POLITECNICO DI MILANO

SCUOLA DI INGEGNERIA INDUSTRIALE E DELL'INFORMAZIONE

Corso di Laurea Specialistica in Ingegneria dell'Automazione

Identificazione tramite reti neurali artificiali NARX di un manipolatore planare a tre bracci per il controllo attivo di vibrazioni

Relatore: Prof. Ferruccio Resta

Correlatore: Ing. Francesco Ripamonti

Tesi di laurea specialistica di:

Isabella Boiocchi

Matricola 783520

Anno Accademico 2012-2013

Sommario

Nelle strutture flessibili, a causa del moto libero o di disturbi esterni, possono instaurarsi vibrazioni, un fenomeno dannoso che può compromettere il corretto funzionamento di un qualsiasi sistema meccanico quali macchine utensili, manipolatori industriali. Il controllo attivo di vibrazioni permette di limitare questo fenomeno; in particolare i controlli model based raggiungono l'obiettivo attraverso la stima dei parametri fondamentali che caratterizzano la struttura. Se il sistema ha un comportamento non lineare, tuttavia l'approccio classico, basato sulla teoria dei sistemi lineari, non risulta particolarmente efficace.

Nel presente lavoro di tesi è stato studiato presso i laboratori del Politecnico di Milano un banco prova di un manipolatore a tre gradi di libertà. Il banco prova riproduce il comportamento dinamico di sistemi industriali quali i bracci per la distribuzione di calcestruzzo. Tali sistemi sono per natura flessibili e soggetti ad affaticamento strutturale causato dalla presenza di vibrazioni. L'obiettivo principale della tesi è quello di identificare un modello non lineare del sistema al fine di favorire lo sviluppo di opportune logiche di controllo adattativo. Le simulazioni numeriche sono state eseguite sfruttando la cosimulazione del modello multibody del manipolatore esportato in Simulink. Il modello è ottenuto addestrando una rete neurale artificiale, strumento molto potente in caso di sistemi affetti da non linearità. È stato quindi implementato un controllo di

vibrazioni LQR modale ottenuto dall'aggiustamento della legge di controllo in funzione della configurazione in cui si trova il sistema.

La prima parte dell'elaborato definisce il problema da un punto di vista teorico per fornire al lettore le basi sull'identificazione non lineare e sullo stato dell'arte delle reti neurali, compreso il loro funzionamento. Una seconda parte mostra l'applicazione del metodo a un caso di studio più semplice: un singolo braccio robotico. La terza parte estende lo sviluppo al sistema a tre gradi di libertà, introducendo ulteriori problematiche non visibili nel caso di braccio singolo.

Abstract

Mechanical vibrations could compromise correct functioning of every mechanical system, as tools machines and industrial robots. This dramatic phenomenon is due to free movement or external disturbances. Vibration active control could limit this occurrence; indeed model based controls reach goal throw valuation of fundamentals parameters characterizing the structure. The classical approach, based on linear system theory, doesn't provide good results for non-linear systems.

In this work a test bench of a three degree of freedom set in Politecnico di Milano is been studied. Test bench reproduces dynamic behavior of industrial systems, as robot arms for concrete distribution. These systems are flexible by nature and subject to structural weariness due to vibration occurrence. The main goal of thesis is non-linear model identification of system, developing specific adaptive control strategies. Numerical simulations are run thanks to co-simulation of manipulator multi-body model, exported in Simulink. An artificial neural network, a real powerful tool in non-linear field, is trained to identify the model. A modal LQR vibration control is been implemented, achieved by control law update, according to actual system configuration.

First part of thesis details problem from a theoretical point of view, to provide reader all non-linear identification basis and neural network state of art, functioning included. The second part shows application of chosen method to a simple case: single robotic arm. The third part extends method to system provided with three degrees of freedom, introducing set of problems, not usual in the single robot arm case.

Ringraziamenti

Questa sezione mi permette di ringraziare il mio relatore e il mio correlatore per la collaborazione ottenuta, ma soprattutto per avermi fatto conoscere parti del mio carattere di cui non sapevo l'esistenza. Occorre che ringrazi anche i miei compagni di tesi, che mi hanno prestato aiuto in momenti di difficoltà.

È impossibile non ricordare tutte le persone che hanno creduto in me e che continuano a farlo, che mi danno consigli e mi supportano. Tra queste persone ci sono chiaramente tutti i componenti della mia famiglia, colgo quindi questa occasione per ringraziarli per tutto quello che hanno dovuto sopportare, senza di loro infatti non sarei mai stata capace di affrontare tutto questo. Ringrazio i miei genitori, che fin da piccola hanno cercato di prepararmi ad affrontare un mondo non sempre roseo, dandomi esempio di umiltà e spronandomi al miglioramento continuo.

Ringrazio i miei nonni, che hanno accresciuto in me la voglia di renderli fieri, in qualsiasi modo possibile. Ringrazio la mia sorellona, l'altra metà di me con cui voglio condividere questa gioia. Ringrazio anche tutte le persone a cui tengo e che non ci sono più, perché fin da piccola mi hanno insegnato molte cose.

Ringrazio in particolare Alessandro, a cui mi rivolgo sempre per avere consiglio e che mi offre sempre spunti nuovi. Spero me ne possa dare anche in futuro per molto tempo, fino a quando vorrà...

Devo ringraziare anche gli amici, supporto fidato e di consolazione, che mi hanno saputo ascoltare in tutti i momenti che hanno accompagnato questo lavoro, e che mi offrono sempre la possibilità di vedere le cose da una prospettiva migliore. Devo aggiungere anche i miei colleghi, che mi hanno accolta nel migliore dei modi e con i quali spero di avere una collaborazione proficua anche in futuro.

Mi sento in obbligo di ringraziare anche tutte le donne che non mi hanno mai conosciuta, ma che con la loro ostinazione e la loro caparbietà hanno permesso che arrivassi a concludere la mia istruzione. Senza di loro infatti non avrei mai potuto aspirare a tanto, e come me non avrebbero potuto farlo anche tutte le altre che puntualmente si dimenticano della fortuna e del privilegio di cui godono.

Ultimi, ma non mancano, i ringraziamenti per chi ha fondato l'università che mi ha offerto la formazione che mi resterà per tutta la vita, un insegnamento che non trapela dai libri né dalle lezioni, che mi accompagnerà per sempre.

Questo che sto conseguendo è un traguardo importante, e voglio pensare che non stabilisca un punto finale ma, semmai, l'inizio di una grande avventura.

Indice

Indice delle figure

1. Introduzione

Particolarmente rilevante nelle grandi strutture, il problema legato alla riduzione delle vibrazioni strutturali è tuttora un caso di studio aperto e stimolante in molti sistemi meccanici. Il controllo di vibrazioni applicato a strutture flessibili infatti è una problematica rilevante che tocca molti settori disparati. Quando i fenomeni vibratori raggiungono elevate ampiezze di oscillazione, essi possono essere nocivi sia alla struttura stessa che all'uomo che si trova ad operare nelle vicinanze. La struttura risulta infatti sollecitata eccessivamente e in modo progressivo, provocando cricche e dissesti collegati direttamente a malfunzionamenti e a una riduzione della sua vita utile, se non addirittura a cedimenti di componenti. È chiaro come una situazione di questo tipo sia altamente pericolosa per l'uomo, costretto a trovarsi in un ambiente di lavoro caratterizzato da poca sicurezza. Strettamente legato alle vibrazioni, un discorso a parte va fatto per il rumore causato dal fenomeno vibratorio a cui è soggetta la struttura in alta frequenza.

Al fine di migliorare queste situazioni sfavorevoli, sono state studiate molte tecniche tra le quali si annoverano sia metodi di controllo passivo che di controllo attivo.

Tra le prime si possono citare quelle tecniche che prevedono il raggiungimento dell'obiettivo tramite alcuni accorgimenti in fase di progetto. Si consideri ad esempio l'equilibramento dei motori a combustione interna a fronte di forze centrifughe indesiderate. Spostandosi al settore civile, per i grattacieli sono spesso adottate tecniche di *controllo risonante* (TMD), che hanno come obiettivo l'aggiunta sul sistema di una massa risonante sulla frequenza propria più critica del grattacielo. A seconda del problema e degli obiettivi prefissati, esistono varianti in cui si impiegano diversi TMD in serie [1].

Tra gli studi attinenti al controllo attivo di vibrazioni, si distinguono tecniche *strutturate* e *non strutturate*. Tra le tecniche strutturate (note anche come *model based*) si possono citare i controllori ottimi e modali. Questi ultimi sfruttano la decomposizione modale per creare un modello ridotto del sistema. Nell'ipotesi di ridurre l'ampiezza delle oscillazioni tramite incremento dello smorzamento, la legge di controllo è funzione delle velocità modali $\dot{q}$ e può essere quindi espressa come:

$$u = [A_c]\,\dot{q} \qquad (1.1)$$

dove $[A_c]$ è una opportuna matrice di controllo. Nel caso di sistemi da controllare con dinamiche tempo varianti molto lente, esse vengono trascurate nella modellizzazione e si parla quindi di sistemi *lineari tempo invarianti* (LTI). Per questi sistemi il calcolo della matrice di controllo può essere semplicemente effettuato offline. Nel caso di controllori modali, a prescindere dalla natura del modello, l'anello di controllo è quindi composto anche da un osservatore di stato che calcola online i valori dello stato modale, che moltiplicano la matrice $[A_c]$.

Esistono anche casi in cui risulta utile esprimere il modello come un sistema *lineare tempo variante* (LTV) [2], proprio quando non si possono trascurare le dinamiche temporali. In questi casi si impiega un'azione di controllo adattativa, performante anche in presenza di incertezze su alcuni parametri del sistema. Si

consideri ad esempio una struttura flessibile da controllare, un manipolatore, in cui è necessario mostrare l'effettiva dipendenza dalla configurazione che esso assume, anch'essa dipendente da un'altra variabile: il tempo.

Per sistemi *non lineari o tempo varianti*, occorrerebbe quindi ripetere l'identificazione più volte, in particolare tante volte quante sono le configurazioni di lavoro di interesse. L'identificazione inoltre presenta problematiche più complesse a causa della non linearità, e ancor più se il sistema presenta più variabili [3], [4]. È evidente come questo approccio comporti un onere computazionale notevole, aspetto fortemente negativo per qualsiasi tipo di applicazione, oltre ad una elevata capacità di elaborazione richiesta.

Tra le varie tecniche sviluppate per gestire un controllo strutturato applicato ad un sistema tempo variante, si può optare per la soluzione di controllo a *gain scheduling.* Questo approccio è molto valido nei sistemi non lineari. Prevede di linearizzare a tratti il sistema in esame e consente di calcolare le diverse matrici utili al controllo. Esistono numerosi schemi di controllo in cui si utilizza la tecnica del gain scheduling, dall'impiego di semplici PD di cui si vogliono adattare i parametri fino a soluzioni più innovative [5], [6]. Questa tecnica di controllo non prevede l'esistenza di un modello non lineare di partenza, ma mira subito ad una linearizzazione a tratti.

Partendo da queste considerazioni, questo lavoro di tesi presenta una tecnica di controllo di vibrazioni che si basa sull'impiego di un controllore ottimo LQR modale applicato ad un sistema meccanico non lineare tempo variante. Il sistema è un manipolatore planare a tre bracci controllato in posizione tramite tre regolatori PD. Tale sistema riproduce le dinamiche di un braccio per la distribuzione di calcestruzzo.

Al fine di superare i limiti legati all'onere computazionale precedentemente introdotti, viene illustrata l'identificazione tramite *reti neurali artificiali* del sistema. I dati raccolti per l'identificazione a scatola nera provengono da una co-simulazione MSC-ADAMS e MATLAB-Simulink. Questo approccio consente di generare un modello non lineare robusto che possa essere linearizzato a tratti nelle configurazioni di interesse per le operazioni che deve svolgere il robot, ispirandosi alle *reti di modelli lineari locali* [7]. Una volta ottenuto il modello completo, si può implementare un anello di controllo esterno con l'obiettivo di limitare le vibrazioni sulla struttura. Tale anello di controllo impiega un LQR già noto in letteratura, aggiornando però la legge di controllo in funzione della configurazione assunta del manipolatore.

Nei capitoli due e tre si richiama brevemente la teoria riguardante l'identificazione dei sistemi non lineari e lo stato dell'arte delle reti neurali, supportando la scelta delle reti neurali NARX addestrate secondo l'algoritmo Levemberg-Marquardt. L'approccio al problema è affrontato gradualmente dal capitolo quattro, partendo dal caso semplificato di un sistema meccatronico costituito da un controllore PD e da un sistema non lineare ad un grado di libertà.

Le considerazioni a supporto delle scelte effettuate sui dati di addestramento della rete neurale nel caso a un grado di libertà sono approfondite e adattate al caso di studio effettivo, il sistema meccatronico a tre gradi di libertà, che risulta essere un sistema non lineare MIMO del secondo ordine (capitolo cinque). Il fatto che sia un sistema MIMO presenta alcune difficoltà, a partire dalla scelta del miglior modello possibile. Per questa scelta sono state addestrate quattro reti a partire da set di addestramento differenti, e si è costruita una serie di configurazioni di test secondo cui far evolvere i quattro modelli. Questo gruppo di configurazioni rappresenta una traiettoria che più si avvicina a quella realmente compiuta dal robot durante le normali condizioni operative.

Per poter impiegare il modello non lineare NARX di ordine due così ottenuto per implementare un controllo di vibrazioni, è stato necessario linearizzare il modello in diverse configurazioni del robot e successivamente calcolare le matrici di stato del modello. Le frequenze proprie identificate sono state confrontate con quelle ricavate dalla funzione di risposta in frequenza della cosimulazione.

Il controllo aggiorna le matrici in funzione della configurazione corrente, basandosi sull'informazione ottenuta da un gruppo di sensori predisposti. In questo modo si ottiene un'azione online adattativa sul sistema, lasciando offline solamente l'identificazione a rete neurale.

2. Identificazione: tratti generali

Per la cultura occidentale, la consapevolezza dell'importanza di definire dei modelli per descrivere i sistemi in relazione con l'uomo ha radici molto antiche, basti pensare ad Aristotele che riconosceva l'importanza delle relazioni geometriche e numeriche tra diverse quantità fisiche che descrivono i fenomeni più disparati. Occorre riservare lo stesso riguardo ad un pilastro dell'astronomia come Tolomeo (85-165), che si accorse come gli stessi fenomeni astronomici da lui osservati potessero essere descritti da modelli differenti, con un diverso grado di accuratezza [3]. È la disciplina chiamata *identificazione* che ha il compito di costruire i modelli matematici e che di fatto collega questi due mondi, quello dei fenomeni osservati e quello dei modelli [4].

Essa si avvale di numerose tecniche, che rappresentano il complesso di metodologie atte a definire le caratteristiche di un generico sistema, a partire da misurazioni sperimentali [8]. Il sistema è dotato di ingressi e uscite, e questi metodi si basano sull'analisi della risposta del sistema agli ingressi, disturbi compresi. Sono tecniche impiegate in numerosissimi campi, che vanno ad identificare piccole realtà, come un componente fisico, oppure grandi complessi, come interi processi e impianti industriali.

Il modello matematico utilizzato nel procedimento di identificazione può essere di diversi tipi, in quanto esistono numerose classificazioni che si basano su criteri disparati. La distinzione che più interessa questo lavoro di tesi è quella tra *modello a scatola nera* (*black box*) oppure *parametrico*. Se si effettua la prima scelta non è necessario avere un modello matematico strutturato e ben definito, che invece è fondamentale se si vuole operare con un modello parametrico, che descrive le leggi che riproducono il comportamento di tutti i componenti del sistema e a cui è associato un significato fisico [9]. I metodi parametrici tendono infatti ad identificare i parametri fisici che fanno parte del modello preso in considerazione, e di norma le non linearità del sistema sono rappresentate da una funzione analitica nota a priori. Nel caso di un generico sistema meccanico, attraverso l'uso di un modello parametrico, si identificano proprio i parametri (m, c, k). Un altro tipo di classificazione è quella che presenta il numero di ingressi e di uscite: un modello SISO (Single Input Single Output) è il più semplice e anche quello che presenta le limitazioni più grandi, in quanto nella realtà è più facile trovare processi multivariabili. Un MIMO (Multiple Input Multiple Output) è proprio un modello multivariabile, la cui complessità può essere ridotta disaccoppiandolo in molteplici sistemi MISO (Multiple Input Single Output), ciò però non è sempre attuabile in pratica ma solo in simulazione.

L'identificazione può essere effettuata sia nel dominio del tempo sia nel dominio delle frequenze. Solitamente la funzione di risposta in frequenza è quella più usata nell'ambito dell'analisi modale per descrivere la relazione ingresso-uscita di un sistema lineare. Questo perché una delle sue proprietà è proprio quella di essere invariabile in ingresso, quindi permette di poter utilizzare un ampio range di frequenze in ingresso e di ottenere la stessa funzione [8]. Per le strutture non lineari questo non è possibile, in quanto i coefficienti contenuti del modello matematico della FRF non sono più costanti

ma dipendono dall'ampiezza. La discriminante tra modelli lineari e non lineari è infatti la possibilità di applicare il principio di sovrapposizione degli effetti nel solo caso lineare, ma a volte basta un semplice sistema lineare per descrivere in modo accurato un sistema affetto da non linearità. Saranno presentati solo i modelli non lineari in quanto, per quanto affermato nel capitolo precedente, un modello lineare non conterrebbe le informazioni utili a coprire un vasto range di funzionamento del manipolatore studiato.

2.1 Classi di modelli

2.1.1 Classi di modelli non lineari: dinamica esterna e dinamica interna

Le classi di modelli presentate appartengono al gruppo delle strategie di identificazione *a dinamica esterna.* Oltre ad essere quelle maggiormente impiegate, si basano su modelli ingresso-uscita e il nome *a dinamica esterna* deriva dal fatto che possono essere separate due parti: un *approssimante statico non lineare* e un *filtro esterno dinamico*.

Con approssimante statico non lineare si indica la funzione non lineare che lega gli ingressi all'uscita, mentre con il filtro dinamico esterno si applica una serie di ritardi agli ingressi [4]. In Figura 2.1 si riporta uno schema esplicativo di un sistema con un ritardo del terzo ordine:

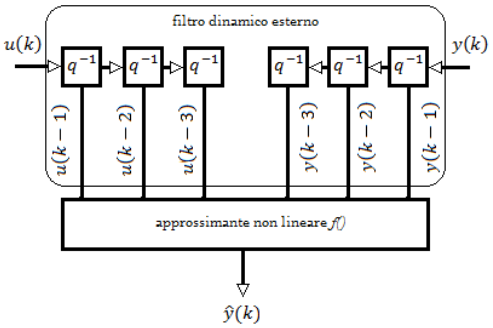

Figura 2.1 Schema sulla strategia a dinamica esterna

Questa classe di modelli si contrappone ad un'altra, quella a *dinamica interna*, che viene qui presentata in modo sommario. I modelli possono essere scritti impiegando la seguente rappresentazione di stato:

$$\begin{matrix}\vec{\boldsymbol{x}}(\boldsymbol{t}+\mathbf{1})=\vec{\boldsymbol{h}}\left(\vec{\boldsymbol{x}}(\boldsymbol{t}),\vec{\boldsymbol{u}}(\boldsymbol{t})\right)\\ \vec{\boldsymbol{y}}(\boldsymbol{t})=\vec{\boldsymbol{g}}(\vec{\boldsymbol{x}}(\boldsymbol{t}))\end{matrix} \quad (2.1)$$

dove gli stati che descrivono il modello spesso non hanno una relazione con gli stati del processo descritto, e il numero di questi stati è correlato all'architettura del modello [9].

2.1.2 Classi di modelli non lineari: in dettaglio

Tra le tre più comuni classi di modelli non lineari si trovano: NARX (Non linear AutoRegressive eXogenous), NARMAX (*Non linear AutoRegressive and Medium Average eXogenous*) e NOE (*Non linear Output Error*). Essi rappresentano rispettivamente l'estensione non lineare dei più classici ARX, ARMAX e OE, ottenuta sostituendo l'espressione di combinazione lineare con una funzione generica non lineare:

NARX:

$$\hat{y}(t) = f\big(y(t-1), \dots, y(t-n), u(t-d), \dots, u(t-d-m+1)\big) \quad (2.2)$$

NARMAX:

$$\hat{y}(t) = f(y(t-1), \dots, y(t-n), u(t-d), \dots, u(t-d-m+1), e(t-1), \dots, e(t-m)), \quad (2.3)$$

NOE:

$$\hat{y}(t) = f\big(\hat{y}(t-1), \dots, \hat{y}(t-n), u(t-d), \dots, u(t-d-m+1)\big) \quad (2.4)$$

Ognuno di questi modelli può essere scritto anche in forma di stato che ha la seguente espressione:

$$\begin{aligned}\vec{\hat{x}}(t+1)&=\vec{h}\left(\vec{x}(t),\vec{u}(t)\right)\\ \vec{\hat{y}}(t)&=\vec{g}(\vec{x}(t))\end{aligned} \qquad (2.5)$$

Se tutti gli stati $\vec{x}$ possono essere misurati, l'identificazione del modello non lineare in forma di stato è equivalente all'approssimazione delle funzioni all'interno del vettore $\vec{h}(\,)$ che descrivono l'aggiornamento dello stato e delle funzioni interne a $\vec{g}(\,)$ che generano l'equazione di uscita.

Esistono anche altre due classi di modelli che consentono una descrizione del rumore più complessa: i NBJ (*Non linear Box and Jenkins*) e I NARARX. Essi però sono molto poco comuni in quanto questo aspetto, che può essere un vantaggio in alcuni casi applicativi, risulta offuscato dall'incremento delle dimensioni del vettore d'ingresso e dalla poca flessibilità.

