Piccolo

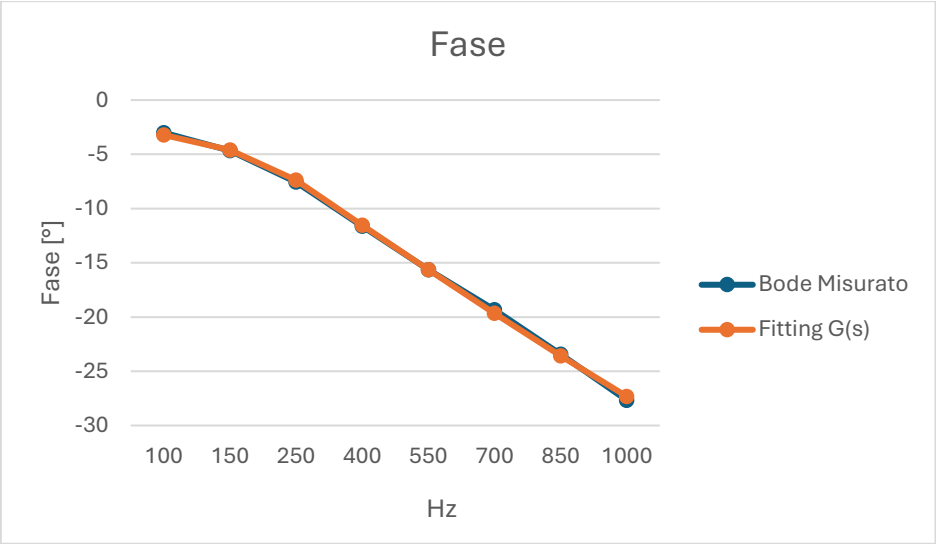

Figura 77-Confronto fase G(s) per Ki =150 e Kp =0.54 Motore Piccolo

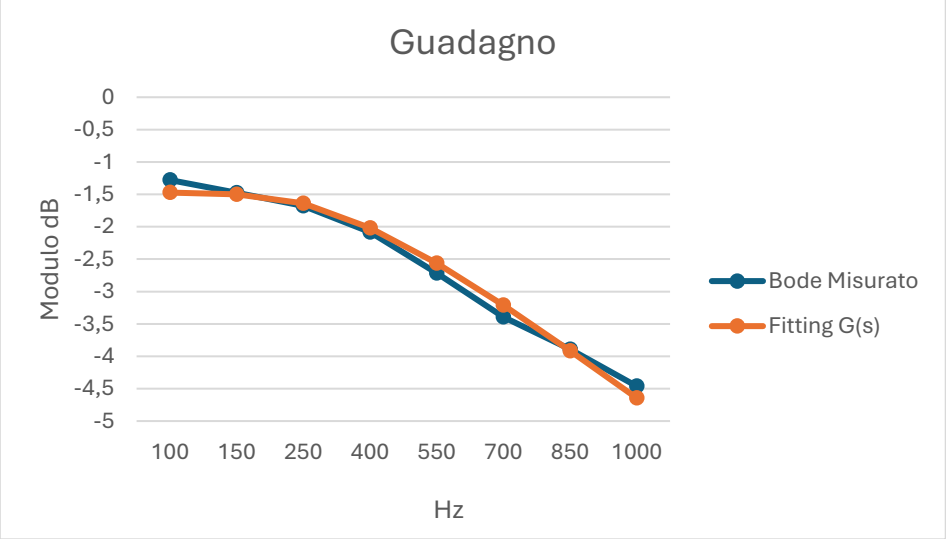

Figura 78-Confronto guadagno per Ki =150 e Kp =0.54 Motore Grande

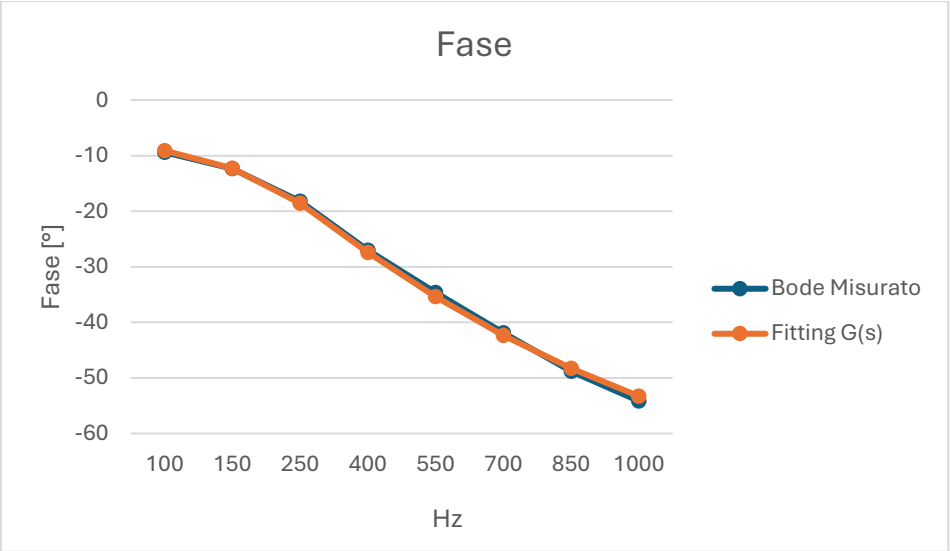

Figura 79-Confronto fase per Ki =150 e Kp =0.54 Motore Grande

Il confronto evidenzia come l'andamento delle funzioni di trasferimento risulti molto simile per i tre motori, a conferma del fatto che la struttura dinamica dell'anello di corrente rimane sostanzialmente invariata al variare della taglia del motore.

Le differenze osservabili tra i diversi casi sono principalmente riconducibili a variazioni di guadagno e di fase, già evidenziate nell'analisi dei diagrammi di Bode, e sono legate alle diverse caratteristiche elettriche e costruttive dei motori.

Successivamente è stata analizzata la posizione dei poli della funzione di trasferimento, i cui valori sono riportati nella Tabella 8. Ai fini della stabilità del sistema risulta particolarmente rilevante la parte reale dei poli, la quale deve risultare negativa affinché il sistema sia stabile.