Per ciò che si può affermare sul secondo svantaggio, ai NBJ e ai NARARX si preferiscono i NARX e i NOE, anche se risulta cruciale la scelta dell'ordine del modello m. La scelta è fondamentale in quanto si riflette sulla performance del modello e perché non sono disponibili dei metodi che lo determinino in modo efficiente. Questo svantaggio è accentuato una volta che si considerano i differenti ordini m_u e m_y per gli ingressi e le uscite oppure un tempo morto d. Bisogna generalmente ricorrere al sistema *trial and error* lasciato al progettista oppure a metodi sofisticati, ma per una trattazione poco più approfondita consultare il Capitolo 5. Un altro svantaggio è che generalmente è difficile provare la stabilità di questi modelli, se non con simulazioni o programmi che ne verificano la stabilità.

Nonostante queste caratteristiche negative, essi forniscono un'espressione compatta del modello. Se il modello viene scritto in questo modo:

$$\vec{\hat{y}}(t) = f(\vec{\varphi}(t)), \qquad (2.6)$$

si pone in evidenza il *vettore regressione*

$$\vec{\varphi}(\mathrm{t}) = [u(t-1)\, u(t-2) \ldots u(t-m)\, y(t-1) \ldots y(t-m)]^T. \qquad (2.7)$$

Come conseguenza della maggior compattezza rispetto agli altri modelli si ha che il vettore di regressione presenta pochi ingressi e ciò comporta una dimensione relativamente ridotta dello spazio del vettore degli ingressi. Questo aspetto può apparire meno importante nei sistemi lineari ma in quelli non lineari risulta fondamentale, specialmente nel caso multivariabile.

2.2 Confronto tra NARX e NOE

Il modello NARX permette un diverso tipo di ottimizzazione a seconda che la funzione $f(\,)$ sia descritta da un approssimante lineare o non lineare. Nel caso di approssimante lineare, la struttura del modello NARX permette l'uso di tecniche di ottimizzazione lineari come i *LS* (*Least Squares*), mentre nel caso non lineare consente l'impiego di metodi di ottimizzazione a gradiente. Per il modello NOE invece si prevede l'uso di tecniche di ottimizzazione a gradiente in entrambi i casi. Questo aspetto non rappresenta un netto vantaggio del modello NARX sul NOE, e in effetti molti aspetti che li caratterizzano non fanno emergere un netto *vincitore:* mentre il NOE consente un miglior raggiungimento dell'obiettivo del modello, il NARX offre dei vantaggi come un addestramento più semplice e veloce [4], [3].

In ogni caso occorre fare attenzione all'effetto di accumulazione dell'errore, che rappresenta un pericolo dal punto di vista della stabilità del modello. Per fare fronte a questa problematica si tratterà in seguito della *modellazione lineare locale*, che rappresenta un metodo semplice per l'estrapolazione di un comportamento stabile del modello.

2.3 Curse of dimensionality

Nei problemi MIMO è ricorrente una problematica rilevante, che lega tra loro l'ordine del modello e il numero di dati necessari per l'identificazione. Questo fenomeno è indicato con il nome *curse of dimensionality* , un termine coniato dal matematico Richard Bellman per descrivere diversi tipi di fenomeni associati all'analisi dei dati in spazi multi-dimensionali di cardinalità elevata [4]. Corrisponde al fenomeno di crescita esponenziale delle dimensioni dello spazio di dati (utili all'identificazione) in ordine con la dimensione degli ingressi. In Figura 2.2 si illustra una semplificazione del fenomeno:

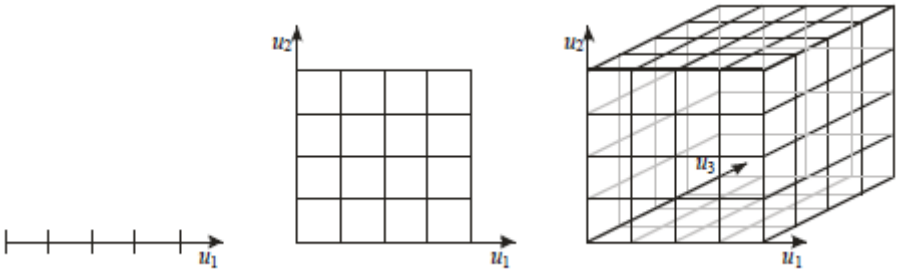

Figura 2.2. Esempio della crescita esponenziale dei dati

Il curse of dimensionality è una proprietà intrinseca del problema e non dipende dal tipo di modello scelto e impiegato per l'identificazione [4]. Un piccolo esempio di come le dimensioni dello spazio di un problema possono divergere se la cardinalità è alta, dopo una semplice operazione di elevamento a potenza, è il seguente:

$$sistema\ 1D \rightarrow 2 * x = 2^1 x$$
$$sistema\ 2D \rightarrow\ 2 * 2x = 4x =\ 2^2 x$$
$$sistema\ 3D \rightarrow\ 2 * 2 * 2x = 8x =\ 2^3 x$$
$$\dots$$

In ambito pratico, fortunatamente, ci si trova spesso ad operare su sistemi in cui alcune limitazioni evitano che si presenti questo fenomeno. Tra questi si annoverano i sistemi con input ridondanti o correlati tra loro, la cui eliminazione (e conseguente riduzione della cardinalità del problema) rappresenta una comodità in quanto non comporta alcuna perdita di informazione. Altri sistemi invece possiedono "zone non raggiungibili" all'interno dello spazio degli ingressi, che quindi non intervengono all'incremento della cardinalità, in quanto corrispondono a combinazioni impossibili degli input. Un esempio è quello relativo ad un qualsiasi sistema fisico in cui alta pressione e bassa temperatura non possono coesistere. Ad ogni modo, questo fenomeno non può essere sottovalutato quando si ha a che fare con un qualsiasi sistema MIMO.

3. Stato dell’arte: reti neurali

Il sistema viene modellato impiegando le reti neurali, e di seguito si riporta un appunto di teoria.

Le reti neurali artificiali prendono spunto dal funzionamento delle reti neurali naturali biologiche. Nonostante ai giorni nostri si possano progettare e costruire macchine in grado di effettuare precisi compiti con rapidità e accuratezza nettamente superiori a quelle degli esseri umani, esistono ancora problemi in cui le macchine sono superate dagli esseri umani. Basti pensare a come si impara facilmente a scrivere una lettera dell’alfabeto, dopo che ci è stata presentata da un insegnante, oppure come è semplice riconoscere un oggetto da un altro, sulla base dell’esperienza. È anche semplice come l’uomo riesca a raggruppare insieme schemi riconosciuti simili, sempre senza insegnante. È proprio sulla base di queste osservazioni che è nato l’interesse verso il cervello animale, con il duplice intento di capirlo e di emulare i suoi punti di forza [10].

Non si è ancora scoperto quale sia il segreto del suo funzionamento ma una importante constatazione è che i neuroni biologici presi singolarmente sono meno veloci dei componenti delle macchine. Non si può quindi spiegare la capacità di elaborazione degli animali in termini di velocità di elaborazione dei componenti elementari [9]. Attualmente si crede responsabile di queste capacità

l'elevato livello di interconnessione tra i neuroni elementari (10^{12}neuroni nel cervello umano, ognuno dei quali connesso fino a 10^5 neuroni contigui). Ecco da dove ha origine l'interesse nel mimare il comportamento del cervello animale studiando come si possano riproporre tali capacità di apprendimento e risoluzione di problemi tramite le cosiddette *reti neurali artificiali*.

Una rete neurale artificiale, o semplicemente *rete neurale*, è un sistema di processamento di informazioni che ha determinate performance in comune con le reti neurali biologiche. Queste reti artificiali sono state sviluppate come generalizzazione di modelli matematici che hanno lo scopo di descrivere la cognizione umana o la biologia neurale, e si fondano sulle seguenti assunzioni:

- ✓ Le informazioni sono trasmesse da entità elementari, indifferentemente chiamate *neuroni*, *nodi* o *celle*
- ✓ I segnali vengono passati da un neurone all'altro tramite i collegamenti
- ✓ Ad ogni collegamento è associato un peso che moltiplica il segnale trasmesso
- ✓ Ogni neurone applica al suo ingresso (somma dei segnali entranti) una funzione di attivazione che restituisce il suo segnale di uscita. Essa è solitamente non lineare.

Lo schema di connessione di una rete è chiamato *architettura* mentre il metodo secondo cui si associano i pesi alle connessioni è chiamato *algoritmo di addestramento* (o *algoritmo di training*) [10]. In una prima parte si analizzeranno i principali tipi di architetture, introducendo la nomenclatura necessaria. Solo in un secondo momento verranno analizzati gli algoritmi di addestramento impiegati dalle reti neurali.

In figura viene proposto il cosiddetto *neurone standard*, vale a dire l'entità elementare comune a tutte le reti, anche complesse:

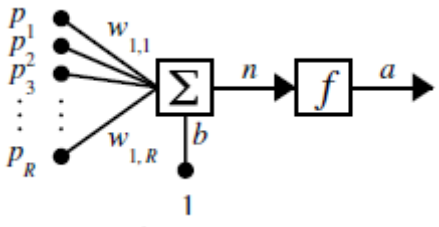

Figura 3.1. Neurone standard

in cui in ingresso vengono passati dei dati p_i che vengono pesati dalle quantità $w_{i,R}$, successivamente sommati tra loro e ad un eventuale costante chiamata *bias* o *threshold* b fino a confluire nel segnale n in ingresso alla funzione di trasferimento neurale f, che dà come risultato l'uscita della rete artificiale.

Per comodità è utile vedere l'uscita del neurone, e a breve dell'intera rete, come il risultato di una serie di operazioni tra vettori, quindi lo schema a cui occorre rifarsi diventa il seguente:

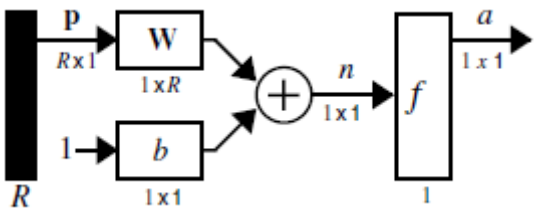

Figura 3.2. Neurone standard in versione vettoriale

in cui gli ingressi sono stati organizzati in un vettore di lunghezza R e i pesi in una matrice $1xR$.

Avendo definito il neurone standard, occorre mostrare come nelle reti complesse esistano architetture composte da diverse combinazioni di neuroni e strati di neuroni. Queste reti sono chiamate semplicemente *reti statiche*, e si differenziano da quelle dinamiche di cui si parlerà in seguito. Una rete in cui sono presenti neuroni nascosti ha questa architettura:

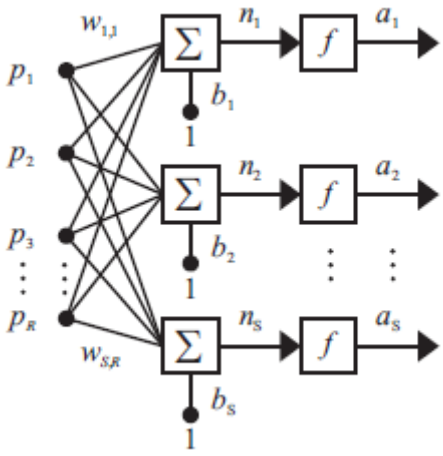

Figura 3.3. Rete a più neuroni nascosti

in cui si possono ancora notare gli input della rete, i pesi, le funzioni base di attivazione f (solitamente multidimensionali e dello stesso tipo), ma l'aspetto più importante è l'evidente infittirsi dei collegamenti, che permette di ottenere un modello più complesso e quindi più aderente al sistema generico. Oltre ai neuroni multipli all'interno di un solo strato, esistono anche reti aventi altri strati oltre a quelli di ingresso e di uscita. Essi sono chiamati *strati (o livelli) nascosti* e, prendendo ad esempio una rete a 3 strati nascosti, sono connessi tramite la struttura di Figura 3.4.

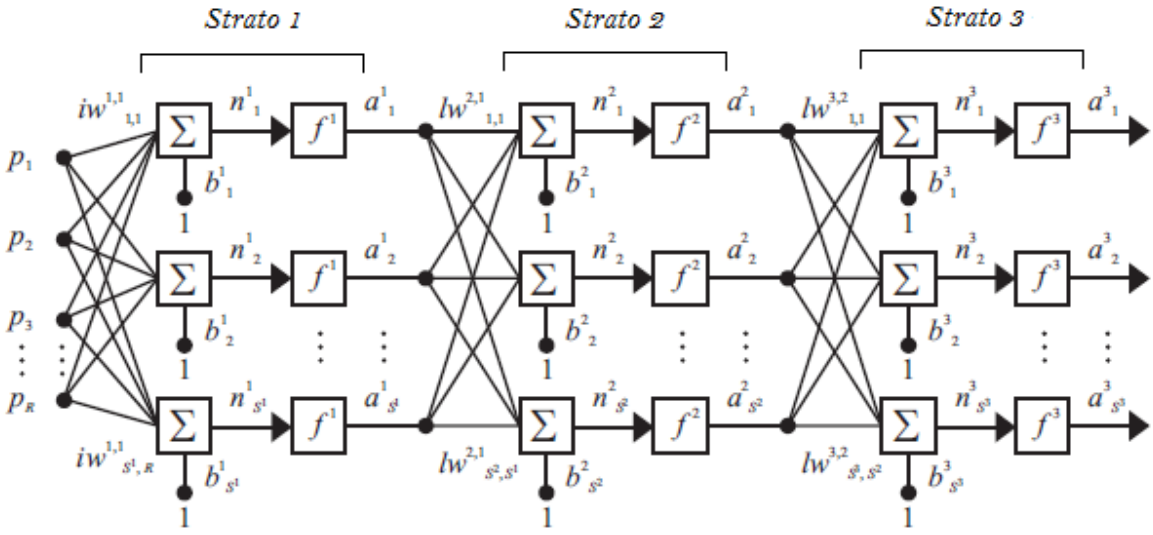

Figura 3.4. Rete a più strati nascosti

Esistono ovviamente alcuni tipi di reti più ricorrenti di altri in funzione del tipo di applicazione. Si annoverano le reti *a singolo strato*[1], che sono le strutture più semplici, oppure le più comuni reti *multistrato a perceptrone* (*MLP*). Per completezza si riporterà solo in seguito anche un accenno alle reti *a base radiale (RBF)*. La differenza più importante tra le reti sopra citate è il tipo di architettura, che evidentemente si riflette anche sul tipo di costruzione del segnale in uscita.

3.1 Costruzione del segnale in uscita

Nelle reti multistrato a perceptrone, nelle quali si riscontra una *costruzione rigida.* Le funzioni di attivazione operano infatti su uno scalare a, generato dalla proiezione del vettore di ingresso sul vettore dei parametri non lineari, ottenendo un prodotto scalare che genera la combinazione lineare illustrata in (3.1).

[1] Con "reti a singolo strato" si vuole intendere una rete avente uno strato di ingresso e uno di uscita. La precisazione si mostra necessaria in quanto in letteratura esistono diversi tipi di convenzioni riguardo al comprendere o meno lo strato di uscita come strato della rete.

$$\boldsymbol{a} = \underline{\boldsymbol{w}}^{(\boldsymbol{n1})}\underline{\tilde{\boldsymbol{p}}} = \boldsymbol{w_o}^{(\boldsymbol{n1})}\,\boldsymbol{p_o} + \boldsymbol{w_1}^{(\boldsymbol{n1})}\,\boldsymbol{p_1} + \ldots + \boldsymbol{w_m}^{(\boldsymbol{n1})}\,\boldsymbol{p_m}, \qquad (3.1)$$

In cui $\underline{\tilde{p}} = \begin{pmatrix} 1 \\ p \end{pmatrix}$ è il vettore degli ingressi. Dato che si è deciso di incorporare nel vettore degli ingressi anche il valore costante del bias, il primo input è fittizio in quanto è stato posto pari a 1, mentre i restanti input sono gli ingressi veri e propri del modello. Per la presenza del bias il vettore degli ingressi può essere definito un *vettore allargato.*

Per dare un'interpretazione all'espressione si può seguire questo ragionamento: se il vettore di parametri non lineari e il vettore allargato di ingressi hanno la stessa direzione, l'attivazione a è massima. Nello stesso modo, se i due vettori hanno direzione opposta a risulta minima, e se sono ortogonali tra loro a è zero.

Un altro modo per ottenere il segnale della rete è la *costruzione radiale*, utilizzata dalle *reti a base radiale* (*radial basis function*). Lo scalare a non è più combinazione lineare dei pesi e degli ingressi ma viene calcolato come la distanza tra il vettore degli ingressi e il centro della funzione base:

$$\boldsymbol{a} = \left\| \underline{\boldsymbol{p}} - \underline{\boldsymbol{c}} \right\| = \sqrt{(\underline{\boldsymbol{p}} - \underline{\boldsymbol{c}})^{\boldsymbol{T}}\,(\underline{\boldsymbol{p}} - \underline{\boldsymbol{c}})}\,, \qquad (3.2)$$

Un altro tipo ancora è la *costruzione prodotto tensore*, utilizzata per diversi tipi di reti neuro-fuzzy.

Dopo aver considerato i diversi tipi di costruzione del segnale, si procede ad una differenziazione più approfondita delle reti. Quelle presentate sono appunto le reti a singolo strato, le MLP e le RBF.

3.2 Tipologie di reti neurali

3.2.1 Reti a singolo strato

Le *reti a singolo strato* sono quelle che hanno un solo strato di pesi di connessione [10]. Solitamente le unità si distinguono in *unità di ingresso* (che ricevono segnali dal mondo esterno) e *unità di uscita* (dalle quali si può leggere la risposta della rete). Esistono diversi tipi di collegamenti in una rete *single layer*, ad esempio la configurazione più semplice è quella in cui le unità di ingresso sono tutte completamente connesse alle unità di uscita, senza che siano connesse ad altre unità di ingresso (come le unità di uscita non sono connesse con altre unità di uscita). In questo modo i pesi su un'unità di uscita non influenzano i pesi di altre unità di output.

3.2.2 Reti multistrato a perceptrone (MLP)

È chiaro come sia utile introdurre le *reti neurali multistrato* al fine di mappare problemi più complessi. Queste reti rappresentano infatti il tipo di architettura maggiormente conosciuto ed utilizzato, e addirittura in molte pubblicazioni le MLP sono usate come sinonimo di reti neurali artificiali [4]. Una rete multistrato presenta uno o più strati di nodi tra le unità di ingresso e le unità di uscita.

Si ha tipicamente uno strato di pesi tra due livelli adiacenti di unità, che possono essere strato d'ingresso e strato di uscita come anche livello d'ingresso e livello nascosto o ancora livello nascosto e livello di uscita. Esse possono risolvere problemi più complessi delle reti a singolo strato ma hanno come svantaggio la possibilità che l'addestramento possa risultare più difficile. L'addestramento può tuttavia rivelarsi efficace e rendere la rete in grado di risolvere problemi per i quali le reti a singolo strato non possono essere addestrate, e che non li rappresenterebbero correttamente in quanto troppo semplici. Questo tipo di reti è

addestrato impiegando un algoritmo chiamato *backpropagation* per cui, nel corso degli anni, sono state ideate diverse varianti che consentono maggiori velocità esecutive. Questo aspetto si tratterà nel prossimo capitolo.

In Figura 3.5 è rappresentato un neurone nascosto di una rete multistrato, avente più ingressi e un'uscita.

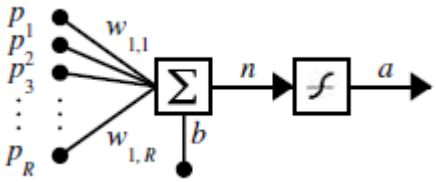

Figura 3.5 Perceptrone

Questo singolo neurone è chiamato *perceptrone*, e svolge un'operazione che può essere vista come separata in due parti distinte. Come prima azione pesa gli ingressi, proiettando ognuno di essi sul peso corrispondente, mentre in un secondo momento prende tutti gli ingressi pesati, li somma ad un eventuale bias e li porta in ingresso ad una funzione di attivazione non lineare, generando il segnale in uscita dal perceptrone.

La funzione di attivazione è scelta di modo che abbia certe caratteristiche quali la continuità, la differenziabilità, essere monotona non-decrescente, e per efficienza computazionale occorre che la sua derivata sia semplice da calcolare.

Esistono alcune funzioni che risultano più adatte ad essere impiegate come funzioni di attivazione rispetto ad altre, e questo perché la derivata di esse (rispetto ad un certo valore della variabile indipendente, ad esempio t) può essere espressa in termini del valore della funzione (in corrispondenza di quel valore di t). Nelle reti neurali si utilizzano spesso le funzioni di trasferimento sigmoidali in quanto godono della seguente proprietà:

$$\frac{d}{dt} sig(t) = sig(t)\big(1 - sig(t)\big),$$

indicando con $sig(t)$ la generica funzione sigmoide. Questa proprietà risulta comoda da implementare, e ciò si riflette direttamente sulla velocità di computazione [10]. Solitamente ci si aspetta che tale funzione di attivazione saturi, e una delle più tipiche è la *funzione logistica*, con un range tra 0 e 1:

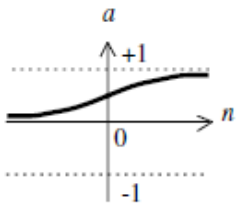

Figura 3.6. Funzione logistica: $\boldsymbol{a = logsig(n)}$

Un'altra è la *tangente sigmoidale* (che non è altro che la tangente iperbolica), con un range tra -1 e 1, comodo in quanto simmetrico:

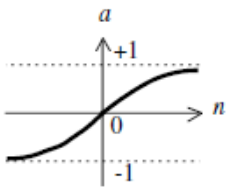

Figura 3.7. Tangente sigmoidale: $\boldsymbol{a = tansig(n)}$

oltre alla funzione di trasferimento neurale lineare, la cosiddetta *purelin*, tangente alla *tansig* nell'origine:

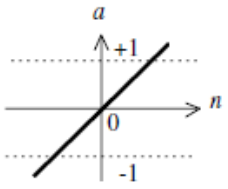

Figura 3.8. Funzione lineare: $\boldsymbol{a = purelin(n)}$

La forma dei dati è un fattore importante nella scelta di una funzione piuttosto che di un'altra, e risulta molto diffuso l'utilizzo di *tansig* e *purelin*.

3.2.3 Reti a funzione base radiale (RBF)

Inizialmente usate senza una connessione con le reti neurali artificiali, erano impiegate in matematica come funzioni di interpolazione [10]. Esse hanno una struttura in feedforward che consiste in uno strato nascosto composto da m unità sincronizzate che sono connesse ad uno strato di uscita composto da altre m unità. Tutte le unità nascoste hanno come ingresso un vettore reale n-dimensionale ma non sono più presenti i pesi dello strato nascosto [11]. Come affermato in precedenza, l'uscita di ogni unità nascosta è calcolata in base alla vicinanza dell'input al parametro associato all'ennesima unità nascosta che rappresenta il centro della funzione base. Permettono di implementare algoritmi di addestramento più veloci rispetto alle reti MLP, in particolare danno prestazioni superiori nei casi di processamento adattativo dei segnali o nel controllo adattativo quando non si possono salvare dati che arrivano molto rapidamente [11].

In conclusione a questo appunto teorico sull'architettura delle principali tipologie di reti neurali, si riportano le principali proprietà di quelle che, come affermato antecedentemente, sono le reti più interessanti tra quelle visionate. Le reti multistrato a perceptrone.

3.3 Proprietà delle reti MLP

- ✓ L'ottimizzazione parametrica deve essere spesso migliorata utilizzando una tecnica di ottimizzazione non lineare che però rallenta l'esecuzione.
- ✓ La velocità di addestramento è bassa, a causa del fatto che occorre applicare tecniche di ottimizzazione non lineari, che inoltre vengono

ripetute per diverse inizializzazioni dei pesi della rete se non si è raggiunto un risultato soddisfacente. Per questo motivo si usano algoritmi di addestramento a cui sono apportate modifiche atte a velocizzarli [11]. Nel prossimo capitolo viene riportata la presentazione di alcuni algoritmi veloci, tra cui quello impiegato nell'addestramento della rete.

- ✓ L'accuratezza è molto elevata, e spesso si preferisce questa tipologia di architettura alle altre in quanto permette di ottenere simili livelli di accuratezza ma con un minor numero di neuroni e di pesi sugli strati. Dato che comporta un maggior livello di compattezza, richiede anche lunghi tempi per l'addestramento.
- ✓ Il fenomeno *curse of dimentionality* descritto nel capitolo precedente non si presenta spesso perché il meccanismo di costruzione del vettore delle uscite utilizza le proiezioni, e questo riduce il numero di dati [12].
- ✓ L'interpretazione delle proiezioni non è però molto agevole soprattutto quando sono numerose, infatti è semplice pensare e rappresentare spazi multidimensionali [11].
- ✓ L'utilizzo è molto vasto, in accordo con quanto detto sin dall'inizio.

3.4 Algoritmo di addestramento di reti MLP

Dato che una delle principali capacità di una rete è l'abilità di imparare, a fronte dell'interazione con il suo ambiente, un'altra caratteristica per distinguere una rete da un'altra, oltre al tipo di architettura, è il tipo di *apprendimento*. Con *apprendimento* si indica una procedura adattativa attraverso la quale si aggiustano i valori dei pesi della rete in modo incrementale, in quanto si vogliono ottenere determinate performance stabilite a priori [11].

Si farà un'iniziale distinzione tra due diversi tipi di apprendimento: *supervisionato* e *non supervisionato,* escludendo quei tipi di reti neurali per i

quali i pesi vengono fissati senza l'uso di un algoritmo iterativo di addestramento.

La classificazione però non è rigorosa ma varia da autore ad autore, infatti alcuni di essi riconoscono l'esistenza di una terza classe, la *self-supervised training* [11]. In ogni caso si può affermare che esiste una utile corrispondenza tra il tipo di addestramento e il tipo di problema che si vuole risolvere. Di seguito si riportano le sostanziali differenze tra le prime due classi di algoritmi di addestramento sopra citate.

Apprendimento supervisionato: nelle configurazioni di reti neurali più frequenti, l'addestramento è possibile grazie a una sequenza di vettori di addestramento, ognuno associato ad uno specifico vettore target di uscite, e i pesi vengono regolati seguendo un algoritmo di apprendimento che ha come obiettivo la riduzione graduale dell'errore tra input e target.

Apprendimento non supervisionato: le reti raggruppano insieme vettori di ingresso simili senza usare i dati di addestramento. Viene fornita una sequenza di vettori in ingresso ma non sono specificati dei vettori target, e la rete modifica i pesi in modo tale che i vettori di ingresso più simili siano assegnati alle stesse unità di uscita.

Reti a pesi fissati: queste reti possono funzionare bene per problemi per cui risulta difficile adottare le tecniche tradizionali, come problemi con vincoli che non possono essere soddisfatti contemporaneamente. Spesso in questi casi è soddisfacente una soluzione quasi ottima [10].

Mentre le reti a singolo strato sono limitate nelle mappature che possono imparare, le reti neurali multistrato possono imparare ogni mappatura continua con un'accuratezza arbitraria (in alcune applicazioni è necessario introdurre più di uno strato nascosto per ottenere benefici, nonostante solitamente anche un

solo strato sia sufficiente a raggiungere buoni risultati). Ed è per questo che verso gli anni Settanta si è perso di interesse per le reti neurali, in quanto non erano state ancora ideate delle metodologie atte all'addestramento di reti multistrato, e quelle a singolo strato risultavano essere limitate a pochi campi di applicazione [1].

In realtà già verso il 1974 era stato scoperto il metodo per propagare all'indietro l'informazione sull'errore e scegliere i pesi in funzione di questa informazione propagata, ma la scoperta non aveva riscosso molto successo. A questa scoperta, a cui sono arrivati in modo indipendente anche i ricercatori David Parker (1985) e LeCun (1986), ha avuto un grande ruolo nel prendere nuovamente in considerazione le reti neurali per risolvere molti problemi, soprattutto in ambito non lineare [10].

Questo metodo è detto *retropropagazione* (backpropagation) degli errori, o *generalized delta rule*. È un metodo a gradiente decrescente che minimizza l'errore quadratico totale tra l'uscita calcolata dalla rete e il target, ovvero l'uscita desiderata [11]. L'addestramento di una rete tramite retropropagazione è costituito da tre fasi: propagazione in avanti degli ingressi, il calcolo della retropropagazione dell'errore associato, la regolazione dei pesi.

Dopo l'addestramento, l'applicazione della rete coinvolge solo i calcoli della fase di propagazione in avanti, e la rete addestrata fornisce velocemente l'uscita anche se il training è stato lento. Esistono però numerose variazioni al metodo di retropropagazione che sono state sviluppate con l'intento di velocizzare il procedimento di addestramento al fine di rendere competitivo l'uso delle reti stesse. Nel seguito si riporta inizialmente il metodo standard di backpropagation applicato ad una rete composta da diversi strati nascosti, per poi passare a metodi più evoluti quali il *Gauss-Newton* e il *Levenberg-Marquardt*, che

utilizzano il metodo ai minimi quadrati rendendo più rapido il calcolo e sono quelli maggiormente impiegati nella pratica.

3.4.2 Retropropagazione standard

Fase 1: *propagazione in avanti del segnale di ingresso*. Durante questa prima fase, il segnale di ingresso passa da un'unità all'altra fino all'unità di uscita, dove viene calcolato l'errore. Più in dettaglio: ogni unità ingresso riceve un segnale di input e lo trasmette ad ognuna delle unità nascoste. Ognuna di queste effettua la propria attivazione e invia il segnale ad ognuna delle unità di output, che a loro volta effettuano la propria attivazione per fornire la risposta della rete per lo schema di ingresso dato, confrontando il segnale ottenuto con il proprio valore target per determinare l'errore associato a quello schema e riferito a quella unità.

Fase 2: retropropagazione dell'errore. Basandosi sul calcolo di quell'errore, si distribuisce all'indietro l'errore relativo all'unità di output (e questa operazione è effettuata da ogni unità all'unità dello strato precedente).

Fase 3: aggiornamento dei pesi. Più avanti, l'errore viene usato per aggiornare i pesi tra l'output e lo strato nascosto. In modo simile si aggiornano i pesi tra stato nascosto e strato in ingresso. Dopo che vengono determinati tutti i pesi dei segnali, si aggiustano simultaneamente i pesi di tutti gli strati.

Il problema dell'apprendimento di una rete neurale può essere visto come un problema di ottimizzazione. Il processo di apprendimento può infatti essere interpretato come una ricerca di una soluzione all'interno di uno spazio multidimensionale di parametri, che sono i pesi, la quale ottimizza gradualmente una certa funzione predefinita [11]. In specifico, occorre determinare quali siano i migliori parametri stimati (pesi e bias) con l'obiettivo di minimizzare l'errore

in uscita. Risulta quindi necessario un algoritmo che renda possibile questa operazione.

Di seguito si riportano i principali algoritmi impiegati nei problemi di ottimizzazione di funzioni generiche, ossia l'algoritmo *Gauss Newton* e soprattutto la sua versione alternativa: l'algoritmo di *Levemberg-Marquardt*.

3.4.3 Algoritmi per risolvere problemi ai minimi quadrati

Si definisce *problema ai minimi quadrati* il seguente: si trovi $\boldsymbol{x}^*$, un minimo locale per la funzione:

$$F(x) = \frac{1}{2}\sum\big(f_i(x)\big)^2, \qquad (3.3)$$

dove $f_i : R^{\mathrm{n}} \to R$, $i = 1,2 \ldots . m$ sono funzioni assegnate, con $m \geq n$.

Esistono diversi metodi per risolvere problemi a minimi quadrati. Tutti i metodi di ottimizzazione non lineare sono iterativi, cioè da un punto iniziale x_0 il metodo risolutivo produce una serie di vettori $x_1,\ x_2, \ldots$, che nel migliore dei casi converge a x^* (un minimo locale della funzione assegnata) [13].

Molti metodi seguono delle procedure che applicano la condizione decrescente:

$$F(x_{k+1}) < F(x_k) \qquad (3.4)$$

ad ogni passo di iterazione e sono quindi chiamati *descent methods*. I metodi *steepest descent method, newton's method* e *line search* soddisfano la condizione decrescente ad ogni passo di iterazione (ma in questa trattazione sarà presentato solo il principio generale che sta alla base di un *descent method)*. Ogni passo consiste nel trovare una direzione decrescente h_d e la lunghezza del

passo che fornisce una buona decrescita, per un maggiore dettaglio si riporta quanto segue.

Si consideri una variazione di F lungo la direzione h e la seguente espansione di Taylor:

$$F(\vec{x} + \alpha\vec{h}) = F(\vec{x}) + \alpha\vec{h}^T\boldsymbol{F}'(\vec{x}) + O(\alpha^2) \cong F(\vec{x}) + \alpha\vec{h}^T\boldsymbol{F}'(\vec{x}) \qquad (3.5)$$

per α piccolo.

Si dice che h è una direzione decrescente se $F(x + \alpha h)$ è funzione decrescente di α per $\alpha = 0$.

3.5. Differenziazioni al metodo tradizionale

3.5.1 Trust region e damped method

Si può assumere di avere un modello L del comportamento di F nell'intorno dell'iterazione corrente x, tanto migliore quanto h è sufficientemente piccolo:

$$F(x + \vec{h}) \cong L(\vec{h}). \qquad (3.6)$$

In un metodo definito *trust region method* si assume di conoscere un intero positivo Δ tale che il modello sia abbastanza accurato all'interno di una sfera di raggio Δ centrata in x, e il passo si determina nel modo seguente:

$$h = h_{tr} = argmin_{\|h\|<\Delta}\{L(h)\}. \qquad (3.7)$$

In un *damped method* invece il passo si determina così:

$$h = h_{dm} = argmin_h(L(h) + \tfrac{1}{2}\mu h^2), \qquad (3.8)$$

in cui è stato aggiunto il parametro di smorzamento $\mu \geq 0$. Il termine in cui è presente tale parametro serve a penalizzare i passi lunghi.

Si parla di un *problema ai minimi quadrati non lineare* quando chiaramente la funzione f è non lineare. Per questi tipi di problemi si introducono dei metodi speciali, più efficienti dei generici metodi di ottimizzazione. Quelli qui presentati sono metodi che, in molti casi, raggiungono la convergenza meglio di altri [2], [21].

3.5.2 Gauss-Newton method

Può essere facilmente definito come il *metodo base* per quelli più efficienti e si fonda sull'idea di implementare le derivate prime delle componenti del vettore funzione. Si basa sull'idea di approssimare linearmente le componenti del vettore f nell'intorno di x, trovandone un modello lineare. Dall'equazione seguente:

$$f(x+h) \cong l(x) \equiv f(x) + J(x)h \qquad (3.9)$$

in cui $J(x)$ è lo jacobiano, si passa alla prossima (omettendo i passaggi intermedi):

$$(J(x)^T J(x))h_{gn} = -J(x)^T f(x). \qquad (3.10)$$

Quest'ultima equazione è utile al calcolo dello step h_{gn} che minimizza $L(h)$:

$$h_{gn} = argmin_h\{L(h)\}. \qquad (3.11)$$

3.5.3. Levemberg-Marquardt method

Il metodo *Levenberg-Marquardt* è un algoritmo tanto semplice quanto robusto e si può definire un metodo Gauss-Newton smorzato, infatti come visto sopra apporta delle modifiche all'algoritmo precedente inserendo un termine dipendente da μ, detto *coefficiente di damping*. Analogamente al G*auss-Newton method*, consiste nel risolvere la seguente equazione in funzione di h_{lm}:

$$(J(x)^T J(x) + \mu I) h_{lm} = J(x)^T f(x)\,, \qquad (3.12)$$

Ad ogni iterazione il fattore di smorzamento viene aggiornato ed è proprio questo parametro a guidare il processo di ottimizzazione. Accade infatti che se l'errore E decresce velocemente allora può essere usato un valore di μ più piccolo e al contrario se la decrescita di E risulta insufficiente occorre aumentare il valore del coefficiente di smorzamento.

Gli effetti dell'inserzione di μ sono notevoli:

- ✓ per tutti i $\mu \geq 0$, h_{lm} è una direzione decrescente, mentre nell'algoritmo originale era necessario verificarlo
- ✓ a grandi valori del parametro μ corrisponde un piccolo passo nella direzione più ripida, ed è comodo quando l'iterazione corrente è lontana dalla soluzione
- ✓ se il parametro μ è molto piccolo (e ciò accade in corrispondenza di una veloce decrescita dell'errore $f(x)$) allora $h_{lm} \approx h_{gn}$, che è un buon risultato nei passi finali dell'iterazione.

3.6. Levenberg Marquardt per le reti neurali

Quest'ultimo algoritmo presentato è uno di quelli più utilizzati nella pratica per l'addestramento delle reti neurali. Per effettuare l'addestramento delle reti si è impiegato il Neural Networks Toolbox di Matlab, scegliendo tra gli algoritmi di addestramento proprio il Levemberg-Marquardt. La scelta deriva dal fatto che lo si considera il più veloce algoritmo di addestramento per le reti neurali feedforward di media grandezza [help Matlab *trainlm*]. Un altro pregio che presenta l'algoritmo è la possibile estensione all'apprendimento di reti aventi molteplici unità [11], che sono quelle più interessanti da scegliere come modello del sistema meccatronico. Tra le limitazioni di questo algoritmo sono presenti la sensibilità ai pesi iniziali della rete e la tendenza all'overfitting di rumore ma, nel caso si presentassero inconvenienti di questo tipo, sono state ideate diverse tecniche tralasciate in questa trattazione, per il cui approfondimento si rimanda a testi specifici.

Per una rete neurale multistrato viene utilizzata la seguente regola a gradiente decrescente:

$$\vec{w}_{k+1} = \vec{w}_k - \eta \frac{\partial E(\vec{w})}{\partial \vec{w}_k}, \qquad (3.13)$$

in cui $\vec{w}$ è il vettore dei pesi della rete, η è chiamato *learning rate* e $E(\vec{w})$ è la funzione errore, che si può esprimere così:

$$E(\vec{w}) = \frac{1}{2} \sum_{k=1}^{M} (d_k - y_k)^2 \qquad (3.14)$$

con $\vec{d}$ le uscite desiderate della rete, chiamate *target*, le uscite proprie della rete $\vec{y}$ e M la dimensione dello spazio degli output [11]. L'obiettivo è proprio quello di minimizzare la funzione errore, che sopra è scritta come la somma degli errori

quadratici, anche se spesso al posto della somma degli errori quadratici si trova l'*errore quadratico medio*.

È l'errore sull'uscita $(d_k - y_k)$ ad essere propagato all'indietro nella rete e in ogni caso ad ogni passo occorre che si verifichi la condizione:

$$E(\vec{w}_{k+1}) < E(\vec{w}_k). \qquad (3.15)$$

Dato che si è impiegato proprio il metodo Levenberg-Marquardt nell'addestramento delle reti neurali che descrivono il modello meccatronico, si mostrano i passi dell'algoritmo:

- ✓ Calcolo dello Jacobiano $J(x)$ della funzione errore $f(x)$
- ✓ Calcolo del termine $J(x)^T f(x)$
- ✓ Risoluzione dell'equazione $(J(x)^T J(x) + \mu I)h_{lm} = J(x)^T f(x)$ al fine di trovare h_{lm}
- ✓ Aggiornamento del vettore dei pesi della rete grazie a h_{lm}
- ✓ Nuovo calcolo della somma dell'errore quadratico (usando il nuovo vettore di pesi)
- ✓ Se la somma dell'errore quadratico non è diminuita, effettuare un nuovo calcolo del vettore dei pesi, aumentare μ andando a moltiplicarlo per un vettore di aggiustamento v, dopodiché tornare al passo in cui si risolve l'equazione $(J(x)^T J(x) + \mu I)h_{lm} = J(x)^T f(x)$
- ✓ Altrimenti diminuire il valore di μ e fermarsi

Ad ogni iterazione è impiegata una funzione chiamata *gain ratio:*

$$\varrho := \frac{(F(x) - F(x_{new}))}{L(0) - L(h_{lm})} \qquad (3.16)$$

che ha lo scopo di valutare la bontà del modello in funzione dei parametri valutati al valore corrente dell'iesima iterazione.

Si riporta un tipico andamento di μ nella scelta di una rete neurale:

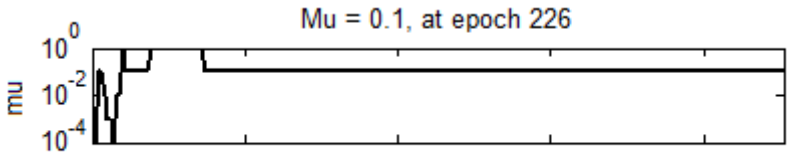

Figura 3.9. Andamento di μ in funzione delle epoche

4. Caso di studio di un sistema vibrante non lineare a 1 gdl

Come anticipato, le strategie a dinamica esterna sono quelle maggiormente usate nell'identificazione non lineare. Esistono, ovviamente, anche reti neurali artificiali che impiegano questo tipo di strategie e che, nell'identificazione di sistemi non lineari, consentono di utilizzare una rete statica come approssimante. Molte proprietà dell'approccio a dinamica esterna non dipendono dal tipo di approssimante usata del sistema (che possono essere di diverso tipo).

Occorre definire gli ingressi della rete statica come l'ingresso e l'uscita misurata del blocco *sistema reale*, ma ritardati opportunamente (in base alla scelta fatta sugli ordini del modello scelto). Gli ingressi possono, però, essere anche l'ingresso del blocco *sistema reale* e l'uscita della rete statica, ancora opportunamente ritardati.

Uno degli aspetti importanti da concordare è la scelta del segnale di input della rete dinamica (che viene successivamente ritardato). Questa decisione è più importante rispetto al caso di un sistema lineare, in quanto la complessità dei sistemi non lineari è decisamente maggiore e i dati contengono inevitabilmente maggiori informazioni [14].

A prescindere dal tipo di problema in esame, si riportano di seguito alcuni tra gli aspetti più importanti che riguardano la scelte dell'ingresso:

- ✓ È importante specificare l'obiettivo del modello, in questo caso si vuole individuare la dinamica del robot nella sua complessità.
- ✓ Il set di dati deve avere una lunghezza massima, in quanto è bene che sia grande ma occorre prestare attenzione alla capacità computazionale (per il caso del sistema a 3 gdl esaminato non si sono superati i 1000 campioni per ingresso, per un totale di 6 ingressi ritardati).
- ✓ È necessario avere chiare le caratteristiche dei diversi segnali di input, in questo caso è necessario che essi presentino una dinamica tale da sollecitare almeno un modo di vibrare della struttura.

Spesso accade che i dati raccolti non siano così informativi da permettere di identificare un modello a scatola nera, che sia in grado di descrivere il sistema in tutte le sue condizioni di operatività. È ovviamente una buona idea quella di raccogliere dati quando il sistema si trova in una condizione di operatività con un comportamento più complesso e/o più rilevante di altre, di modo che la manovra contenga più informazioni possibili sul comportamento del sistema [14].