POLI	1 POLO					
KI/KP	MOTORE PICCOLO		MOTORE RIFERIMENTO		MOTORE GRANDE	
	Re	Im	Re	Im	Re	Im
Ki=100 Kp=0,14	-4188,434	-104,2558	-2424,629	167,45429	-1384,099	111,55051
Ki=100 Kp=0,19	-5606,891	120,74656	-3410,081	302,57723	-1993,447	56,220685
Ki=100 Kp=0,21	-6011,82	467,29675	-3545,145	1052,4913	-2325,602	96,512192
Ki=100 Kp=0,27	-7456,282	767,29094	-4587,912	471,52434	-3183,103	84,085624
Ki=100 Kp=0,30	-8041,705	900,15972	-4611,821	1271,1975	-3515,527	244,39532
Ki=100 Kp=0,38	-9728,518	1300,123	-6125,627	910,32576	-4295,857	481,7045
Ki=100 Kp=0,54	-12694,59	1737,7811	-8165,172	1294,3502	-5682,119	662,426
Ki=150 Kp=0,19	-5105,155	950,30979	-3028,048	796,20028	-2039,592	-9,333072
Ki=150 Kp=0,27	-7201,432	611,21237	-4226,416	1015,4318	-2998,935	6,9394761
Ki=150 Kp=0,38	-9560,407	1051,5762	-5596,689	1337,1671	-4144,491	108,2123
Ki=150 Kp=0,54	-12965,27	1507,059	-8043,437	1116,004	-5624,7	443,44142

Tabella 8-Valori dei poli

Dall'analisi dei risultati ottenuti emerge che tutti i poli presentano parte reale negativa per l'intero intervallo di parametri analizzati, indicando che l'anello di corrente risulta stabile per tutte le configurazioni considerate.

Inoltre, si osserva che i poli presentano generalmente una componente immaginaria non nulla, indicativa della presenza di una dinamica leggermente oscillatoria, sebbene la componente reale negativa garantisca comunque un comportamento stabile e smorzato.

Per evidenziare l'andamento della parte reale dei poli al variare del guadagno proporzionale K_p, sono stati realizzati i grafici riportati nella Figura 80 e Figura 81, relativi rispettivamente ai casi $K_i = 100$ e $K_i = 150$.

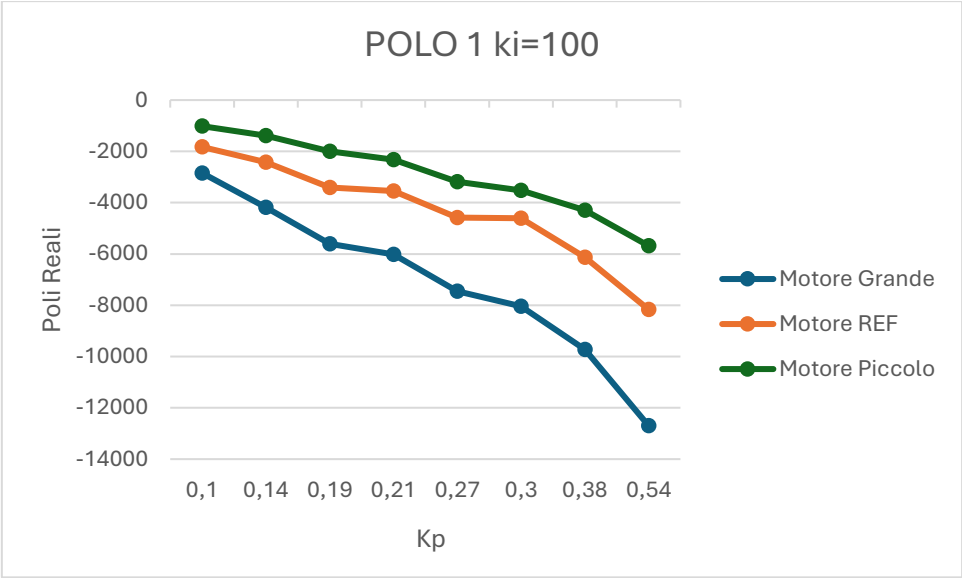

Figura 80-Andamento della parte reale dei poli per Ki = 100 motori reali

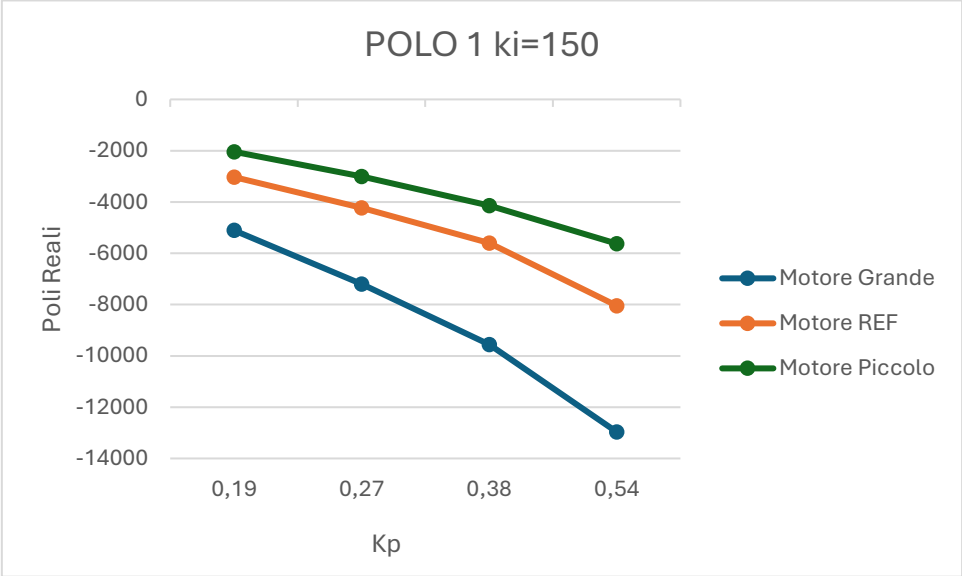

Figura 81-Andamento della parte reale dei poli per Ki = 150 motori reali

Dall'osservazione dei grafici è possibile individuare un comportamento comune ai tre motori: all'aumentare del guadagno proporzionale K_p la parte reale dei poli tende a spostarsi verso valori progressivamente più negativi, indicando una dinamica dell'anello di corrente più rapida. Questo comportamento risulta coerente con l'effetto dell'incremento dell'azione proporzionale nel regolatore PI, che comporta un aumento della banda passante del sistema di controllo.

Inoltre, analogamente a quanto osservato nelle analisi PLECS precedenti, il motore di riferimento presenta generalmente valori intermedi della parte reale dei poli rispetto al motore piccolo e al motore grande. In particolare, il motore piccolo tende a presentare poli con modulo maggiore, associati a dinamiche più rapide, mentre il motore grande mostra valori più vicini all'origine, indicativi di una risposta leggermente più lenta.

Infine, il confronto tra i risultati ottenuti per $K_i = 100$ e $K_i = 150$ evidenzia come l'aumento del guadagno integrale influenzi la posizione dei poli nel piano complesso, modificando la dinamica del sistema pur mantenendo invariata la stabilità complessiva dell'anello di corrente. Anche nel caso dei motori reali, quindi, il comportamento osservato risulta coerente con quanto previsto teoricamente e con quanto emerso dall'analisi in frequenza dei diagrammi di Bode.