Spesso però risulta più semplice addestrare una rete neurale se il sistema di interesse possiede una o più condizioni nominali di operatività, come ad esempio una velocità o una temperatura di regime. In tal caso è più intuitivo allenare la rete offrendo una porzione maggiore di dati contenenti questa condizione di lavoro. Questa situazione non contempla l'attività svolta da un robot, per cui il riferimento dato in addestramento alla rete dovrà contenere l'intera dinamica della struttura, affinché si possa arrivare allo scopo del controllo di vibrazioni.

Le reti neurali a dinamica esterna possono avere due diverse configurazioni: *modello serie-parallelo* e *modello parallelo*. Queste due soluzioni si differenziano sia per lo schema che per il training della rete. Per prima cosa conviene focalizzarsi sulla distinzione riguardante il tipo di architettura del modello. Questa differenza di architettura è legata al tipo di impiego necessario, infatti, come per l'identificazione di sistemi lineari, il modello dinamico non lineare può essere usato in queste due differenti configurazioni per ottenere due scopi ben diversi: *predizione* o *simulazione*.

Predizione: sulla base degli input e degli output di un processo, il modello predice uno o più step futuri, e lo schema ad esso associato è il *modello serie-parallelo* (si richiede che l'output del processo si possa misurare durante il funzionamento). Esso è uno schema in anello aperto in quanto è necessario conoscere y al posto di utilizzare la retroazione dell'uscita $\hat{y}$.

Simulazione: soltanto in base all'input si può prevedere l'output futuro, ad esso corrisponde il *modello parallelo* (in questo caso non è richiesto di conoscere l'output desiderato). Solitamente le prestazioni nel caso del *modello parallelo* sono inferiori a quelle del *modello serie-parallelo*, e la scelta di uno schema o di un altro è legata alle particolari condizioni di impiego e a determinate assunzioni sulle proprietà del rumore. È comune addestrare una rete in configurazione serie-parallelo e poi chiudere l'anello per generare il modello parallelo [23], [24].

Tra le condizioni di impiego di un modello in simulazione si annoverano tutti quei casi in cui non si può contare sull'uscita del sistema reale, e quindi occorre sostituire il sistema reale con il suo modello (ad esempio quando un sensore è in avaria).

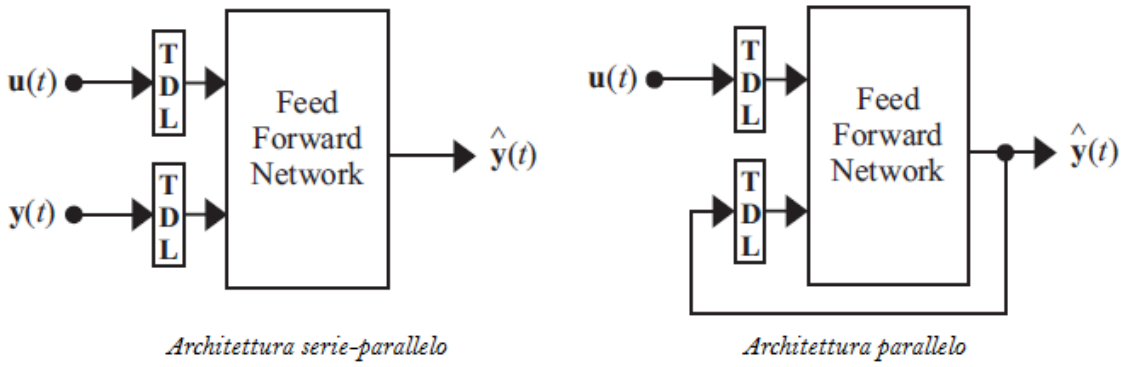

Figura 4.1. Schema esplicativo delle architetture parallelo e serie parallelo

Come affermato inizialmente, le due diverse configurazioni si distinguono anche durante il training: il modello apprende minimizzando una funzione di costo che dipende dall'errore, che nel caso predittivo è chiamata *equation error* mentre nell'altro caso *output error*. Anche se, quando cresce l'orizzonte predittivo, la differenza tra i due diventa sempre più piccola. In questa tesi però il modello parallelo sarà impiegato semplicemente retroazionando l'uscita predetta, di modo che si possano mantenere i pesi della rete in configurazione serie-parallelo. Questa scelta è data dal fatto che le prestazioni di una rete addestrata in questa configurazione sono migliori di quelle associate ad una rete allenata in configurazione parallelo.

In Figura 4.2 si riporta come esempio lo schema di una rete NARX sotto forma di modello serie-parallelo.

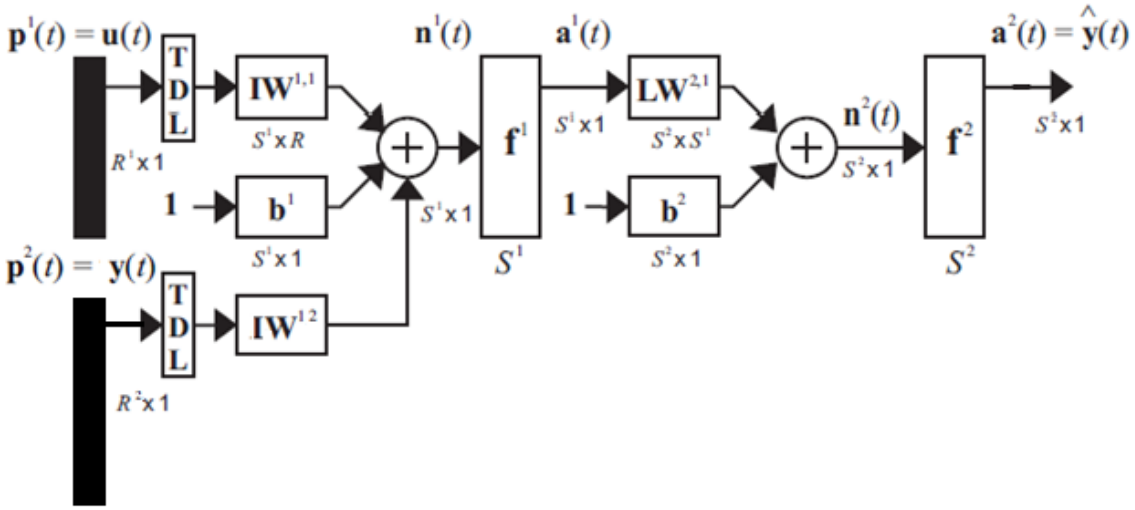

Figura 4.2. Architettura serie-parallelo in dettaglio

Chiudendo l'anello diventa come in Figura 4.3.

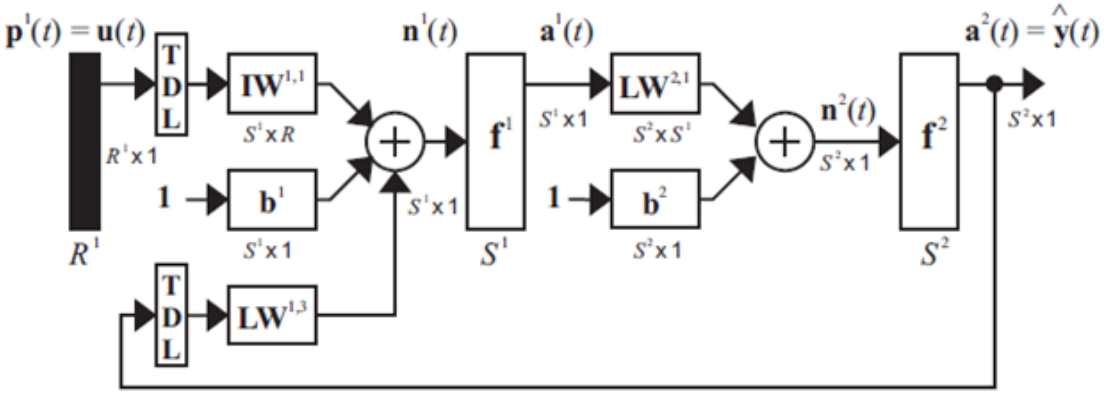

Figura 4.3. Architettura parallelo in dettaglio

4.2 Rete neurale

4.2.1 Creazione della rete

Per addestrare la rete neurale è stato impiegato il *Neural Network Toolbox* di *Matlab*, accedendo alla sezione *time series toolbox*, e in questo caso specifico scegliendo come modello del sistema robot una rete NARX serie-parallelo [25]. La rete NARX è già la NN dinamica (*dynamic time series*) che serve per l'identificazione del sistema robot, di cui si è già precedentemente trattato.

I primi parametri da scegliere sono i ritardi sull'uscita e sull'ingresso (che per comodità sono scelti uguali[2]) e il numero di neuroni nascosti che si vuole imporre. Queste operazioni iniziali servono per dare una struttura a priori del modello, e sono anzi fondamentali dato che in genere è sconsigliabile impiegare un modello NARX non strutturato [9]. Prima di effettuare il training della rete stessa è opportuno scegliere le percentuali di dati da utilizzare per il training, per la validazione e il testing finale. L'opzione migliore è avere una alta percentuale di dati da usare per il training (come il 70% generalmente impiegato) e il restante 30% dividerlo equamente in un gruppo di dati utili alla validazione e al testing [14].

L'addestramento è reso possibile grazie all'algoritmo Levenberg-Marquardt, precedentemente discusso. Al termine dell'addestramento si valutano le performance che caratterizzano la rete appena creata, e se non sono soddisfacenti è possibile modificare alcuni parametri della rete stessa, come modificando l'architettura della rete (utilizzando un diverso numero di neuroni nascosti) e quindi variandone la dimensione. Si può altrimenti agire sul ritardo

[2] Se non si fossero scelti uguali i ritardi, si sarebbe andati incontro al costituirsi di relazioni non sincrone. Queste non avrebbero permesso una trasformazione diretta nel predittore nella solita forma NARX [MultSysId].

da applicare agli ingressi e, volendo, effettuare prove multiple in funzione delle diverse combinazioni di ritardi.

Una volta ultimato l'addestramento che porta alla creazione di una rete soddisfacente, dato che tutte queste operazioni possono essere effettuate anche tramite codice Matlab senza l'utilizzo del toolbox, è utile salvare lo script generato automaticamente dal programma. Questo passaggio è importante per conoscere le funzioni chiamate da Matlab nella creazione della rete e avere chiari quali sono i passaggi che portano il programma al risultato.

4.2.2 Rete neurale: in dettaglio

Inizialmente si prepara l'addestramento per una rete neurale artificiale di tipo NARX, composta da uno strato d'ingresso, formato da un numero limitato di neuroni nascosti (per limitare la complessità della rete), e da uno strato di uscita composto da tre uscite.

La funzione *mapminmax* è necessaria per mappare i segnali in ingresso all'interno di un intervallo tra -1 e 1, dato che il calcolo dei pesi e dei bias della rete è effettuato tenendo conto che essi moltiplicano quantità all'interno di quell'intervallo. I segnali mappati sono appunto pesati, sommati ai bias e infine condotti all'ingresso della funzione sigmoide *tansig,* la quale non è altro che la funzione tangente iperbolica. La funzione di trasferimento dello strato di uscita è invece la funzione lineare *purelin,* dopo la quale occorre riproporre la funzione *mapminmax* ma in versione *reverse* per riottenere i segnali di uscita mappati all'interno del range di valori opportuno. Gli input costanti vengono eliminati dalla funzione *removecostantrows,* e ciò perché i vettori costanti non introducono un contributo informativo significativo al modello [6].

La criticità nel simulare il comportamento della rete una volta addestrata è data principalmente dalla funzione *mapminmax*, a cui è affidato il compito di normalizzazione (e denormalizzazione) dei segnali. L'algoritmo normalizza i segnali passati alla rete in fase di addestramento e tiene in memoria i dati della mappatura tra -1 e 1 (sia per gli input che per i target) che costituiscono un elemento fondamentale della rete e la caratterizzano come la sua architettura e i suoi pesi. Queste due informazioni sono fondamentali una volta creata la rete, in quanto le condizioni di normalizzazione sono applicate ad ogni nuovo segnale in ingresso alla rete. In questo modo essa può essere impiegata in un funzionamento *online*, in quanto accetta in ingresso per ogni vettore un dato alla volta, sul quale applica la mappatura conseguenza dei dati di addestramento. È da questo aspetto che si comprende ancora di più come sia importante la scelta accurata del gruppo di ingressi per l'addestramento.

Per descrivere il sistema a 1 gdl nella maniera più corretta, occorre alimentare la rete neurale con le ϑ°, di modo che generi in uscita le ϑ o le $\dot{\vartheta}$ o entrambe. Si scelgono le $\dot{\vartheta}$ in quanto le velocità angolari contengono l'informazione riguardante il fenomeno vibratorio a cui è soggetto il braccio meccanico.

In seguito ai risultati ottenuti, si è optato per avere nel vettore di ingresso sia le ϑ° che le ϑ, e questo per rafforzare il modello della rete in anello chiuso. Con un modello ϑ°- $\dot{\vartheta}$ si ottengono infatti delle buone prestazioni in anello aperto ma pessime in anello chiuso, le quali invece crescono molto se sono fornite in ingresso alla rete anche le ϑ misurate. Questo accorgimento è necessario perché poter impiegare il modello in anello chiuso permette di poter calcolare i poli del modello e, quindi, procedere all'identificazione.

4.3. Diversi modi per simulare la risposta della rete

Nell'ultima parte della sezione *Creazione della rete neurale*, si accenna alla possibilità di generare un file Matlab automatico per la creazione della rete utilizzando il *Neural Network Toolbox*. Per far fronte alle esigenze di questo caso, questo file è stato modificato rispetto a quello originale del programma (file *AdvancedScript.m* riportato nell'ultima sezione della tesi).
Una delle modifiche principali dello script riguarda la necessità di generare una rete che abbia lo stesso tempo di campionamento (*sample time*) della co-simulazione. Questa richiesta non dipende solo dal fatto che il modello che descrive la rete deve ricevere in ingresso gli stessi segnali del sistema meccatronico, e che quindi deve campionarli nello stesso modo per fornire delle uscite confrontabili. L'altro aspetto è figlio del fatto che può risultare utile che la NN giri online con la cosimulazione.
La velocità di campionamento è pari a $0.01\ sample/sec$ e quindi per imporre questo valore si sovrascrive il sample time di default $1\ sample/sec$ con quello desiderato. Questa è certamente una soluzione più comoda rispetto all'utilizzo di un blocco Simulink del tipo *rate transmission* per passare da un sample time all'altro, nel caso i tempi di campionamento fossero rimasti diversi. Sono stati inoltre eliminati i comandi per generare una rete che effettui predizione ad un passo, poco utile per la presente applicazione.

Sono state apportate altre modifiche allo script, al fine di ottenere i blocchi Simulink della rete sia in anello aperto sia in anello chiuso (tramite la funzione *gensim*) in open-loop e closed-loop, che corrispondono rispettivamente alle due differenti architetture serie-parallelo e parallelo già precedentemente discusse nel capitolo 5. Oltre alla modifica per plottare i blocchi Simulink, si aggiunge il comando *sim* che sarà utile per simulare il comportamento della rete a fronte di ingressi non utilizzati per il suo addestramento. I blocchi Simulink contenenti la

mask della rete non sono altro che l'equivalente Simulink del comando *sim*, perciò restituiscono in uscita gli stessi risultati della funzione Matlab.

Entrambi i blocchi Simulink hanno il compito di emulare il comportamento della rete, e in particolare la rete in anello aperto è adatta a problemi di predizione, mentre quella in anello chiuso a problemi di simulazione. Questa distinzione è stata già affrontata all'inizio di questo capitolo ma solo in questo frangente si scorge un riscontro pratico. Poiché lo schema in anello chiuso è un modello che dato il riferimento restituisce un'uscita simulata, solitamente si ricorre a questo se non è disponibile una qualche misura dei target [1], e in mancanza di essi è proprio l'uscita simulata che viene retroazionata dentro alla rete. In questo caso i target sono presenti, ma occorre capire se può risultare comodo sostituire completamente il modello a partire dal quale si sta addestrando la rete. Questo schema però è caratterizzato da performance generalmente inferiori rispetto alla stessa rete in anello aperto, che sono accettabili solamente quando le performance in anello aperto sono ottime, e cioè quando si verifica la condizione $\vec{\vartheta}_{NN} \approx \vec{\vartheta}$. Dato che non è possibile dare per scontato il poter impiegare una rete in anello chiuso ottenendo un'uscita di qualità [1], l'idea è quella di cominciare a descrivere il sistema tramite un modello in anello aperto, decidendo solo in seguito che approccio impiegare. Di seguito si riportano le figure dei blocchi:

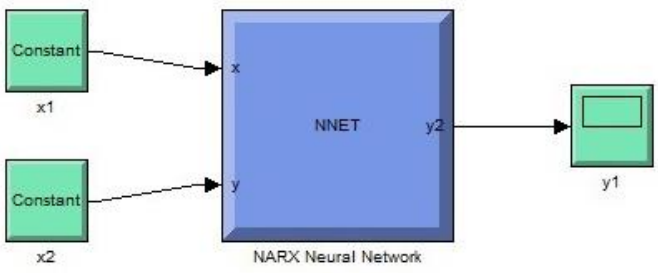

Figura 4.4 Blocco Simulink in predizione

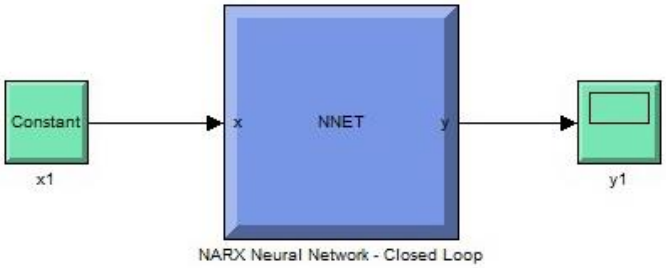

Figura 4.5 Blocco Simulink in simulazione

Come si nota dalle figure, in origine lo schema Simulink ottenuto da *gensim* richiede in ingresso dei valori costanti (un vettore campione di un istante di tempo), perché in ingresso prende un valore alla volta. Utilizzando in prova una coppia di vettori costanti, la stima della rete è un'uscita che si assesta dopo n istanti di tempo, e questo risultato deve essere confrontato con quello dei target effettivi.

Per una migliore interpretazione dei risultati, e per una più rapida analisi, si possono scegliere diverse strade. La prima è già stata discussa, ed equivale ad effettuare la prova online con la co-simulazione del robot. Se però si hanno già a disposizione i dati da precedenti co-simulazioni, si sceglie di dare per ogni input (e target) non il solo campione temporale ma l'intera storia temporale. Procedere in questo modo è equivalente a collegare la rete alla co-simulazione, in quanto si ottengono gli stessi risultati, ma con il vantaggio di impiegare un tempo più ridotto, vantaggio che si presenta solo perché manca la componente lenta della co-simulazione. Lo schema online risultante è riportato in Figura 4.6.

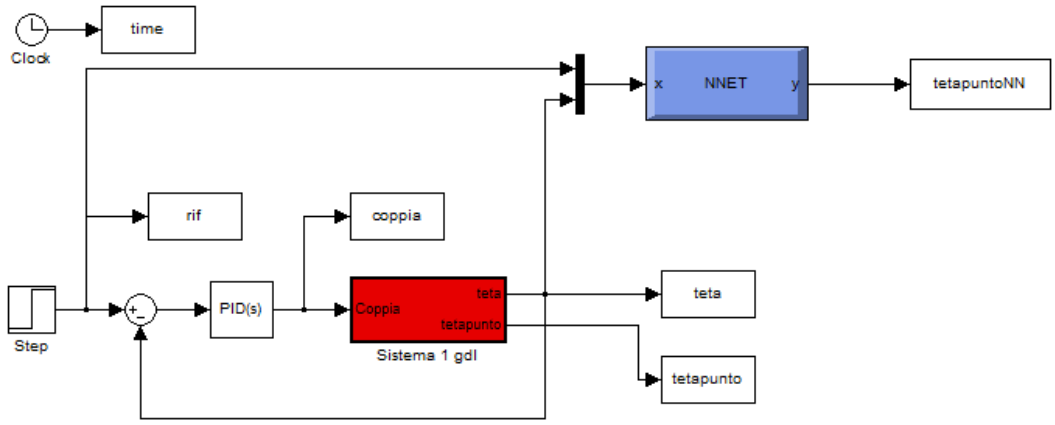

Figura 4.6. Sistema a 1gdl collegato alla rete neurale in simulazione

Un ultimo modo per simulare l'uscita della rete è quello più interessante, perché è stato ideato e sviluppato con lo scopo di sfruttare la relazione input-output della rete.