5.4. Confronto tra risultati teorici, simulati e sperimentali

In questo sottocapitolo vengono confrontati i risultati ottenuti tramite simulazione nel software PLECS con quelli derivanti dalle prove sperimentali effettuate al banco prova. L'analisi è stata condotta considerando sempre tre differenti taglie di motore: un motore di riferimento, un motore di taglia maggiore e un motore di taglia minore.

L'obiettivo di questo confronto è valutare il grado di corrispondenza tra il modello simulativo e il comportamento reale del sistema, analizzando in particolare le differenze in termini di guadagno e fase della risposta in frequenza. Le prove sono state effettuate su un intervallo di frequenze compreso tra 100 Hz e 1000 Hz, confrontando per ciascun punto i risultati ottenuti tramite simulazione con quelli misurati sperimentalmente al banco prova.

Per ogni configurazione dei parametri del regolatore PI è stata quindi calcolata la differenza tra i valori simulati e quelli sperimentali sia per il guadagno, espresso in dB, sia per la fase, espressa in gradi. Successivamente tali differenze sono state sintetizzate attraverso valori medi, al fine di ottenere un indicatore complessivo dello scostamento tra il modello simulativo e il comportamento reale del sistema.

Nel seguito vengono analizzati in maniera più dettagliata i due casi rappresentativi, già mostrati in precedenza, del comportamento del sistema, corrispondenti alle configurazioni del regolatore $K_i = 100, K_p = 0.19$ e $K_i = 150, K_p = 0.54$. Queste due configurazioni sono state scelte in quanto rappresentano rispettivamente una situazione in cui le differenze tra simulazione e prova sperimentale risultano più evidenti e una condizione in cui il modello simulativo mostra invece un'elevata corrispondenza con il sistema reale.

Analisi del caso $K_i = 100, K_p = 0.19$

Nel primo caso analizzato il regolatore PI è caratterizzato da un guadagno proporzionale relativamente contenuto e da un termine integrativo pari a 100. Tale configurazione è stata analizzata per le tre taglie di motore considerate, confrontando i risultati ottenuti tramite simulazione con quelli derivanti dalle prove sperimentali.

I risultati completi del confronto tra simulazione e dati sperimentali sono riportati nelle tabelle riportate di seguito (Tabella 9, Tabella 10, Tabella 11), nelle quali vengono mostrati, per ciascuna frequenza analizzata, i valori di guadagno e fase ottenuti in simulazione e quelli misurati sperimentalmente, insieme alla relativa differenza.

MOTORE GRANDE							
PLECS			Sperimentale			DIFF.	
f [Hz]	gain_db	phase_deg	f [Hz]	gain_db	phase_deg	gain_db	phase_deg
100	-7,86197	-35,6564	100	-1,25495	-23,7803	6,60702	11,87607
150	-9,09625	-38,7227	150	-1,96088	-31,2596	7,135378	7,46306
250	-9,50111	-43,5884	250	-3,37599	-43,5508	6,125116	0,037641
400	-9,93176	-50,6919	400	-5,44767	-56,6115	4,484092	5,9196
550	-9,97503	-57,6966	550	-7,18494	-66,1819	2,79009	8,485227
700	-10,544	-63,7275	700	-8,71176	-71,955	1,832263	8,227529
850	-11,0954	-69,9306	850	-10,2194	-77,7689	0,876019	7,838281
1000	-11,4204	-77,5071	1000	-11,5173	-80,9975	0,096904	3,490495
						3,74336	6,667238

Tabella 9-Confronto motore grande ki=100, kp=0.19

MOTORE RIFERIMENTO							
PLECS			Sperimentale			DIFF.	
f [Hz]	gain_db	phase_deg	f [Hz]	gain_db	phase_deg	gain_db	phase_deg
100	-8,09222	-31,9918	100	-1,96858	-14,5804	6,123636	17,41147
150	-10,4051	-48,2269	150	-2,0926	-20,7673	8,312526	27,45957
250	-10,7776	-46,8645	250	-2,69102	-30,141	8,086624	16,72349
400	-11,2177	-48,5689	400	-3,7543	-42,1789	7,463355	6,39003
550	-11,5667	-49,4282	550	-4,95933	-51,4526	6,607359	2,024448
700	-11,6594	-52,4193	700	-5,97104	-59,1069	5,688342	6,687568
850	-11,6017	-57,9051	850	-7,07278	-66,0826	4,528894	8,1775
1000	-11,684	-60,7236	1000	-8,00895	-71,4049	3,675035	10,68129
						6,310721	11,94442

Tabella 10-Confronto motore riferimento ki=100, kp=0.19

MOTORE PICCOLO							
PLECS			Sperimentale			DIFF.	
f [Hz]	gain_db	phase_deg	f [Hz]	gain_db	phase_deg	gain_db	phase_deg
100	-10,0137	-42,8584	100	-1,94369	-11,0684	8,070015	31,78998
150	-10,6434	-41,6901	150	-2,01495	-15,1098	8,628475	26,58025
250	-11,29	-35,6784	250	-2,33261	-21,3206	8,957347	14,35778
400	-11,5999	-30,3689	400	-2,8287	-29,3061	8,77125	1,062809
550	-11,6453	-34,6398	550	-3,38983	-36,7078	8,255507	2,067997
700	-11,6677	-39,8567	700	-4,1059	-43,3873	7,561828	3,530561
850	-11,6856	-42,9298	850	-4,75127	-49,2848	6,934314	6,35497
1000	-11,6521	-46,0268	1000	-5,27477	-55,09	6,377287	9,06316
						7,944503	11,85094

Tabella 11-Confronto motore piccolo ki=100, kp=0.19

Dall'analisi dei dati riportati in tabella emerge come l'andamento della risposta in frequenza ottenuta sperimentalmente segua in maniera qualitativamente simile quello previsto dal modello simulativo. Tuttavia, si osserva la presenza di uno scostamento sistematico tra i valori simulati e quelli misurati, che si manifesta sia nel guadagno sia nella fase della risposta.

In particolare, analizzando le differenze medie calcolate sull'intero intervallo di frequenze considerato, si osserva che il motore di taglia maggiore presenta lo scostamento più contenuto tra simulazione e prova sperimentale, mentre il motore di taglia minore mostra le differenze più elevate. Il motore di riferimento si colloca in una posizione intermedia tra questi due casi.

Per una migliore interpretazione dei risultati, nelle figure seguenti sono riportati alcuni grafici in cui viene plottata, per una specifica taglia di motore e per ogni frequenza analizzata, la differenza di guadagno e di fase tra il caso sperimentale e quello simulato in PLECS.

Dal confronto dei diagrammi di guadagno si osserva come le curve ottenute sperimentalmente presentino un andamento molto simile a quello previsto dal modello simulativo, pur mostrando un offset che risulta più marcato nel caso del motore di taglia minore (Figura 82).

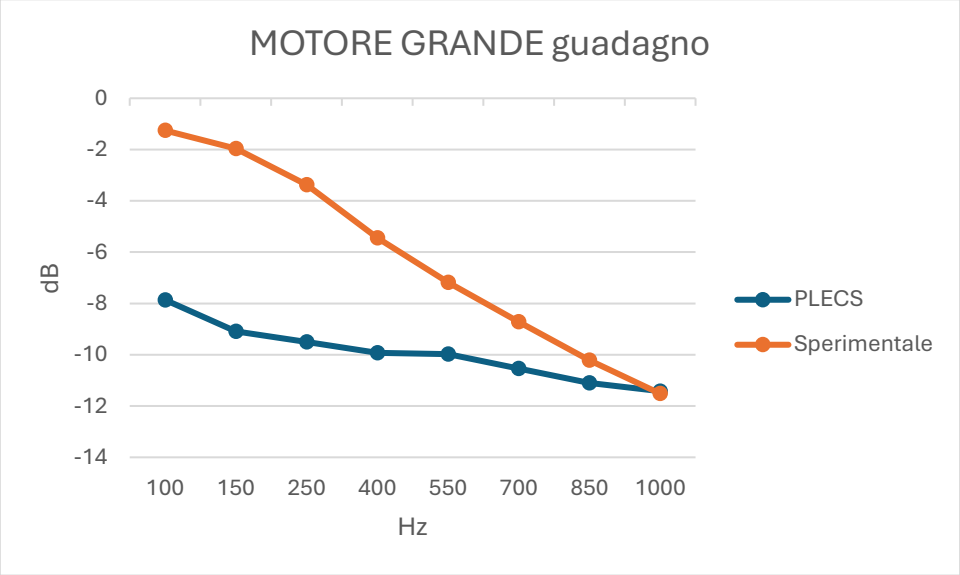

Figura 82-Differenza di guadagno del motore di taglia grande (ki=100, kp=0.19)

Analogamente, anche nel diagramma di fase si osserva una buona corrispondenza nella forma complessiva delle curve, mentre lo scostamento tra simulazione e dati sperimentali tende ad aumentare al crescere della frequenza. Questo comportamento può essere attribuito alla presenza di effetti dinamici non completamente modellati nella simulazione, quali ritardi introdotti dall'elettronica di controllo, fenomeni dissipativi e non linearità presenti nel sistema reale (Figura 83).

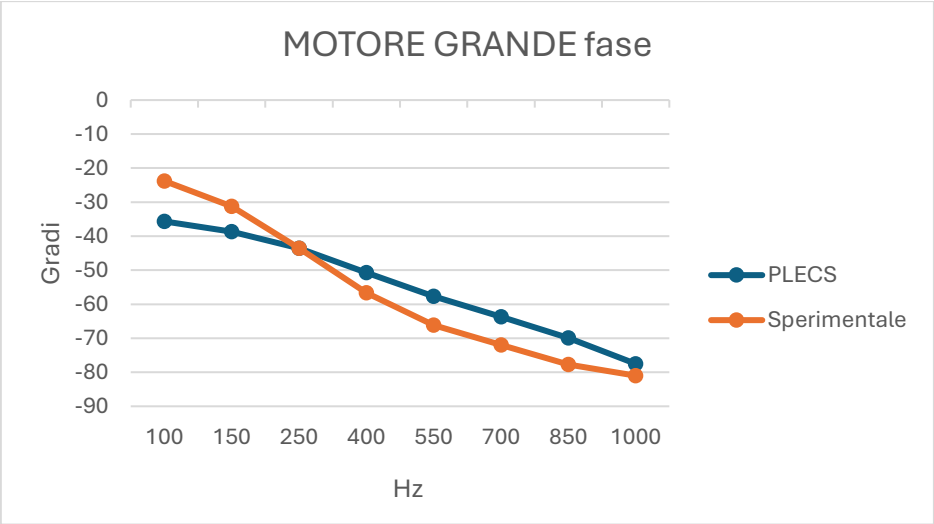

Figura 83-Differenza di fase del motore di taglia grande (ki=100, kp=0.19)

Analisi del caso $K_i = 150, K_p = 0.54$

Il secondo caso analizzato è caratterizzato da valori più elevati dei parametri del regolatore PI, con un guadagno proporzionale pari a 0.54 e un termine integrativo pari a 150. Questa configurazione è stata scelta in quanto rappresenta una condizione in cui il sistema mostra una dinamica più reattiva e consente di valutare come varia la corrispondenza tra simulazione e risultati sperimentali al variare dei parametri di controllo.

Anche in questo caso il confronto dettagliato tra i risultati simulati e quelli misurati sperimentalmente è riportato nelle tabelle seguenti: Tabella 12, Tabella 13, Tabella 14.

MOTORE GRANDE							
PLECS			Sperimentale			DIFF.	
f [Hz]	gain_db	phase_deg	f [Hz]	gain_db	phase_deg	gain_db	phase_deg
100	-2,91531	-12,6667	100	-1,27825	-9,38812	1,63706	3,27856
150	-3,2378	-14,172	150	-1,47306	-12,3406	1,764737	1,83146
250	-3,5194	-18,3642	250	-1,67559	-18,2325	1,843805	0,13167
400	-3,8128	-25,2098	400	-2,08375	-27,0253	1,729044	1,815526
550	-4,13091	-31,2498	550	-2,71281	-34,6411	1,418098	3,391301
700	-4,48676	-37,1705	700	-3,39378	-41,9067	1,092975	4,736185
850	-4,87638	-42,4489	850	-3,89246	-48,8165	0,983926	6,36766
1000	-5,26798	-47,649	1000	-4,46152	-54,1527	0,80646	6,503751
						1,409513	3,507014

Tabella 12-Confronto motore grande ki=150, kp=0.54

MOTORE RIFERIMENTO							
PLECS			Sperimentale			DIFF.	
f [Hz]	gain_db	phase_deg	f [Hz]	gain_db	phase_deg	gain_db	phase_deg
100	-2,96578	-11,3136	100	-0,93832	-4,56613	2,027458	6,747443
150	-3,25945	-11,746	150	-0,89628	-7,20221	2,363171	4,543802
250	-3,48152	-13,5427	250	-0,99259	-11,8563	2,488931	1,686344
400	-3,62602	-19,184	400	-1,15788	-18,2628	2,468139	0,921253
550	-3,76147	-24,5039	550	-1,41212	-24,5046	2,349342	0,000681
700	-3,81751	-28,4978	700	-1,72383	-30,9555	2,093676	2,457768
850	-4,42921	-35,082	850	-1,99686	-36,2143	2,432348	1,132373
1000	-4,61658	-38,9411	1000	-2,35294	-41,1137	2,263637	2,172599
						2,310838	2,457783