4.4 Script: equazione input-output della rete

Le risposte della rete presentate in tutto il lavoro di tesi non sono state ottenute grazie ai metodi di simulazione della NN finora presentati, ma sono infatti frutto di uno script creato appositamente per simulare la rete a fronte di numerosi riferimenti. Questo script genera l'uscita non lineare del modello a partire da un precedente addestramento effettuato impiegando *AdvancedScript.m* e alla base di esso è presente l'idea di sfruttare la relazione tra ingressi e uscite in funzione dei pesi e dei bias, che si può scrivere così:

$$\hat{\vartheta}_{NN} = purelin(b_2 + dotprod\left(LW, tansig\left(b_1 + dotprod\left(IW1, \begin{matrix} rit1 \\ rit2 \end{matrix}\right) + dotprod\left(IW2, \begin{matrix} tar1 \\ tar2 \end{matrix}\right)\right)\right)), \quad (4.1)$$

$$\text{con } rit1 = \begin{Bmatrix} \vartheta^{\circ}{}_1(t-1) \\ \vartheta^{\circ}{}_2(t-1) \\ \vartheta^{\circ}{}_3(t-1) \\ \vartheta_1(t-1) \\ \vartheta_2(t-1) \\ \vartheta_3(t-1) \end{Bmatrix} \text{ , } rit2 = \begin{Bmatrix} \vartheta^{\circ}{}_1(t-2) \\ \vartheta^{\circ}{}_2(t-2) \\ \vartheta^{\circ}{}_3(t-2) \\ \vartheta_1(t-2) \\ \vartheta_2(t-2) \\ \vartheta_3(t-2) \end{Bmatrix},$$

$$\text{e } tar1 = \begin{Bmatrix} \dot{\vartheta}_1(t-1) \\ \dot{\vartheta}_2(t-1) \\ \dot{\vartheta}_3(t-1) \end{Bmatrix}, tar2 = \begin{Bmatrix} \dot{\vartheta}_1(t-2) \\ \dot{\vartheta}_2(t-2) \\ \dot{\vartheta}_3(t-2) \end{Bmatrix},$$

in cui LW è la matrice dei pesi del secondo strato (quello di uscita), $IW1$ la matrice dei pesi che moltiplica i riferimenti ritardati e $IW2$ quella relativa ai target ritardati. In particolare, questa espressione è relativa al modello non lineare in forma predittiva, ma il modello in forma ingresso uscita ha espressione analoga a differenza dei termini $tar1$ e $tar2$ che non contengono più i target ai passi precedenti ma le $\hat{\vartheta}_{NN}$, sempre ritardate.

Si riporta di seguito un grafico che mostra il confronto tra l'uscita della rete impiegando il blocco Simulink in anello chiuso e il risultato dello script da me ideato. Il riferimento dato in ingresso è quello utilizzato in addestramento. Questo risultato rafforza l'idea di impiegare lo script sviluppato, sostituendolo completamente a quello generato da Matlab.

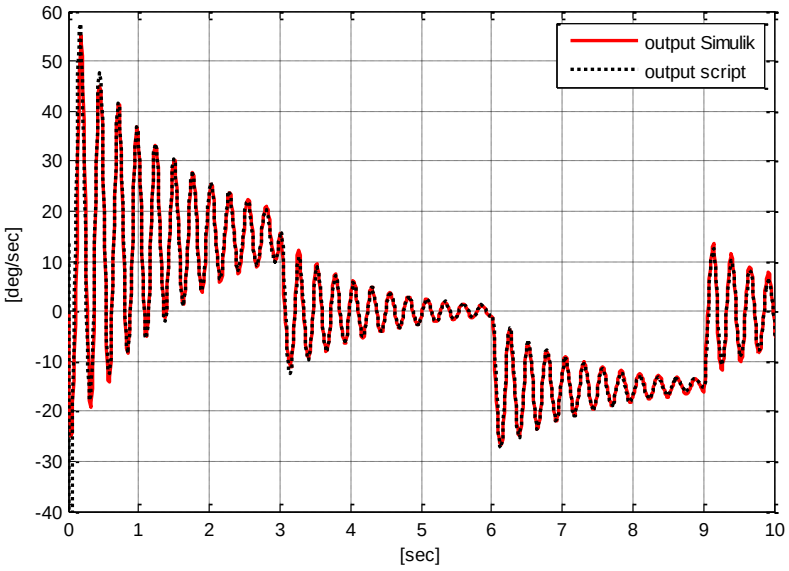

Figura 4.7 Confronto tra le uscite del blocco Simulink e dello script implementato.

4.5 Modello a 1gdl

Come primo approccio alle reti neurali viene progettato un sistema meccatronico non lineare ad un grado di libertà in ambiente Simulink. Esso rappresenta il *sistema reale* da cui collezionare i dati per la creazione del modello a rete neurale. Partire da questo sistema più semplice può essere utile per capire alcuni aspetti, quali l'ordine del modello da identificare, e evidenziare possibili problematiche nell'impiego delle reti, quali la scelta del riferimento per l'addestramento.

Il sistema meccatronico è rappresentato in Figura 4.8.

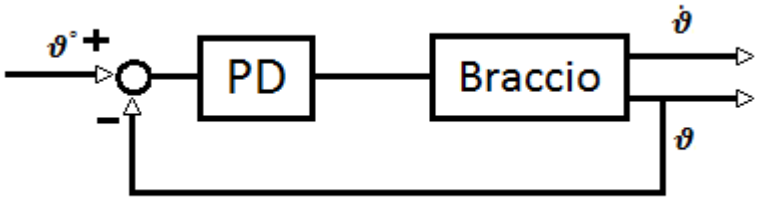

Figura 4.8 Schema a blocchi per lo studio del minimo ordine del modello a 1gdl

In questo caso, il problema si riduce ad un sistema SISO, con in ingresso il riferimento ϑ° e in uscita la velocità angolare del giunto $\dot{\vartheta}$. Esprimendo la velocità come un modello NARX sarà ovviamente scritta nel seguente modo:

$$\hat{\dot{\vartheta}}(t) = f(\dot{\vartheta}(t-1), \dots, \dot{\vartheta}(t-m), \vartheta^{\circ}(t-1), \dots, \vartheta^{\circ}(t-m)) \qquad (4.2)$$

L'idea è quella di passare a tempo continuo e calcolare la funzione d'anello $L(s)$ del sistema meccatronico PD + robot osservando quanti sono i poli della suddetta funzione, e da questa informazione risalire all'ordine minimo del modello NARX che descrive il sistema.

Le funzioni di trasferimento sono le seguenti:

$$\frac{\vartheta(s)}{C(s)} = \frac{1}{ms^2+rs+k} = \frac{1}{(s+p_1)(s+p_2)}, \frac{C(s)}{\vartheta^{\circ}(s)} = g\frac{(s+z_0)(s+z_1)}{s\,(s+p_{01})}, \qquad (4.3)$$

da cui:

$$L(s) = \frac{gs(s+z_0)(s+z_1)}{s(s+p_0)(s+p_1)(s+p_2)}, \qquad (4.4)$$

dove p_1 e p_2 sono complessi coniugati ed esprimono il grado di libertà del sistema, mentre p_0e 0 sono i poli corrispondenti del PD scritto nella seguente forma:

$$P + {}^{I}/_{s} + \frac{D\,N}{1+{}^{N}/_{s}} \qquad (4.5)$$

con P coefficiente proporzionale, I integrale, D derivativo e N coefficiente di filtraggio.

Occorre ora chiudere l'anello scrivendo la funzione di trasferimento $\frac{\dot{\vartheta}(s)}{\vartheta^{\circ}(s)}$. I poli in anello chiuso sono sempre quattro, ciò significa che il denominatore della funzione in anello chiuso è di quarto grado. Si potrebbe affermare quindi che sia necessario orientarsi su un modello a rete NARX di ordine quattro per l'identificazione, in quanto i ritardi $(t-i)$ sono da associarsi ai termini z^{-i} della trasformata Zeta, e ne determinano l'ordine ed ugualmente il numero di poli. Occorre però considerare che i poli utili da identificare sono la coppia di complessi coniugati, che dà l'informazione sulla frequenza propria del sistema. I poli del PD possono essere infatti trovati senza difficoltà nei problemi reali, quindi non ci si preoccuperà di questi, facendo ricadere la scelta su un modello NARX di ordine 2. Questa scelta sarà esposta in modo più dettagliato nella trattazione del sistema MIMO nel capitolo 8.

4.5.1 Scelta dei dati di addestramento

Di seguito si riporta cosa accade quando erroneamente si prende un set di dati di addestramento non idoneo all'obiettivo che ci si è prefissati.

Si prenda come esempio l'andamento di Figura 4.9. Il robot compie una manovra da 0° a 45° , resta in posizione e poi l'angolo comincia a decrescere. Non collezionando anche l'andamento decrescente completo del braccio robotico, ci si deve aspettare che, una volta addestrata la rete, questo modello non possa emulare manovre nell'intorno dei dati che mancano.

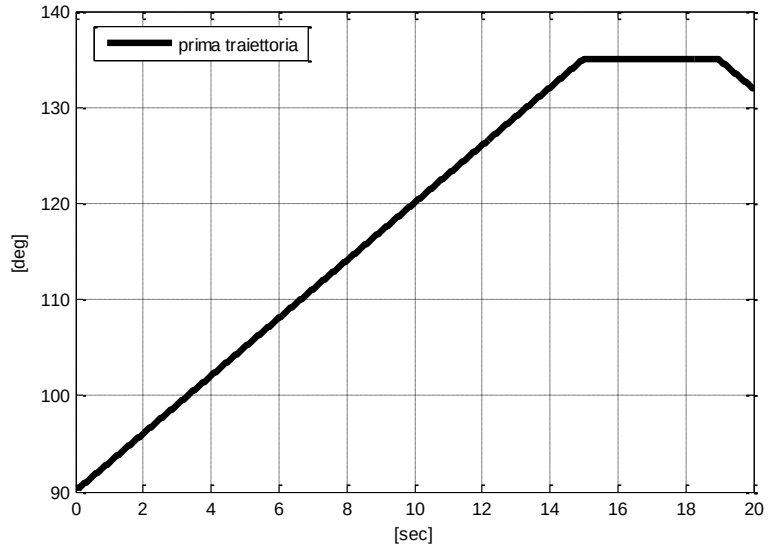

Figura 4.9 Prima traiettoria di addestramento

In Figura 4.10 si riporta proprio l'andamento dell'uscita del modello e l'andamento dell'uscita reale quando in ingresso è dato un riferimento esterno ai dati di addestramento di mediocre qualità.

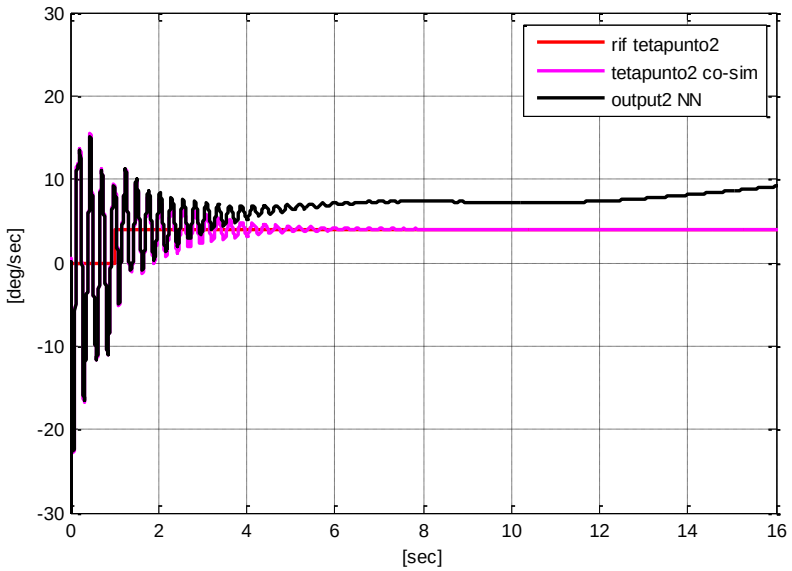

Figura 4.10 Risposta della rete neurale con la prima traiettoria di addestramento

Come tipo di addestramento si opta per quello riportato in Figura 4.11, che porta il braccio robotico dalla posizione a 0° a quella a 45°.

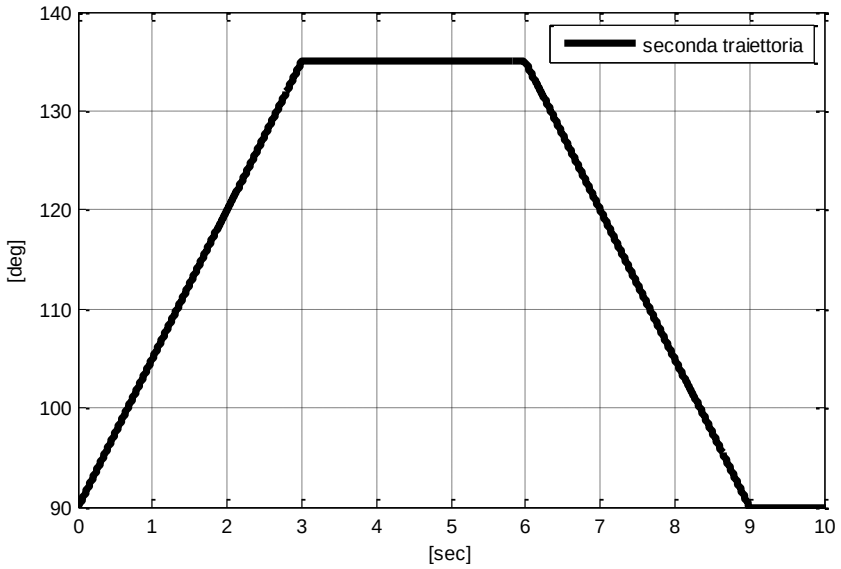

Figura 4.11 Seconda traiettoria di addestramento

Si sceglie una frequenza di campionamento pari a 100 Hz, perché dall'analisi della risposta si nota come sia presente una componente a circa 3Hz: non occorre quindi salire troppo con l'ampiezza della banda di frequenze per evitare il fenomeno di overfitting sui dati in fase di addestramento.

Si fissa l'ordine a due, ma si varia il numero di neuroni seguendo un metodo a complessità crescente partendo da un solo neurone. L'errore tra l'uscita della rete e l'uscita desiderata è l'indice di paragone per effettuare il passaggio da una rete con un'architettura semplice ad un singolo neurone, ad una più complessa avente più neuroni. Come prima analisi si concentra l'attenzione sull'uscita predetta del modello in configurazione serie parallelo, ma è fondamentale osservare anche il modello parallelo (dato che l'obiettivo finale è ottenere un modello ingresso-uscita e tendenzialmente offre performance peggiori dell'anello aperto). Dopo aver generato diversi modelli a parità di ordine, si

riporta l'andamento della storia temporale del modello NARX(2,2) scelto per il sistema, descritto da una rete a 3 neuroni, in configurazione *anello chiuso*.

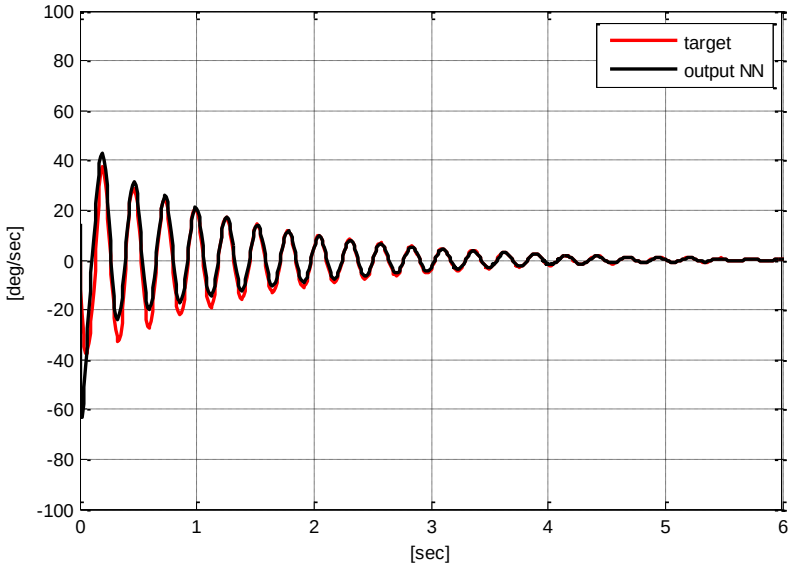

Figura 4.12 Confronto tra uscite desiderata e del modello non lineare

Una nota ulteriore che si può presentare riguarda il comportamento generale della rete. Dato che si sta impiegando una rete NARX alimentata da ingressi affetti da ritardo, si noti come all'istante di tempo *zero* e ai due passi successivi la rete presenti un'uscita molto diversa dal target. Ciò è conseguenza del fatto che per i primi due istanti si forniscono alla rete dei dati "fittizi" inizializzati a zero, quindi l'uscita si discosta molto da quella desiderata. La stima di $\dot{\vartheta}$ viene così falsata finché al passo 5 non possiede tutti i dati per una più corretta simulazione. Si riporta il dettaglio in Figura 4.13.

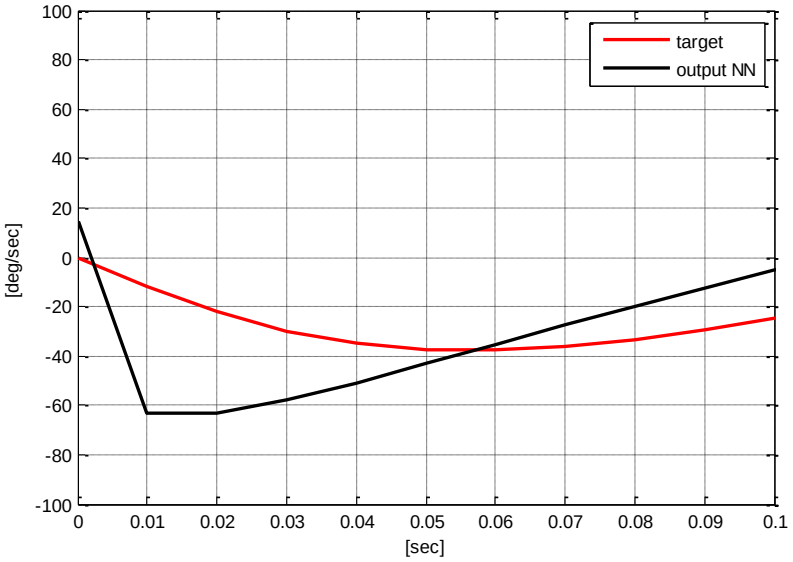

Figura 4.13 Dettaglio del confronto tra uscite desiderata e del modello non lineare

Si può affermare che il modello in anello chiuso sia valido se valutato in questa configurazione del braccio a 1 gdl, sia per la frequenza che per lo smorzamento dell'oscillazione, e questo perché difatti nei dati di addestramento è contenuta l'informazione relativa a questa configurazione.

4.5.2 Linearizzazione e forma di stato

Una volta trovato il modello non lineare del sistema, è comodo effettuare la linearizzazione per portarlo in forma di stato. Questo passaggio è fondamentale per poter implementare l'anello di controllo di vibrazioni.

In letteratura esistono numerosi metodi per linearizzare l'uscita di una rete neurale. Questa operazione si riconduce alla linearizzazione dell'unica funzione causa della non linearità: il sigmoide. In questa trattazione si è pensato di espandere l'espressione secondo il polinomio di Taylor, invece di impiegare

tecniche innovative. Esistono infatti metodologie che approssimano il sigmoide e la sua derivata fornendo una versione quantizzata in N passi [15]. In Figura 4.14 è riportato l'andamento della risposta linearizzata, sovrapposta agli andamenti non lineare e desiderato.

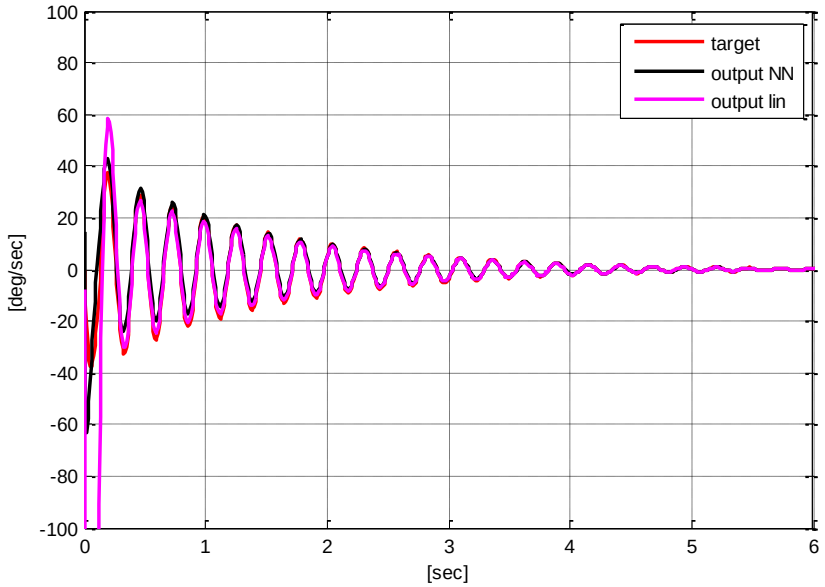

Figura 4.14 Confronto tra uscite desiderata, del modello non lineare e del modello linearizzato

La coppia di autovalori in uscita dal processo di identificazione, in seguito alla linearizzazione, è la seguente:

$$\begin{cases} -14.1790 + 19.5219\mathrm{i} \\ -14.1790 - 19.5219\mathrm{i} \end{cases}$$

che corrisponde ad una frequenza propria identificata di 3.84 Hz. Si noti che il valore della frequenza identificata è molto simile a quello mostrato dal grafico della Figura 4.14. Si può quindi affermare che il modello abbia identificato correttamente il sistema.

4.5.3 Robustezza modello

Per valutare la bontà del modello si sono forniti al modello alcuni dati corrispondenti ad una delle configurazioni che il braccio può assumere, e si sono riportati risultati ancora positivi. In Figura 4.15 è presentato un andamento pensato da far replicare al modello, la cui espressione di ingresso uscita viene linearizzata attorno alla configurazione di equilibrio a 45°. In Figura 4.16 si riporta l'andamento della $\dot{\vartheta}$, riscontrando chiaramente la bontà dei modello non lineare nell'intorno della condizione di equilibrio. Per verificare il fitting del modello linearizzato occorre superare infatti un transitorio iniziale.

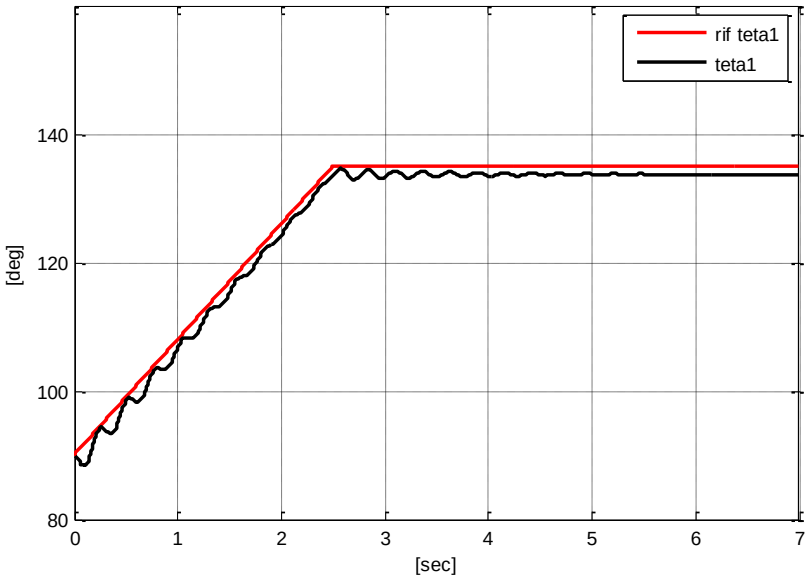

Figura 4.15 Primo test di robustezza

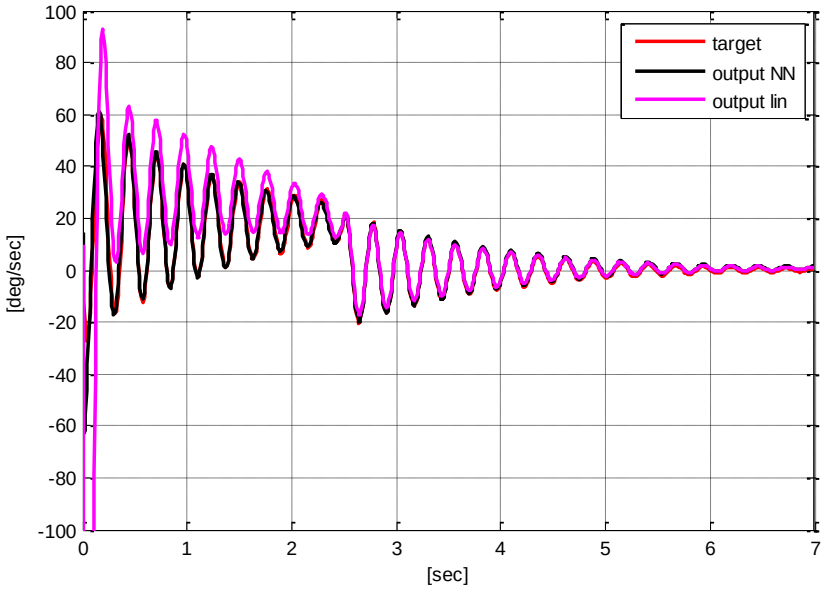

Figura 4.16 Confronto uscite desiderata e del modello linearizzato per il test di robustezza

4.6 Controllo

Per affrontare il tema di controllo delle vibrazioni, occorre pensare ad un controllo da applicare al sistema meccanico che possa essere di tipo *model based*, cioè che generi un'azione di controllo sulla base del modello identificato. Ad esempio, si può pensare ad una strategia tipo LQR che usi le matrici A e B di una forma di stato differente da quella trovata prima [22]: una forma di stato modale. Ipotizzando di scegliere come stati le coordinate modali $\dot{q}$ e q, la matrice A risulta costruita nel seguente modo:

$$A^* = \begin{bmatrix} -\zeta\omega & \omega^2 \\ 1 & 0 \end{bmatrix}, \qquad (4.6)$$

con ω la frequenza propria in $\frac{rad}{s}$ e ζ lo smorzamento associato, calcolati grazie agli autovalori derivanti dall'identificazione.

Dato che lo stato non è noto, si prevede l'impiego di un osservatore di stato. All'interno dell'osservatore entra la ϑ uscente dal modello e la $\dot{\vartheta}$ stimata dalla rete neurale, volendo simulare il caso in cui non sia disponibile durante il controllo online la misura della tetapunto reale. La struttura complessiva di *controllo + osservatore + sistema* è riportata in Figura 4.17.

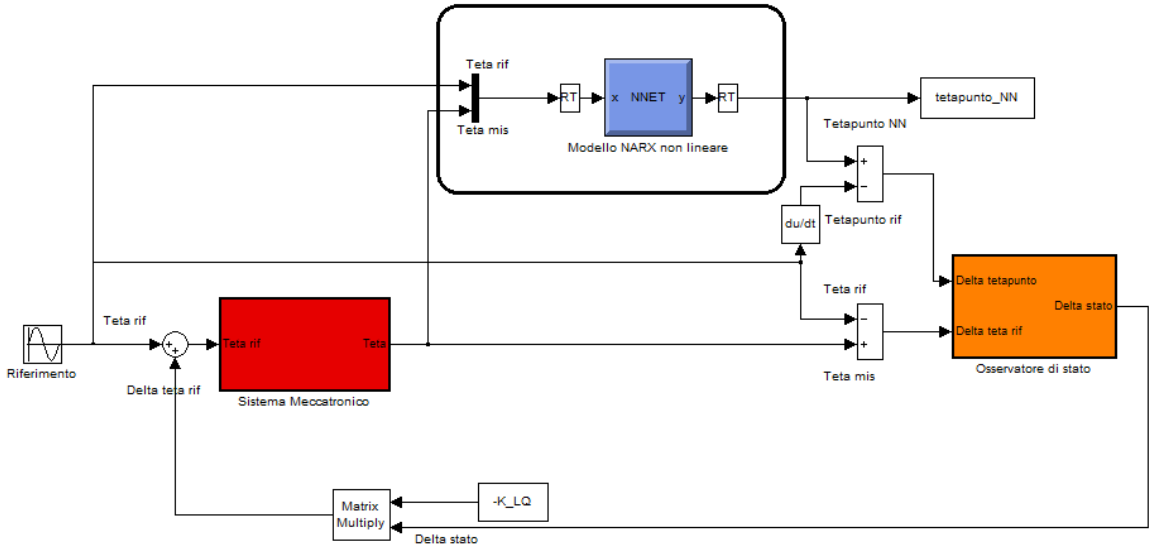

Figura 4.17 Schema di controllo di vibrazioni per il caso di studio a 1 gdl

Il risultato complessivo del controllo è soddisfacente ed è stato analizzato il caso in cui sia presente un disturbo random di coppia, come l'azione del vento sul link. La teta del sistema presenta i seguenti andamenti, in risposta ad un riferimento sinusoidale e al variare dei valori delle matrici Q e R:

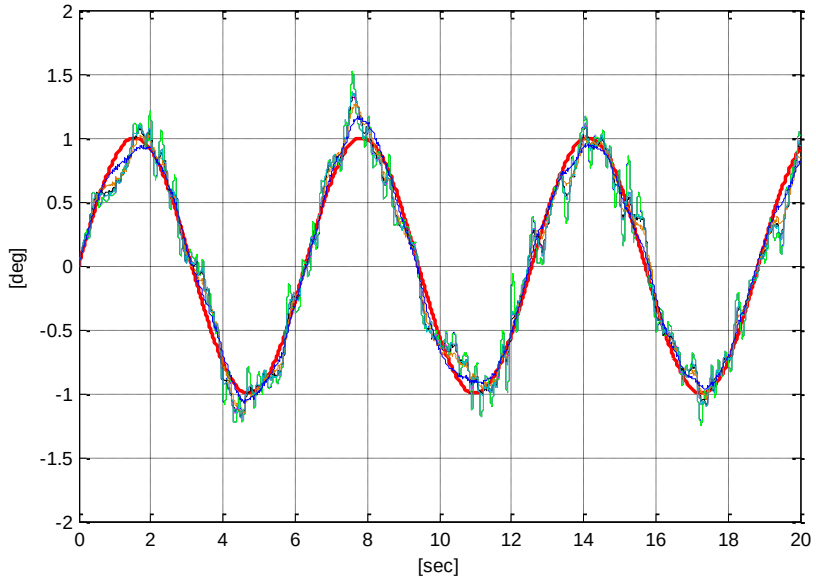

Figura 4.18 Studio scelta delle matrici Q e R

La miglior risposta controllata è quella corrispondente a Q = 1000 e R = 0.1, riportata in figura sotto.

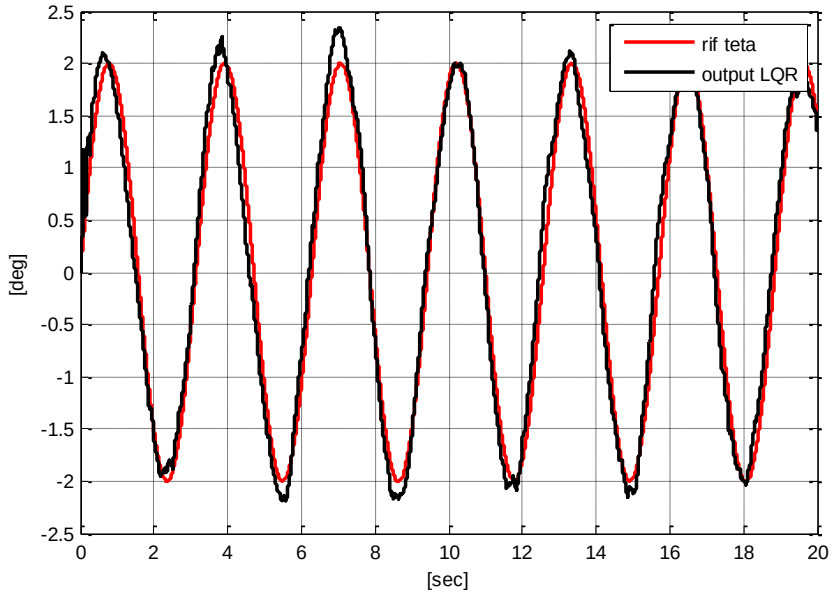

Figura 4.3 Confronto uscite desiderata e del controllo LQR

5. Il manipolatore a 3gdl

5.1 Il banco prova

Il banco prova è costituito da un manipolatore planare a tre link flessibili, progettato e realizzato da un collega di ingegneria meccanica in quanto argomento del suo lavoro di tesi. I bracci del robot sono realizzati in carbonio, e le rotazioni relative sono consentite da tre cerniere rigide in alluminio pieno. Ognuna di esse trasmette la coppia motrice uscente dal singolo motore direttamente al link corrispondente, grazie ad un collegamento a chiavetta. I motori sono tre, così come riduttori e inverter. In Figura 5.1 è mostrato il banco prova ultimato.

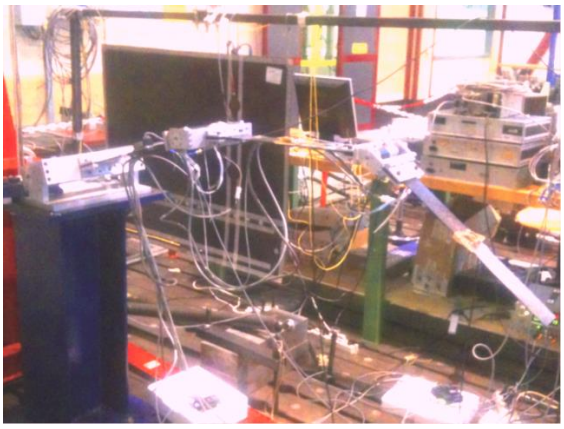

Figura 5.1 Fotografia del banco prova

Nel dettaglio, i bracci in carbonio sono stati realizzati all'interno dell'edificio della galleria del vento del Politecnico di Milano, tramite sovrapposizione multipla di sottili strati di carbonio, ottenuti a partire da un coil iniziale. Hanno medesima lunghezza ma differenti spessore e larghezza. I dati sono riportati di seguito:

	Lunghezza [mm]	Larghezza [mm]	Spessore [mm]	E [Mpa]	ν [-]
Link1	500	60	10	100000	0.33
Link2	500	75	7	100000	0.33
Link3	500	85	3	100000	0.33

nelle ultime due colonne si riportano le proprietà elastiche dei link in carbonio. Si è deciso di impiegare il braccio in carbonio con lo spessore maggiore per il primo link collegato alla base del manipolatore (vincolata a terra), in quanto deve portare il carico del secondo link e dell'ultimo oltre che i pesi delle cerniere in alluminio pieno e degli attuatori. Durante l'assemblaggio dei vari strati di carbonio si è deciso di utilizzare fibre longitudinali per l'anima di ogni link, ognuna di esse posta tra due strati di fibre trasversali (per conferire maggiore rigidezza) e i restanti strati di fibre longitudinali. Dopo l'assemblaggio, i bracci sono stati portati nel laboratorio del dipartimento di Ingegneria Aerospaziale per effettuare la laminazione, processo della durata di due ore per ciascun link, alla temperatura di $120\ °C$ e alla pressione superficiale di $1\ bar$. Dopo tale processo, la lavorazione dei bracci è stata ultimata nell'edificio della galleria del vento, effettuando un taglio tramite fresa con punta di diamante in presenza di grosse sbavature o pieghe laterali derivanti dal processo di laminazione.

Per la movimentazione della struttura sono stati acquistati tre motori DC brushless Maxon, in particolare:

- ✓ Prima cerniera: Ec-max 30, 40W , riduttore: GP-32C con r = 295:1
- ✓ Seconda cerniera: Ec-max 40, 120W, riduttore: GP-52C con r = 285:1
- ✓ Terza cerniera: Ec-max 60, 400W, riduttore: GP-81 con r = 308:1

Gli inverter hanno ingresso analogico ± 10 V: due Epos2 $50/5$ per gli ultimi due link e un Epos2 $70/10$ per il primo vincolato a terra. Per le misure di posizione angolare sono stati impiegati tre inclinometri e per implementare le logiche di controllo è stata messa a disposizione una centralina dSPACE D1006.

In questo lavoro di tesi si è però impiegata una co-simulazione di un modello CAD del manipolatore, effettuato con il software Adams, che ha come ingressi le coppie fornite ai motori, mentre come uscite gli angoli relativi e le velocità angolari di ciascun braccio robotico. Esso viene esportato in ambiente Simulink per l'applicazione del controllo di posizione tramite tre PD in linea d'andata. In ingresso al blocco Adams si è infatti aggiunto uno schema di controllo, il quale riporta come ingressi le posizioni angolari di riferimento (che da adesso in poi saranno chiamate simbolicamente ϑ°) e come uscite le coppie ai motori.

La co-simulazione rappresenta effettivamente il sistema *reale* da cui collezionare i dati per la costruzione del modello del sistema meccatronico simulato. Di seguito si riporta un'immagine del modello multibody.

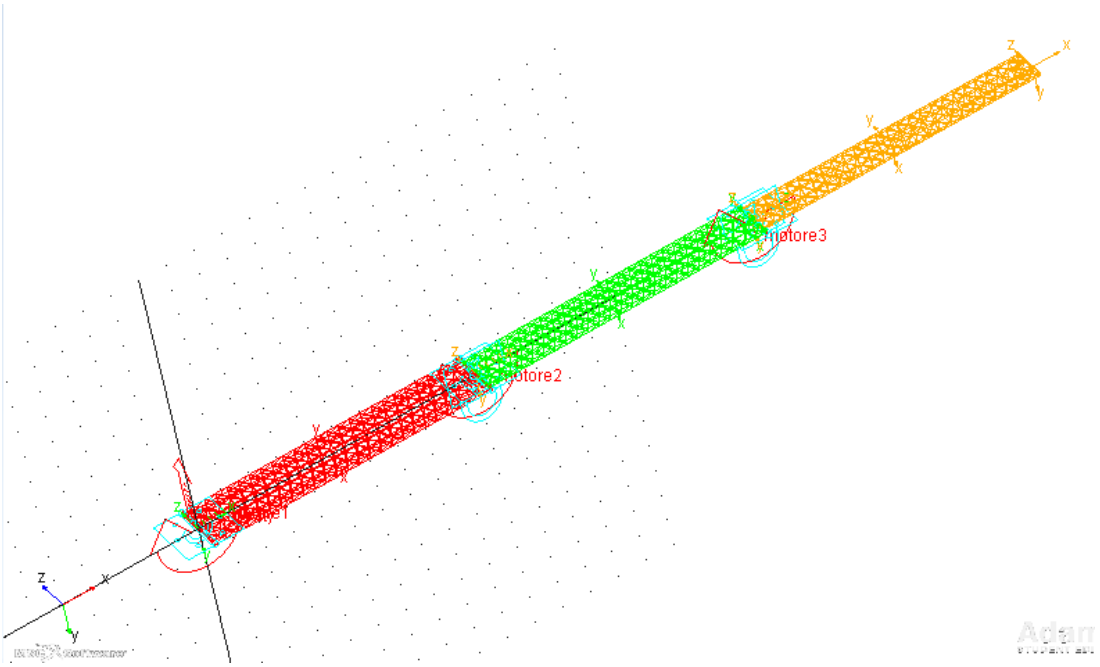

Figura 5.2 Modello multibody del braccio meccanico a 3gdl

5.2 Il modello numerico (ADAMS + Simulink)

È sulla base dei dati ottenuti da questa co-simulazione che viene creata la rete neurale artificiale. La rete che si vuole ottenere rappresenta il modello del sistema meccatronico complessivo, comprensivo del modello fisico del robot e del sistema di controllo del moto. Tutti e tre i PD scelti presentano la seguente espressione:

$$P + \frac{DN}{1+{}^{S}/_{N}}, \qquad (5.1)$$

in cui P è il coefficiente proporzionale, D quello derivativo e N un fattore di filtraggio. Questi parametri non si sono potuti scegliere seguendo le procedure classiche che prevedono una conoscenza aprioristica del modello del robot, per poter effettuare un'indagine sul diagramma di Bode o sul luogo delle radici. In

questo ambito la scelta è stata effettuata seguendo alcuni ragionamenti che vengono presentati in seguito. Come prima cosa si è effettuata una stima dell'ordine di grandezza che avrebbero avuto le coppie motore in funzione di una determinata configurazione del robot (ad esempio la stima delle coppie statiche in funzione di diverse configurazioni di equilibrio). Questa è solo una approssimazione, e infatti le migliorie sono state apportate in seguito all'analisi della risposta simulata del robot. Fissando poi un valore per il coefficiente di filtraggio dei PD, si è modificato il valore assunto dal parametro derivativo (è importante che assuma un valore modesto per evitare problemi legati a eccessive dinamiche [16]). Per ciò che concerne il parametro integrale, non è stato necessario inserirlo, in quanto la presenza di un'azione integrale non ha apportato migliorie alla risposta controllata. Una volta affinati i parametri dei PD e quindi ultimato il progetto del controllo di posizione, si riporta in figura il sottosistema corrispondente:

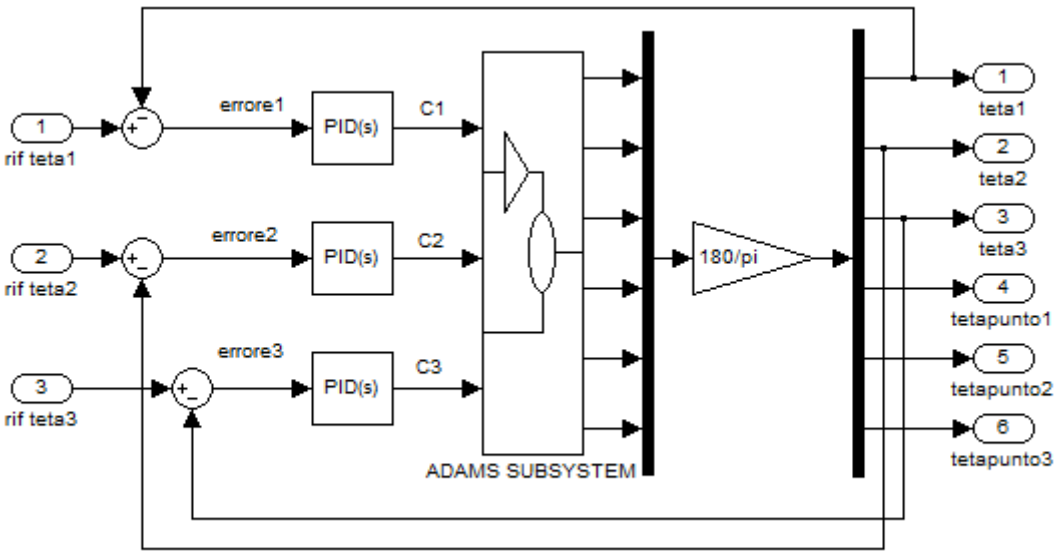

Figura 5.3 Schema del modello multibody importato in Simulink e del controllo PD

Lo schema esterno ha quindi come ingressi i riferimenti angolari ϑ e come uscite le velocità angolari $\dot{\vartheta}$. La rete neurale dovrà emulare il comportamento del sistema meccatronico, e per farlo avrà come input le ϑ e come target le $\dot{\vartheta}$. Come primo obiettivo si vuole giungere ad una identificazione solo parziale di questo modello esteso: si è infatti scelto di dare lo stesso riferimento agli ultimi due link, per fare in modo che il terzo angolo relativo sia nullo. In questo modo si andrà incontro ad un modello che descriverà un sistema meccatronico comprensivo di un robot dotato di almeno tre gradi di libertà, che corrispondono ai tre bracci rigidi. In figura si mostra lo schema meccatronico tramite il blocco centrale:

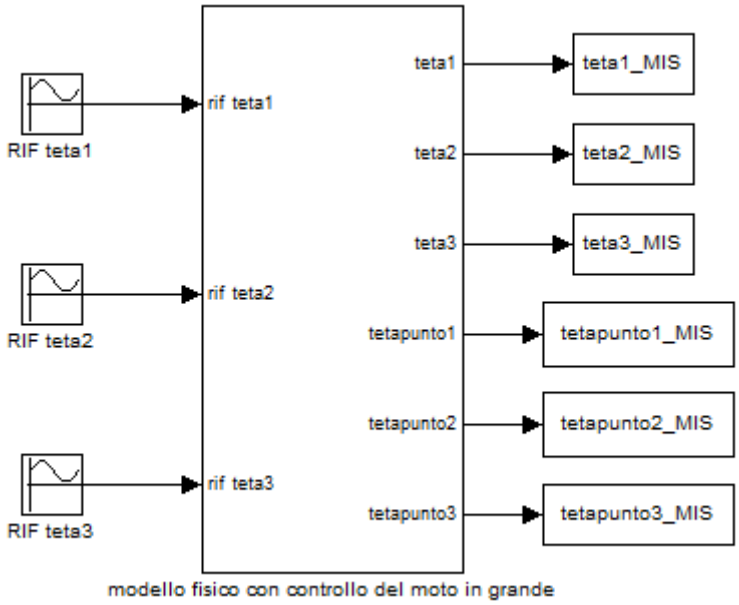

Figura 5.4 Schema Simulink esterno

Occorre passare ora alla creazione della rete neurale. Come affermato nel capitolo precedente, è fondamentale scegliere il tipo di riferimento da dare alla rete per effettuare l'addestramento, e per farlo la prima cosa da decidere è se fare muovere il robot completamente all'interno del suo spazio di lavoro. Con il

termine *spazio di lavoro* si intende quella regione dello spazio descritta dall'origine della terna utensile quando si fanno svolgere ai giunti tutti i moti possibili [17]. Esso è funzione del tipo di geometria del robot e influenzato dai fine corsa dei giunti. In questo caso si è deciso di ridurre lo spazio di lavoro del robot all'intervallo di posizioni angolari tra 0° e 45°[3] per ognuno dei giunti. Questa scelta impone che le traiettorie di addestramento che verranno fatte eseguire prima alla co-simulazione, poi al robot non andranno mai oltre questo range di valori.