Tabella 13-Confronto motore riferimento ki=150, kp=0.54

MOTORE PICCOLO							
PLECS			Sperimentale			DIFF.	
f [Hz]	gain_db	phase_deg	f [Hz]	gain_db	phase_deg	gain_db	phase_deg
100	-3,12248	-12,8842	100	-0,90726	-3,0069	2,215214	9,87732
150	-3,41492	-12,4549	150	-0,85184	-4,657	2,563079	7,797849
250	-3,60342	-12,8982	250	-0,89564	-7,54716	2,707775	5,351035
400	-3,66944	-15,4078	400	-0,96805	-11,6395	2,701393	3,768257
550	-3,71973	-18,4879	550	-1,05498	-15,6445	2,664754	2,843445
700	-3,75418	-21,7954	700	-1,14615	-19,3279	2,608027	2,467467
850	-3,79548	-24,953	850	-1,259	-23,434	2,536476	1,519057
1000	-3,7949	-28,6626	1000	-1,39669	-27,6698	2,398207	0,992813
						2,549366	4,327155

Tabella 14-Confronto motore piccolo ki=150, kp=0.54

Dall'analisi dei dati riportati nelle tabelle si osserva come le differenze tra simulazione e risultati sperimentali risultino sensibilmente ridotte rispetto al caso precedente. In particolare, sia per il guadagno sia per la fase, gli scostamenti medi risultano significativamente inferiori per tutte le taglie di motore considerate.

Per visualizzare in modo più immediato il grado di corrispondenza tra simulazione numerica e prova sperimentale, nelle figure seguenti sono riportati i grafici della differenza di guadagno e di fase, per ciascuna frequenza analizzata, tra i risultati ottenuti sperimentalmente e quelli derivanti dalla simulazione con PLECS per questa configurazione del regolatore.

Dal grafico del guadagno si osserva come le curve ottenute tramite simulazione e quelle sperimentali risultino molto vicine tra loro sull'intero intervallo di frequenze analizzato, evidenziando una buona accuratezza del modello simulativo (Figura 84).

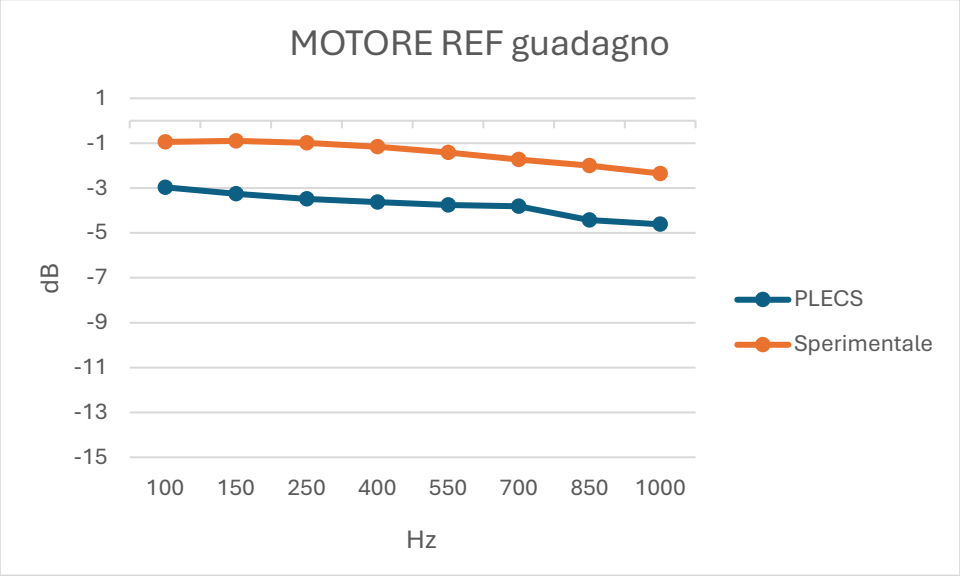

Figura 84-Differenza di guadagno del motore di riferimento (ki=150, kp=0.54)

Anche nel diagramma di fase si osserva una buona sovrapposizione tra le curve, con differenze generalmente contenute entro pochi gradi. Questo risultato indica che, per questa configurazione dei parametri del regolatore, il modello implementato in PLECS è in grado di rappresentare con buona accuratezza il comportamento dinamico del sistema reale (Figura 85).

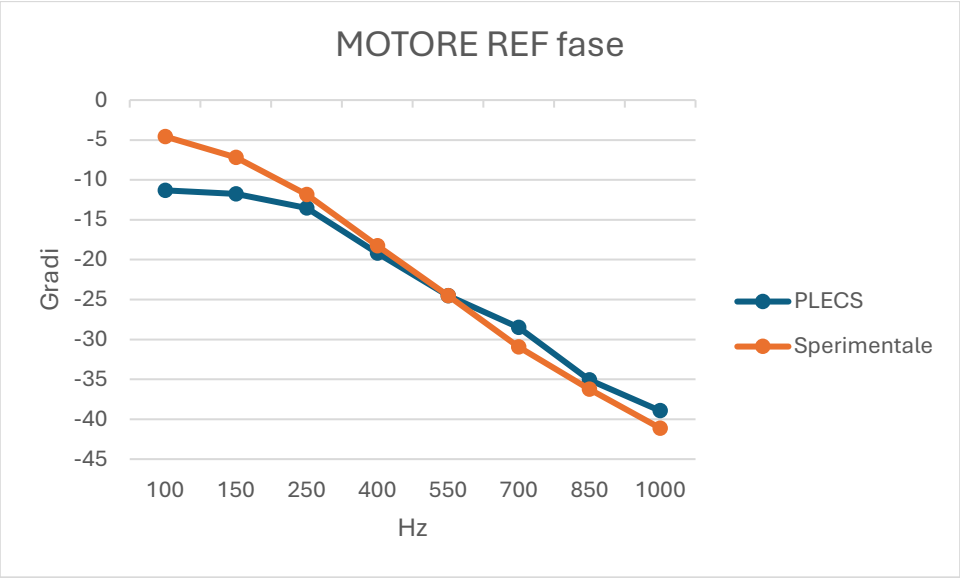

Figura 85-Differenza di fase del motore di riferimento (ki=150, kp=0.54)

Confronto complessivo tra tutte le configurazioni analizzate

Al fine di ottenere una visione complessiva delle prestazioni del modello simulativo, è stata infine realizzata una tabella riepilogativa contenente le medie delle differenze tra simulazioni e prove sperimentali per tutte le configurazioni dei parametri del regolatore PI analizzate (Tabella 15).