5.3 Scelta e ordine del modello

Occorre quindi estendere al caso di studio a 3 gdl le considerazioni già esposte nel capitolo quattro. Una volta imposta la struttura NARX, occorre fissare un possibile intervallo di ordini del modello, come già fatto nel caso del sistema a 1 gdl. La scelta dell'ordine del modello equivale a scegliere quanti siano i segnali di input da dare alla funzione non lineare del NARX, corrisponde cioè ad un problema di selezione degli ingressi. In effetti in una rete NARX gli ingressi (comprendendo sia input che target) raggruppano stessi segnali presi a passi differenti, ad esempio:

$$\vartheta(t-i) \quad \text{e} \quad \vartheta(t-(i+1)), \qquad (5.2)$$

con i un numero intero generico. Essi possono essere considerati input differenti dal punto di vista formale, però sono strettamente correlati, essendo uno il ritardo dell'altro. È questo legame che rende la scelta dell'ordine difficile per problemi di questo tipo, rispetto a problemi in cui gli ingressi sono distinti e meno correlati, se non addirittura senza correlazione. Questo ragionamento vale

[3] Nei grafici il range di valori è tra 90° e 135° in quanto in Adams la retta di riferimento è quella verticale.

non solo per i modelli NARX ma per tutti i modelli a dinamica esterna, per la cui definizione si rimanda al capitolo 2.

La problematica riguardante la determinazione dell'ordine per sistemi non lineari è ancora insoddisfacente, in quanto non esiste un metodo univoco e semplice per giungere ad un risultato. La pratica più comune è infatti una miscellanea di metodi, come il *trial and error* in combinazione alla conoscenza aprioristica del sistema, se disponibile [4].

Per la scelta della complessità del modello si è deciso di procedere per complessità crescente. Solo guardando i risultati si potrà decidere se complicare il modello aumentando entrambi i ritardi (ad esempio aumentandoli di un'unità) o se questo non porterà effettive migliorie. Questo approccio è però il solo *trial and error*.

Dato che procedere solamente per tentativi non è una scelta soddisfacente, si può affiancare un altro ragionamento, derivando le conclusioni per la scelta della complessità del modello a tre gradi di libertà dai risultati ottenuti a partire dal caso più semplice, già trattato nel Capitolo 4: il manipolatore a singolo giunto. In quel caso si avevano due soli poli da identificare (suoi 4 complessivi), e bastava un modello del secondo ordine. Ciò significa che se il sistema meccatronico contenente il sistema ad un solo grado di libertà avesse tre poli da identificare, servirebbe un modello del terzo ordine. Questa derivazione diretta vale solo per i sistemi SISO, non si può infatti affermare una diretta corrispondenza anche nel caso di sistemi MIMO, quindi si procede per derivazione. Dato che nel sistema a 3 gdl sono applicati tre controllori PD, vengono considerati solo i poli corrispondenti alle frequenze proprie del sistema. Si ipotizza infatti che sia semplice reperire le informazioni riguardanti i parametri dei PD. Nel caso a due gradi di libertà, sono presenti due bracci e due PD e quindi complessivamente si raddoppiano i poli del sistema ad anello

chiuso, che diventano sei. Due infatti sono dati dai PD mentre gli altri quattro sono le due coppie di poli complessi coniugati che descrivono i due gradi di libertà. Questo dà origine ad una generalizzazione che porta ad affermare che in anello chiuso si ottiene un numero di poli che può essere calcolato così:

$$n° \, poli\_anello\_chiuso \; = \; n° controllori\_PD \; + \; 2 * n° gdl. \qquad (5.3)$$

È necessario specificare che in questa sezione si stanno considerando i soli moti rigidi associati alla struttura, trascurando tutti i gradi di libertà (in teoria infiniti) che descrivono il sistema. Utilizzando quindi l'equazione, si scopre che per un modello del manipolatore a tre bracci che descriva i soli moti rigidi occorrono $3 + 2 * 3 \; = \; 9$ poli del denominatore della funzione $L(s)$.

Per risalire all'ordine che deve avere la rete neurale NARX in configurazione serie-parallelo, bisogna ricordarsi che questa configurazione corrisponde all'anello aperto. È scorretto quindi pensare che per ricondursi al modello minimo del robot occorra un NARX(9,9), infatti un modello di questo ordine descriverebbe il sistema tenendo conto di altri gradi di libertà che includerebbero sia i tre rigidi che altri moti vibratori associati a frequenze proprie del *moto in piccolo* (con *moto in piccolo* si intende il moto vibratorio della struttura, in contrapposizione con il *moto in grande* su cui si attua il controllo in posizione). Il passaggio da anello aperto ad anello chiuso è importante per quanto già affermato nei capitoli precedenti: conviene sempre addestrare una rete neurale, o in generale un qualsiasi modello a scatola nera, passando sempre prima per l'anello aperto ed eventualmente chiuderlo in un secondo momento. Il passaggio di chiusura dell'anello è fondamentale per arrivare ad una caratterizzazione in forma di stato.

Per effettuare i calcoli si fingerà di aver già compiuto un passo in avanti, che verrà ripreso nel prossimo capitolo, che è quello riguardante la linearizzazione.

Dato che in seguito alla linearizzazione un modello non cambia il proprio ordine, si effettua l'ipotesi comoda di lavorare su un modello ARX.

5.3.1 Ordine minimo di un modello a 2gdl

Come primo appunto occorre ricordare che il sistema meccanico ha due gradi di libertà, corrispondenti ai moti dei due giunti. Le variabili da controllare sono quindi due e sono perciò necessari due controllori PD. Il sistema non è più un SISO come nel caso precedente e il modello di ordine minimo risulta avere 6 poli.

Ciò significa che la funzione $G(s)$ ha questa forma:

$$G(s) = \frac{num(s)}{(s+p_1)(s+p_2)(s+p_3)(s+p_4)(s+p_5)(s+p_6)}, \qquad (5.4)$$

con $num(s)$ lasciato parametrico in quanto ancora non si conosce il grado ma di sicuro inferiore a quello del denominatore.

Con $d1$ e $d2$ si indicano i ritardi scelti per input e target della rete neurale. Dato che il sistema è MIMO allora, a valle del calcolo dei coefficienti della rete, il sistema avrà la seguente forma:

$$\begin{pmatrix} (s+p_{01})(s^{d1}+\cdots+a_1) & \dots \\ \dots & (s+p_{02})(s^{d2}+\cdots+a_2) \end{pmatrix} \vec{\vartheta} = \begin{pmatrix} \cdots & \cdots \\ \cdots & \cdots \end{pmatrix} \overrightarrow{\vartheta^{\circ}}, \qquad (5.5)$$

in cui però non si dà importanza ai termini *misti* sulla antidiagonale e i termini di grado inferiore. Gli elementi della matrice a destra dell'uguale sono dei polinomi generici aventi grado massimo pari al ritardo massimo applicato agli input e anch'essi non influenzano il risultato. I termini sulla diagonale invece sono fondamentali in quanto è proprio da questi che si conoscerà l'ordine del modello minimo. Entrambi sono formati dal prodotto tra il polo dato dal PD e

dal polinomio che ha in sé l'informazione sul numero scelto di passi precedenti. Per portare il modello in forma equivalente alla funzione di trasferimento in anello chiuso occorre effettuare questa operazione:

$$\vec{\vartheta} = \left(inv \begin{pmatrix} (s+p_{01})(s^{d1}+\cdots+a_1) & \cdots \\ \cdots & (s+p_{02})(s^{d2}+\cdots+a_2) \end{pmatrix} \right) \begin{pmatrix} \cdots & \cdots \\ \cdots & \cdots \end{pmatrix} \overrightarrow{\vartheta^{\circ}}, \tag{5.6}$$

in cui l'inversa ha il denominatore costituito dal determinante della matrice stessa. Questo denominatore presenta un termine di grado massimo è pari a:

$$s^m + \cdots + a_m = (s^{d1+1} + \cdots + a_1)(s^{d2+1} + \cdots + a_2), \tag{5.7}$$

dunque $m = (d1+1)+(d2+1)$. Effettivamente $d1 = d2$ perché si è scelto di applicare lo stesso ritardo a tutti i target. Dato che si ottiene un'equazione in un'incognita e $m = 6$, la soluzione è:

$$d = 2.$$

Il ritardo applicato ai target è parte dell'ordine del modello serie-parallelo utilizzato. Scegliendo per comodità lo stesso valore di ritardo per input e target, la rete neurale che rappresenta il modello minimo per un manipolatore planare a due bracci è il NARX(2,2). Scegliendo diversi valori per i due ritardi si troverebbero chiaramente diversi modelli che rappresentano quello minimo, ma questa eventualità non è stata esplorata in questa trattazione. È buona norma però osservare un accorgimento, che consiste nell'evitare di ottenere delle funzioni di trasferimento con un numeratore di grado maggiore rispetto al denominatore (che corrisponde alla scelta di andare indietro di più passi nel segnale di input rispetto a quello di target). In altre parole, è consigliabile mantenere sempre delle funzioni di trasferimento di tipo *causali* [16].

5.3.2 Ordine minimo del modello del manipolatore

Dopo aver effettuato il passaggio da caso SISO a caso MIMO 2x2, è immediato ricavare l'ordine del modello nell'estensione a tre gradi di libertà. Senza riportare tutti i passaggi esplicativi mostrati sopra, occorre estendere il problema ad una matrice 3x3 la cui inversa presenta un denominatore avente termine di grado massimo:

$$s^m + \cdots + a_m = (s^{d1+1} + \cdots + a_1)(s^{d2+1} + \cdots + a_2)(s^{d3+1} + \cdots + a_3), \tag{5.8}$$

e per cui:

$$m = (d1 + 1) + (d2 + 1) + (d3 + 1) = 9.$$

In questo caso $3d = 6$, quindi $d = 2$, e questa conclusione porta a cercare una struttura del modello serie parallelo di tipo NARX(2,2). Tutte le considerazioni inerenti al modello NARX(2,2) valgono anche in questo caso.

5.3.3 Modello non lineare vs modello lineare

Giusto per completezza, si riporta in Figura 5.5. un grafico che presenta la velocità angolare in uscita dal modello lineare del manipolatore, sempre una rete neurale artificiale NARX di ordine due. Come si nota facilmente, il modello lineare non permette di avere un'uscita che segua il target (velocità angolare in rosso, ottenuta in cosimulazione). Questo perché il modello non lineare è più completo e permette di ottenere un'uscita buona anche al di fuori dei dati di addestramento, cosa non vera per il modello lineare.

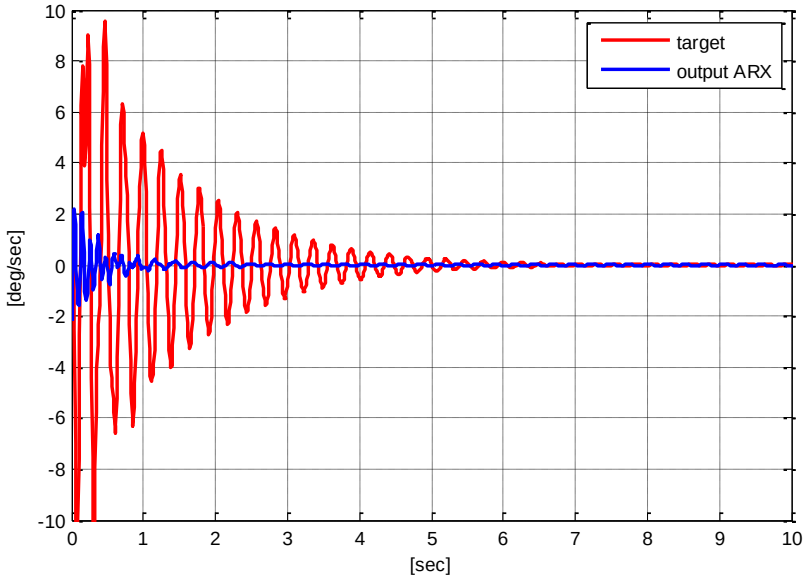

Figura 5.5 Confronto uscita desiderata e del modello ARX

5.4. Addestramento e robustezza

Come già dimostrato nel capitolo 4, occorre avere ben presente qual è l'obiettivo che ci si vuole prefiggere. In questo lavoro si vuole ottenere un modello che possa descrivere bene il comportamento del sistema meccatronico a 3 gdl all'interno di un determinato range di configurazioni, è quindi interessante capire come debba essere costruito il riferimento per addestrare la rete. Esso potrebbe infatti non essere lo stesso usato nel caso del sistema a 1gdl.

Per evitare l'errore commesso con il modello a 1 gdl, risulta chiaro come sia necessario effettuare l'addestramento di una rete basato su un riferimento che contenga più informazioni di quello usato proprio in Figura 4.9 [14], e che contenga i limiti dello spazio di lavoro scelto per il manipolatore. Con l'espressione *contenere più informazione* si vuole indicare un riferimento di ϑ_1, ϑ_2 e ϑ_3 che permetta di valutare più configurazioni possibili. Agendo in questa

direzione, a prescindere dal tipo di riferimento dato in ingresso alla rete, essa dovrebbe garantire un'uscita molto prossima ai target.

Rispetto al caso di studio a 1 gdl conviene mostrare non più solo i riferimenti ma le traiettorie nei piani ϑ_1, ϑ_2 e ϑ_1, ϑ_3, per comprenderne meglio l'andamento. Si mostrano nelle Figure 5.6, 5.7, 5.8, 5.9 le manovre che sono state confrontate per la scelta del tipo di dati di addestramento da fornire alla rete neurale (gli angoli sono espressi in gradi).

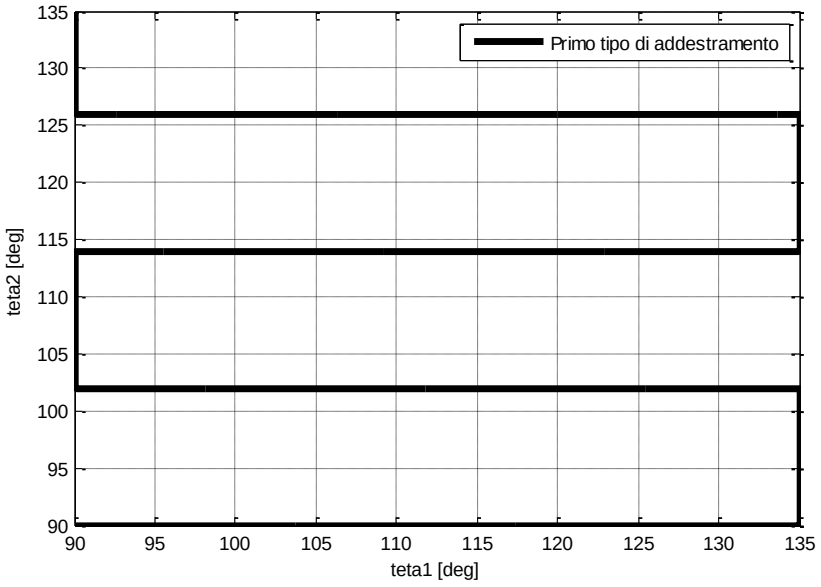

Figura 5.6 Prima traiettoria di addestramento

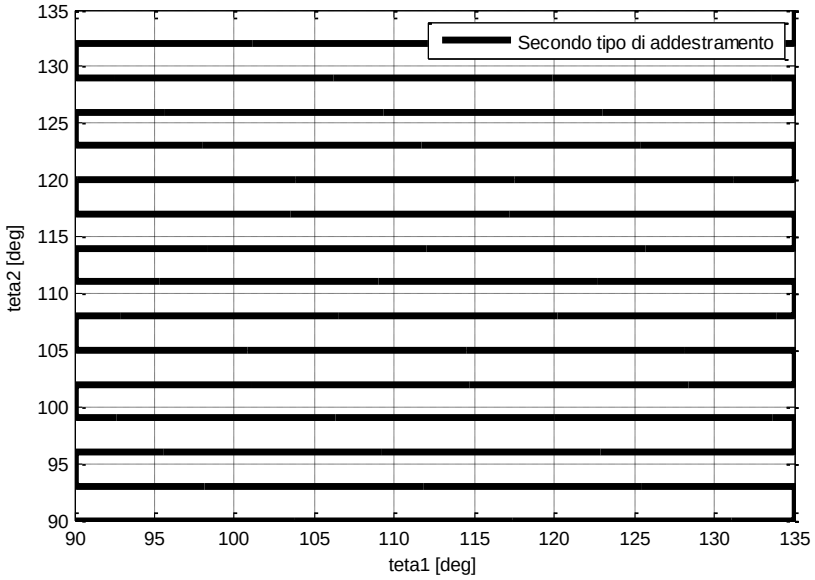

Figura 5.7 Seconda traiettoria di addestramento

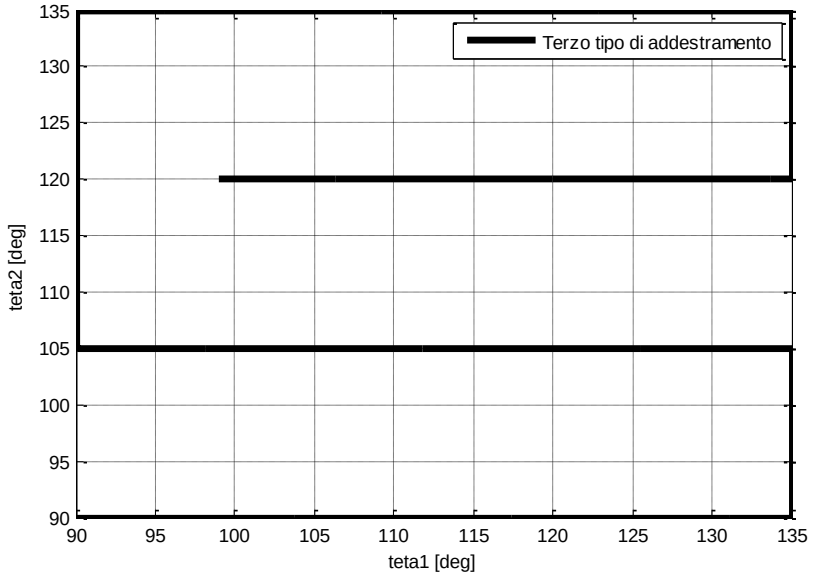

Figura 5.8 Terza traiettoria di addestramento

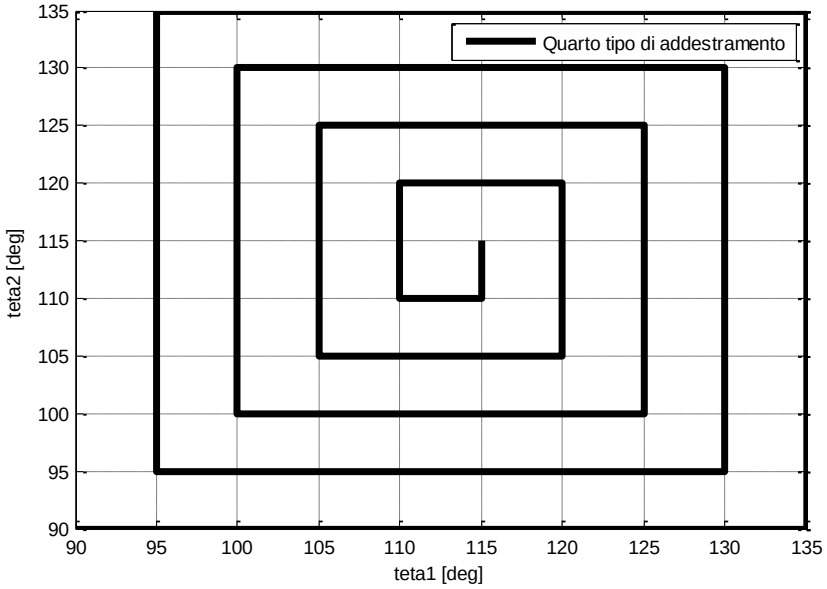

Figura 5.9 Quarta traiettoria di addestramento

Per ogni set di dati ottenuto dalla co-simulazione è stata addestrata una rete neurale. La rete migliore risulta essere quella che dà i risultati migliori una volta valutato il manipolatore in una condizione di funzionamento differente da quella che ha fornito i dati di addestramento. Dopo aver scelto la rete migliore tra queste, essa viene ulteriormente testata a fronte di altre configurazioni ritenute determinanti per la scelta del modello. La rete migliore è quella addestrata con la traiettoria di Figura 5.9, che corrisponde agli andamenti nel tempo presentati in Figura 5.10. Nei paragrafi seguenti saranno riportate le analisi effettuate. In tratteggio è riportato l'andamento del primo link del manipolatore, in linea continua l'andamento degli ultimi due link.