	MEDIE FINALI					
	MOTORE GRANDE		MOTORE RIFERIMENTO		MOTORE PICCOLO	
	gain_db	phase_deg	gain_db	phase_deg	gain_db	phase_deg
Ki=100 Kp=0.14	3,232	6,122	5,084	11,504	6,719	13,245
Ki=100 Kp=0.19	3,743	6,667	6,311	11,944	7,945	11,851
Ki=100 Kp=0.21	3,973	7,546	6,121	14,673	8,108	11,852
Ki=100 Kp=0.27	3,502	6,157	6,427	8,662	7,395	7,741
Ki=100 Kp=0.30	3,441	5,762	5,675	9,543	6,399	7,024
Ki=100 Kp=0.38	2,673	4,43	4,08	3,258	4,468	5,6
Ki=100 Kp=0.54	1,542	3,442	2,281	2,301	2,603	4,037
Ki=150 Kp=0.19	3,349	6,278	6,141	17,304	7,736	15,555
Ki=150 Kp=0.27	3,429	5,962	6,122	11,711	6,425	14,007
Ki=150 Kp=0.38	2,351	6,669	3,969	7,732	4,16	7,522
Ki=150 Kp=0.54	1,41	3,507	2,311	2,478	2,549	4,327

Tabella 15-Tabella riassuntiva delle medie finali

La tabella riporta, per ciascuna combinazione dei parametri K_p e K_i, i valori medi dello scostamento tra simulazione e risultati sperimentali relativamente sia al guadagno, espresso in dB, sia alla fase, espressa in gradi, considerando separatamente le tre taglie di motore analizzate.

Per analizzare in maniera più approfondita il comportamento del modello simulativo, sono stati inoltre realizzati quattro ulteriori grafici che mostrano l'andamento delle differenze tra simulazione e prove sperimentali al variare dei parametri del regolatore; due per il caso con $K_i = 100$, $K_i = 150$. In particolare, le prime due figure (Figura 86, Figura 87) riportano l'andamento della differenza relativa al guadagno, mentre le due successive evidenziano l'andamento della differenza relativa alla fase.

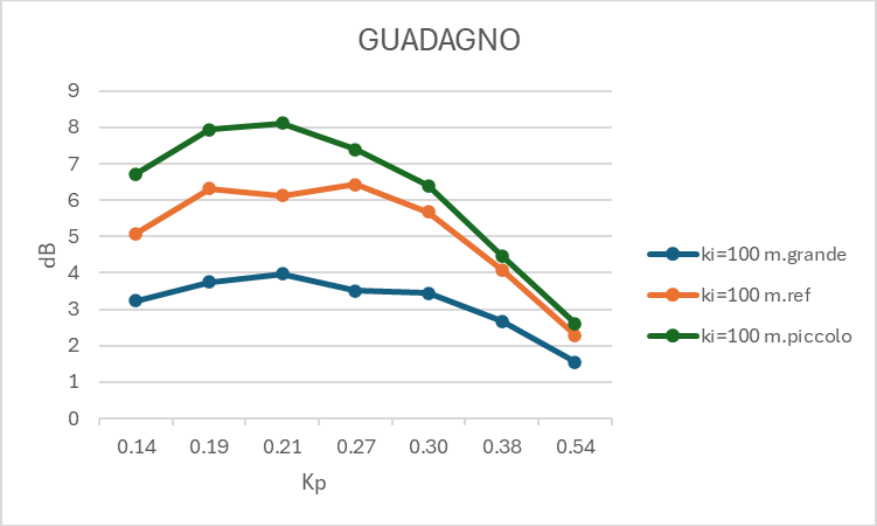

Figura 86-Andamento della differenza di guadagno (ki=100)

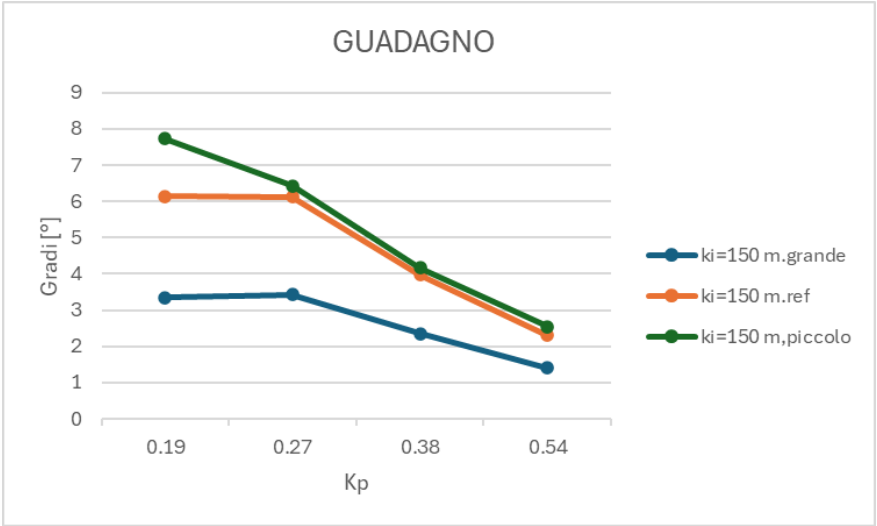

Figura 87-Andamento della differenza di guadagno (ki=150)

Dall'analisi dell'andamento delle differenze di guadagno si osserva come, all'aumentare del guadagno proporzionale K_p, lo scostamento tra simulazione e risultati sperimentali tenda generalmente a ridursi.

Questo comportamento evidenzia come le configurazioni caratterizzate da valori più elevati di K_p consentano al modello simulativo di riprodurre con maggiore accuratezza il comportamento reale del sistema.

Un'analisi analoga può essere effettuata osservando le differenze relative alla fase (Figura 88, Figura 89). Anche in questo caso si nota una tendenza generale alla riduzione dello scostamento tra simulazione e dati sperimentali per configurazioni del regolatore caratterizzate da valori più elevati del guadagno proporzionale. Tuttavia, l'andamento delle differenze di fase risulta generalmente più variabile rispetto a quello osservato per il guadagno, evidenziando una maggiore sensibilità della dinamica del sistema reale a fenomeni non completamente modellati nella simulazione.

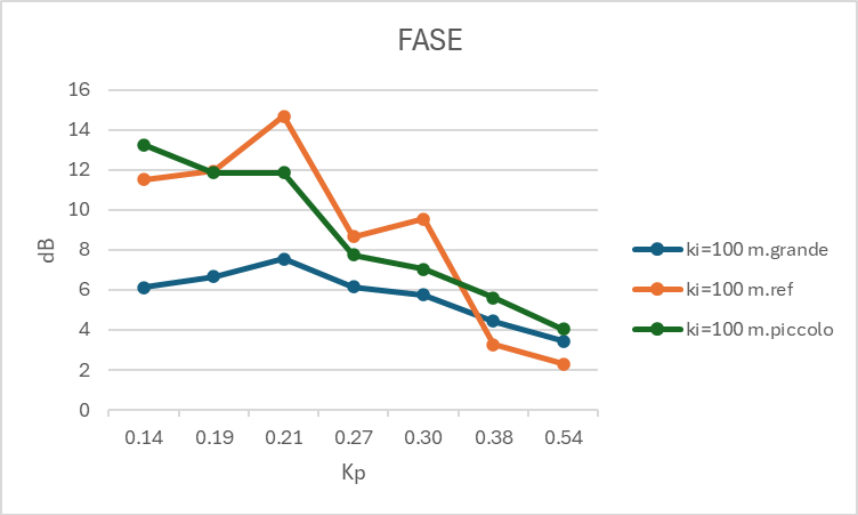

Figura 88-Andamento della differenza di fase (ki=100)

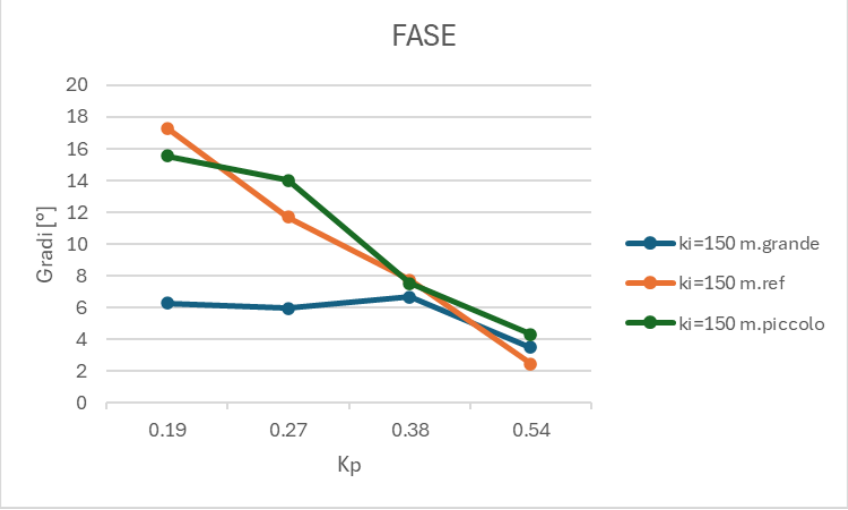

Figura 89-Andamento della differenza di fase (ki=150)

Un ulteriore aspetto che emerge dall'analisi complessiva dei risultati riguarda l'influenza della taglia del motore sul grado di corrispondenza tra simulazione e dati sperimentali. In generale si osserva come il motore di taglia minore presenti differenze più elevate tra i risultati simulati e quelli misurati al banco prova, mentre il motore di taglia maggiore mostri una corrispondenza più accurata con il modello simulativo. Questo comportamento può essere attribuito al fatto che nei motori di dimensioni ridotte alcuni fenomeni non modellati, quali perdite parassite, non linearità magnetiche e tolleranze costruttive, possono avere un impatto relativamente più significativo sul comportamento dinamico del sistema.

Nel complesso, nonostante le inevitabili differenze tra il modello simulativo e il sistema reale, i risultati mostrano come il modello implementato in PLECS sia in grado di rappresentare in maniera soddisfacente la dinamica del sistema. In particolare, il modello riproduce correttamente l'andamento della risposta in frequenza e mantiene gli scostamenti entro valori generalmente contenuti sia per il guadagno sia per la fase, soprattutto nelle configurazioni del regolatore caratterizzate da valori più elevati del guadagno proporzionale.

6. Conclusioni

Nel presente lavoro di tesi è stata affrontata l'analisi e la caratterizzazione dinamica degli anelli di corrente nei sistemi di controllo per motori stepper, con l'obiettivo principale di sviluppare strumenti software in grado di automatizzare le procedure di identificazione e di analisi della risposta in frequenza del sistema. Gli obiettivi prefissati possono essere considerati complessivamente raggiunti, sia dal punto di vista dello sviluppo degli strumenti software sia dal punto di vista della loro applicazione e validazione su sistemi reali.

Uno dei risultati principali del lavoro è rappresentato dalla realizzazione di due software sviluppati in linguaggio Python, progettati per automatizzare l'estrazione e l'elaborazione degli anelli di corrente e dei corrispondenti diagrammi di Bode. In particolare, per la fase di comunicazione con il sistema e per l'estrazione dei risultati sono stati sviluppati due programmi distinti: uno dedicato all'interfacciamento con l'ambiente di simulazione PLECS e uno progettato per operare sul banco di prova reale tramite comunicazione con il microcontrollore che gestisce il controllo del motore. Nonostante la necessità di mantenere separati i due programmi a causa delle differenti modalità di comunicazione con i sistemi coinvolti, la loro struttura è stata progettata in modo il più possibile uniforme, riutilizzando un numero elevato di funzioni comuni. Questo approccio ha permesso di mantenere coerenza nelle procedure di generazione dei test, nella gestione dei parametri di prova e nell'elaborazione dei dati acquisiti, semplificando al contempo la manutenzione del codice e garantendo che le stesse metodologie di analisi potessero essere applicate sia ai risultati delle simulazioni sia a quelli ottenuti dalle prove sperimentali sul sistema reale.

Il secondo software sviluppato è invece dedicato alla fase di post–process dei dati acquisiti. A partire dai valori di guadagno e fase ottenuti durante le prove, il programma ricostruisce automaticamente la risposta in frequenza del sistema e consente di calcolare le principali grandezze utili all'analisi del controllo. In particolare, il software permette di determinare le funzioni di trasferimento del sistema sia in anello chiuso sia in anello aperto, di ricavare la funzione $G(s)$ equivalente del sistema e di identificare poli e zeri del modello dinamico. Inoltre, il programma genera automaticamente le rappresentazioni grafiche e i diagrammi necessari per l'analisi delle prestazioni, rendendo l'intero processo di elaborazione dei dati rapido, ripetibile e facilmente applicabile a differenti configurazioni di prova.

Un ulteriore obiettivo del lavoro è stato la validazione dei software sviluppati attraverso la loro applicazione a casi di studio reali. A tal fine sono state analizzate tre diverse taglie di motori stepper, caratterizzate da differenti parametri elettrici e dinamici. Per ciascun motore sono state eseguite prove sia in ambiente simulato, utilizzando modelli implementati in PLECS, sia su un banco di prova reale controllato tramite microcontrollore. Questa doppia modalità di analisi ha consentito non solo di verificare il corretto funzionamento degli strumenti sviluppati, ma anche di confrontare il comportamento teorico e simulato del sistema con quello effettivamente osservato durante il funzionamento reale dei motori.

I risultati ottenuti mostrano una buona coerenza tra le risposte in frequenza ricavate dalle simulazioni e quelle misurate sperimentalmente. Gli andamenti dei diagrammi di Bode e le caratteristiche dinamiche degli anelli di corrente risultano infatti qualitativamente simili nei due casi, confermando la validità dei modelli utilizzati e delle procedure di analisi implementate. Le differenze osservate tra simulazione e prove reali risultano generalmente contenute e sono principalmente attribuibili alle inevitabili non idealità presenti nei sistemi fisici, quali tolleranze dei componenti, effetti termici, rumore di misura e fenomeni non lineari difficilmente modellabili in modo completo negli ambienti di simulazione.

Il lavoro svolto si inserisce in un contesto più ampio di studio e miglioramento delle tecniche di controllo dei motori elettrici stepper, con particolare attenzione alla caratterizzazione degli anelli di regolazione interni. L'analisi condotta in questa tesi si è concentrata sugli anelli di corrente, che rappresentano il livello più interno e veloce della struttura di controllo dell'azionamento. In prospettiva futura, il lavoro potrebbe essere esteso allo studio degli anelli di velocità, che costituiscono il livello superiore della catena di controllo. L'integrazione dell'analisi degli anelli di corrente con quella degli anelli di velocità permetterebbe infatti di ottenere una caratterizzazione ancora più completa dell'azionamento e di fornire ulteriori strumenti per l'ottimizzazione delle strategie di controllo e per il miglioramento delle prestazioni complessive dei motori stepper.

In conclusione, il lavoro ha permesso di sviluppare e validare una metodologia automatizzata per l'analisi della risposta in frequenza degli anelli di corrente nei sistemi di controllo per motori stepper, dimostrando come l'integrazione tra simulazione, sperimentazione e strumenti software dedicati possa costituire un approccio efficace per la caratterizzazione e il miglioramento delle prestazioni dei sistemi di azionamento elettrico.

Bibliografia

[1] PLECS XML-RPC Interface and Controller Design in Python

[2] PLECS - THE SIMULATION PLATFORM FOR POWER ELECTRONIC SYSTEMS

[3] M. L. Meade. “Lock-in amplifiers: principles and applications”. Peter peregrinus Ltd, 1983

[4] M. L. Meade. “Advances in lock-in amplifiers”. In: Journal of Physics E: Scientific Instruments 15 (1982)

[5] E. Lorenzani, Lezioni di Elettrotecnica e Macchine Elettriche

[6] F. Immovilli, Lezioni di Conversione Statica dell’Energia

[7] L. Maini, “Modellizzazione e controllo di un vettoriale di un motore stepper ibrido in presenza di non linearità”, Reggio Emilia, Italia, 2020.

[8] Matthes, E. (2019). “Python Crash Course: A Hands-On, Project-Based Introduction to Programming” (2nd ed.). No Starch Press.

[9] Ramalho, L. (2022). “Fluent Python” (2nd ed.). O’Reilly Media.

[10] Python Software Foundation. (2024). Python Documentation. https://docs.python.org/

[11] Z. Yansuo, L. Yonggang, L. Wenqi, L. Yu, L. Qingmian, and W. Di, “Research on a high-speed and heavy-duty closed-loop drive system of a two-phase hybrid stepping motor based on a hybrid controller”, Wireless Communications and Mobile Computing, vol. 2021, no. 1, p. 2515820, 2021.

[12] N. Mohan and S. Raju, “Power electronics, a first course: simulations and laboratory implementations”, John Wiley & Sons, 2022.

[13] S. Kumar and S. Rai, “Survey on transport layer protocols: Tcp & udp”, International Journal of Computer Applications, vol. 46, no. 7, pp. 20‑25, 2012.

[14] F. T. AL-Dhief, N. Sabri, N. A. Latiff, N. Malik, M. Abbas, A. Albader, M. A. Mohammed, R. N. AL-Haddad, Y. D. Salman, M. Khanapi et al., "Performance comparison between tcp and udp protocols in different simulation scenarios", International Journal of Engineering & Technology, vol. 7, no. 4.36, pp. 172‑176, 2018.

[15] A. Toscani, "Digistep: azionamento digitale intelligente ad alta dinamica per motori passo passo", Parma, 2002/2003.

[16] F. Cerri, G. Ortolani e E. Venturi , "Hardware dei microprocessori e microcontrollori", in *Corso di sistemi automatici. Per le articolazioni ELETTROTECNICA, ELETTRONICA e AUTOMAZIONE degli Istituti Tecnici settore Tecnologico*, Firenze-Milano, Urlico Hoepli Milano, volume 2, 2013.

[17] B. R. Shahidur, "Comunicazione PC‑Inverter tramite protocollo LAN per l'implementazione in Python di un sistema di autotuning dei regolatori PI", Reggio Emilia, Italia, 2025.

[18] A. Marzo, "Realizzazione di un amplificatore lock-in digitale", Bologna, Italia, 2015.

[19] Karl J. Astrom, "Tore Hdgglund Advanced PID Control Department of Automatic Control", Lund Institute of Technology, Lund Uniuersity

Ringraziamenti

Desidero ringraziare innanzitutto la mia famiglia, che mi ha sostenuto con affetto, fiducia e costante incoraggiamento lungo tutto il mio percorso universitario, rappresentando sempre un punto di riferimento fondamentale nei momenti di difficoltà e di crescita.

Un sincero ringraziamento va anche ai miei amici e a tutte le persone che ho avuto il piacere di incontrare durante questi anni all'università: ciascuno, in modi diversi, ha contribuito a rendere questo percorso non solo formativo dal punto di vista accademico, ma anche significativo sotto il profilo umano.

Ringrazio il professor Fabio Immovilli per avermi dato l'opportunità di svolgere questo lavoro di tesi e per la fiducia accordatami. Un particolare ringraziamento va inoltre al professor Emilio Carfagna, che mi ha seguito durante lo sviluppo della tesi con grande disponibilità e professionalità, offrendo indicazioni preziose e un costante supporto nel chiarire dubbi e approfondire gli aspetti del lavoro.

A tutti loro va la mia più sincera gratitudine per aver contribuito, in modi diversi, al raggiungimento di questo importante traguardo.

"Per aspera ad astra".