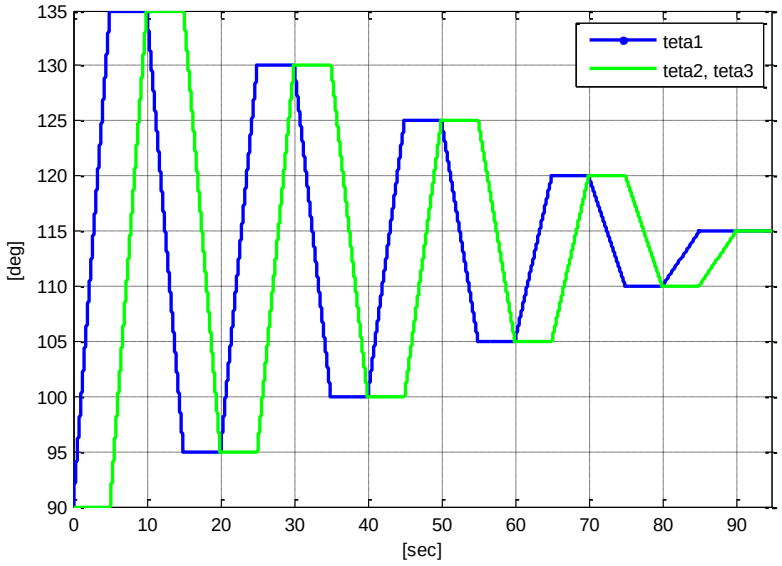

Figura 5.10 Andamento delle posizioni angolari relative alla quarta traiettoria

Questo tipo di traiettoria è risultata la migliore tra quelle scelte nei diversi casi di addestramento, perché permette (relativamente al range considerato) di spazzare il maggior numero di condizioni di "equilibrio", e ciò si rivelerà fondamentale una volta giunti al momento di linearizzare il modello non lineare. Per il controllo di vibrazioni è infatti molto importante avere un modello da poter linearizzare a tratti, come già affermato in precedenza.

È utile illustrare i risultati relativi ai test di robustezza sul modello scelto, tutti riferiti a reti neurali in anello chiuso. Una prima parte è focalizzata sulle motivazioni che hanno portato alla scelta della rete tra le altre, mentre la seconda parte mostra ulteriori analisi sulla rete presa singolarmente. Sono state effettuate diverse prove al fine di scegliere la rete migliore, ma si riporta la più significativa. Come già detto in precedenza, l'obiettivo fissato è ottenere un modello che possa identificare bene il robot una volta che esso assume differenti configurazioni. Occorre però affiancare a questo obiettivo anche quello di

sottoporre il modello ad un set di dati che rappresenti effettivamente una configurazione realistica che assume il manipolatore durante il suo funzionamento. Dato che il robot studiato è impiegato nella distribuzione di calcestruzzo, è fondamentale che l'analisi sul comportamento migliore tra le reti sia basata su un ciclo di lavoro che sia il più vicino possibile a quello del robot reale. Il ciclo di lavoro scelto per l'analisi è costituito dalla sequenza di configurazioni riportata in Figura 5.11.

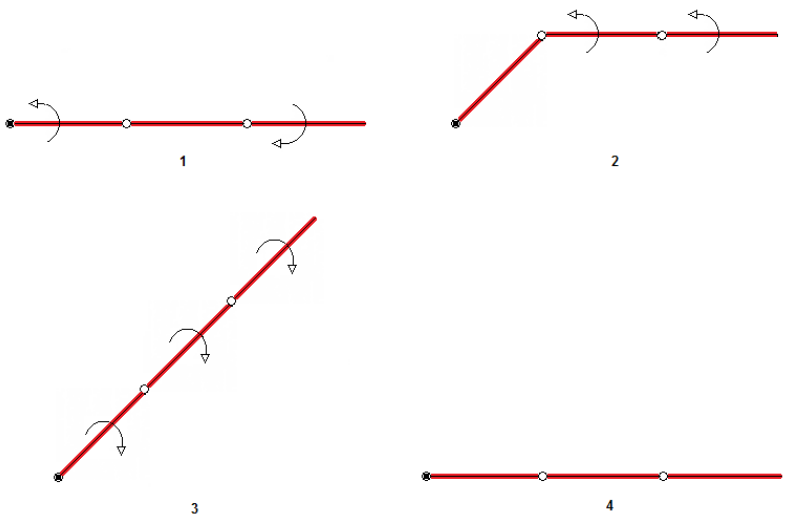

Figura 5.11 Manovra di studio che simula un reale ciclo del banco prova

I modelli NARX confrontati presentano comportamenti assai differenti, riassunti nell'ingrandimento delle sovrapposizioni di Figura 5.12. Si è riportata per semplicità la sola velocità angolare $\dot{\vartheta}_1$, relativa al primo link, in quanto le altre due velocità evolvono in maniera analoga.

È evidente come la prima rete da scartare sia la prima, che si allontana molto dal target in un intervallo della manovra. Si ricorda che essa invece rappresentava il set di dati impiegato per l'addestramento della rete neurale nel caso a 1 gdl. Questo risultato mostra come sia stato utile effettuare un'analisi più

approfondita nel caso a 3 gdl, che comporta la presenza di dinamiche incrociate che possono portare a risultati differenti e non scontati.

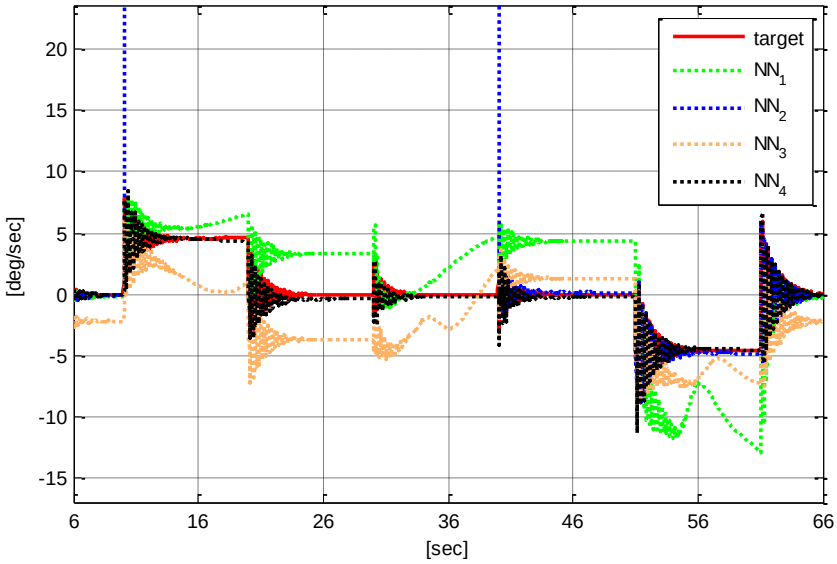

Figura 5.12 Confronto risposte dei quattro modelli addestrati

La rete che presenta un'uscita nettamente più vicina al target è proprio l'ultima rete, come già anticipato. Gli andamenti temporali in addestramento spazzano molte condizioni di equilibrio, quindi la risoluzione con cui viene descritto il piano $\vartheta_1 - \vartheta_2$ è decisamente migliore, anche rispetto alla seconda rete che invece, a prima vista, sembrava essere quella che avrebbe potuto descrivere il piano in modo più dettagliato. Nel caso della seconda rete entra in gioco però un altro elemento: l'ordine del modello. Non è detto infatti che un set di dati così ricco possa descrivere in modo completo un modello che abbia un ordine così basso (a parità di dati passati alla rete), anzi presenta un comportamento simile al modello "a serpente", caratterizzato da bassa risoluzione. La scelta ricade sull'ultima rete neurale.

In Figura 5.13 si riporta la sovrapposizione della traiettoria corrispondente alla manovra di studio e della traiettoria di addestramento. Come si può notare, solo una parte della traiettoria è sovrapposta ai dati di addestramento, quindi sarà molto più interessante vedere come il modello si comporta nel tratto in cui vengono passati dati ad esso estranei all'addestramento.

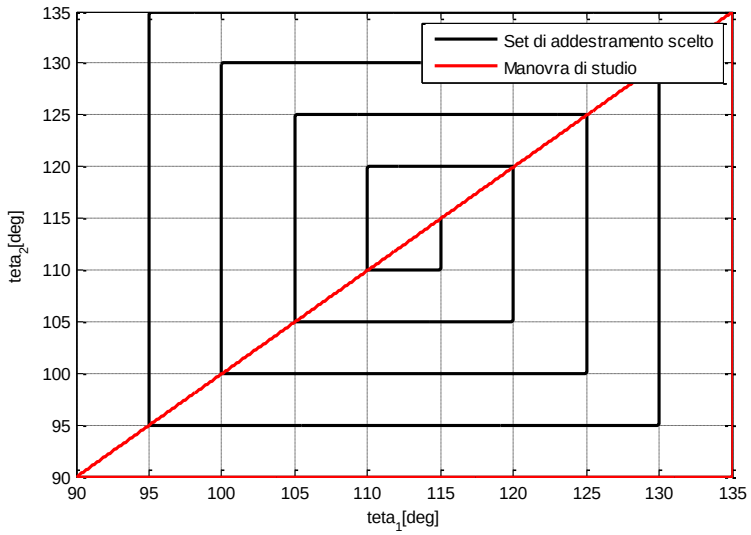

Figura 5.13 Sovrapposizione traiettorie di addestramento e della manovra di studio

In Figura 5.14 si riporta un ingrandimento della prima uscita del modello relativo agli istanti in cui la manovra passa per dati che il modello già conosce, contenuti nella fase di "training", corrispondenti alla configurazione orizzontale. In Figura 5.15 invece si illustra l'ingrandimento (sempre relativo alla prima uscita) relativo agli istanti in cui la manovra si porta da configurazione allineata a 45° a quella orizzontale (corrispondente alla diagonale in Figura 5.13).

Come era facile aspettarsi, il modello replica meglio la configurazione contenuta nell'addestramento, ma a parte un leggero scostamento ha ottime performance anche nel secondo caso riportato in Figura 5.15.

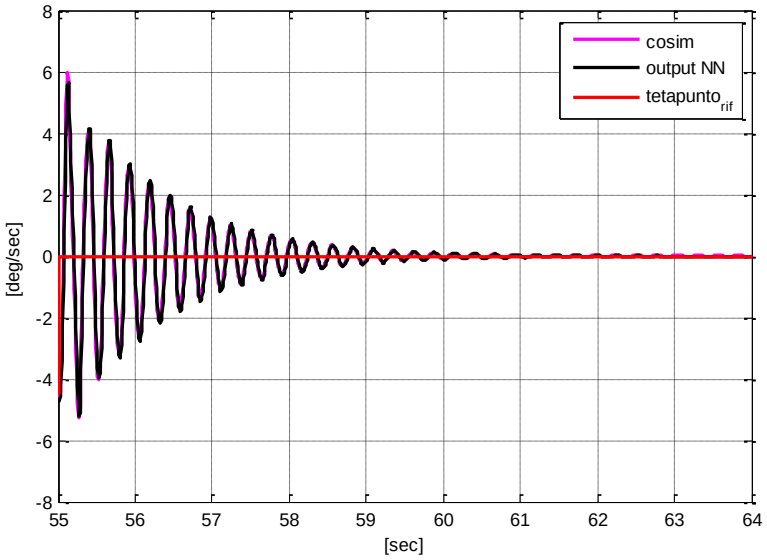

Figura 5.14 Confronto uscite desiderata e del modello a 3gdl per i dati di addestramento

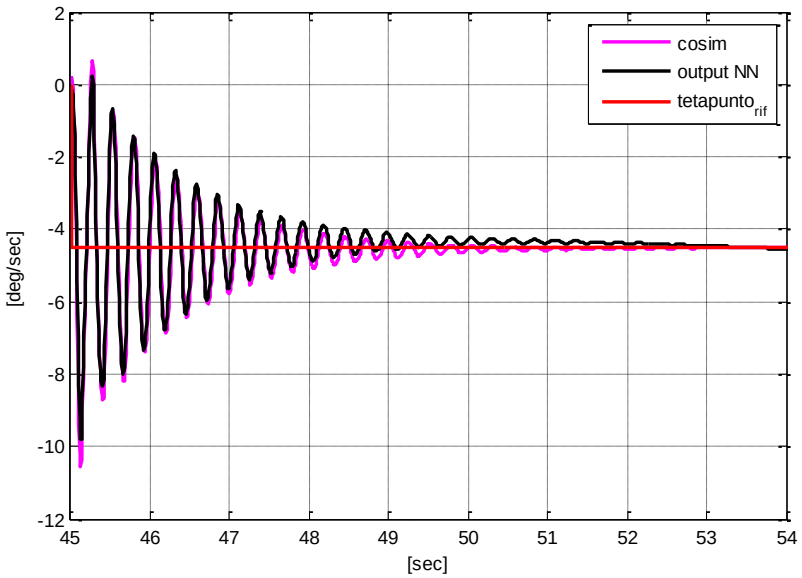

Figura 5.15 Confronto uscite desiderata e del modello a 3gdl per i dati che non appartengono a quelli di addestramento

Sempre seguendo questo ragionamento, dopo la scelta del miglior modello tra i quattro iniziali si effettuano delle prove che consistono nel testare le uscite della rete a fronte di altri dati che non sono contenuti in quelli impiegati in addestramento. Tra quelli ideati, i casi più critici sono quelli che contengono insiemi di dati che escono dal range di addestramento del modello. Sottoponendo il modello a un test di questo tipo, si intende testare la capacità del modello di estendere la mappatura degli ingressi anche nel caso in cui gli ingressi siano esterni all'intervallo memorizzato dalla rete in fase di addestramento. I risultati associati sono accettabili anche in questi casi ma non vengono qui riportati.

Una volta testato il modello non lineare trovato, esso può passare alla linearizzazione, che consente di identificare le frequenze proprie del sistema, come già fatto nel capitolo 4.

5.5. Linearizzazione e controllo

Viene di seguito presentata la tecnica di linearizzazione impiegata per tradurre il modello non lineare in linearizzato. La prima configurazione attorno cui si linearizza è quella da cui parte il robot quando evolve secondo la manovra di studio di Figura 5.11: la configurazione a braccio steso.

$$\begin{cases} \vartheta^{\circ}_1 = 0° \\ \vartheta^{\circ}_2 = 0° \\ \vartheta^{\circ}_3 = 0° \end{cases}, \qquad \begin{cases} \dot{\vartheta}_1 = 0°/sec \\ \dot{\vartheta}_2 = 0°/sec \\ \dot{\vartheta}_3 = 0°/sec \end{cases} \tag{5.7}$$

Solo successivamente si linearizza anche attorno alle configurazioni che il robot assume durante la manovra di Figura 5.11. È quasi superfluo ricordare che linearizzando il sistema si passa da una rete neurale artificiale non lineare ad una

matrice di trasferimento che collega tra loro input e output del modello. Come anticipato nel capitolo 4, è la presenza del sigmoide tangente che rende non lineare l'espressione ingresso-uscita:

$$tansig(x) = \frac{e^{2x} - e^{-2x}}{e^{2x} + e^{-2x}}, \qquad (5.8)$$

Esistono alcune tecniche per la linearizzazione del sigmoide che hanno come obiettivo raggiungere la miglior implementazione della funzione nel calcolatore [15], [20], e in funzione del fatto che i bit della macchina si contano in base due approssimano il sigmoide con una serie di step, assumendo la seguente espressione quantizzata:

$$h(n) = \frac{2^n - 2^{-n}}{2^n + 2^{-n}}, \ \forall n \in Z. \qquad (5.9)$$

Queste considerazioni non sono però il punto cardine della trattazione che si sta elaborando, e si preferisce linearizzare seguendo le tecniche tradizionali basate sul polinomio approssimante di Taylor, troncato al primo ordine. Nella trattazione seguente, si indicherà con il simbolo ϑ sia la posizione angolare effettiva del singolo braccio del robot sia la posizione di riferimento.

Partendo dal seguente modello:

$$\vec{\vartheta}_{NN}(t) = f\left(\vec{\vartheta}(t-1), \vec{\vartheta}(t-2), \vec{\vartheta}(t-1), \vec{\vartheta}(t-2)\right), \qquad (5.10)$$

caratterizzato da uno dei suoi equilibri:

$$\overline{x} = f\left(\overline{\dot{\vartheta}_1}, \overline{\dot{\vartheta}_2}, \overline{\dot{\vartheta}_3}, \overline{\vartheta_1}, \overline{\vartheta_2}, \overline{\vartheta_3}\right), \qquad (5.11)$$

occorre portarlo nella forma:

$$\delta\overrightarrow{\dot{\vartheta}}_{NN}(t) = [A_1]\,\delta\vec{\dot{\vartheta}}(t-1) + [A_2]\,\delta\vec{\dot{\vartheta}}(t-2) + [B_0]\,\delta\vec{\vartheta}(t-1) + [B_1]\,\delta\vec{\vartheta}(t-2), \qquad (5.12)$$

$$[A_1] = \begin{Bmatrix} \frac{\partial f_1()}{\partial \dot{\vartheta}_1(t-1)}\Big|_{\bar{x}} & \frac{\partial f_1()}{\partial \dot{\vartheta}_2(t-1)}\Big|_{\bar{x}} & \frac{\partial f_1()}{\partial \dot{\vartheta}_3(t-1)}\Big|_{\bar{x}} \\ \frac{\partial f_2()}{\partial \dot{\vartheta}_1(t-1)}\Big|_{\bar{x}} & \frac{\partial f_2()}{\partial \dot{\vartheta}_2(t-1)}\Big|_{\bar{x}} & \frac{\partial f_2()}{\partial \dot{\vartheta}_3(t-1)}\Big|_{\bar{x}} \\ \frac{\partial f_3()}{\partial \dot{\vartheta}_1(t-1)}\Big|_{\bar{x}} & \frac{\partial f_3()}{\partial \dot{\vartheta}_2(t-1)}\Big|_{\bar{x}} & \frac{\partial f_3()}{\partial \dot{\vartheta}_3(t-1)}\Big|_{\bar{x}} \end{Bmatrix} \quad (5.13)$$

$$[A_2] = \begin{Bmatrix} \frac{\partial f_1()}{\partial \dot{\vartheta}_1(t-2)}\Big|_{\bar{x}} & \frac{\partial f_1()}{\partial \dot{\vartheta}_2(t-2)}\Big|_{\bar{x}} & \frac{\partial f_1()}{\partial \dot{\vartheta}_3(t-2)}\Big|_{\bar{x}} \\ \frac{\partial f_2()}{\partial \dot{\vartheta}_1(t-2)}\Big|_{\bar{x}} & \frac{\partial f_2()}{\partial \dot{\vartheta}_2(t-2)}\Big|_{\bar{x}} & \frac{\partial f_2()}{\partial \dot{\vartheta}_3(t-2)}\Big|_{\bar{x}} \\ \frac{\partial f_3()}{\partial \dot{\vartheta}_1(t-2)}\Big|_{\bar{x}} & \frac{\partial f_3()}{\partial \dot{\vartheta}_2(t-2)}\Big|_{\bar{x}} & \frac{\partial f_3()}{\partial \dot{\vartheta}_3(t-2)}\Big|_{\bar{x}} \end{Bmatrix} \quad (5.14)$$

$$[B_0] = \begin{Bmatrix} \frac{\partial f_1()}{\partial \vartheta_1(t-1)}\Big|_{\bar{x}} & \frac{\partial f_1()}{\partial \vartheta_2(t-1)}\Big|_{\bar{x}} & \frac{\partial f_1()}{\partial \vartheta_3(t-1)}\Big|_{\bar{x}} \\ \frac{\partial f_2()}{\partial \vartheta_1(t-1)}\Big|_{\bar{x}} & \frac{\partial f_2()}{\partial \vartheta_2(t-1)}\Big|_{\bar{x}} & \frac{\partial f_2()}{\partial \vartheta_3(t-1)}\Big|_{\bar{x}} \\ \frac{\partial f_3()}{\partial \vartheta_1(t-1)}\Big|_{\bar{x}} & \frac{\partial f_3()}{\partial \vartheta_2(t-1)}\Big|_{\bar{x}} & \frac{\partial f_3()}{\partial \vartheta_3(t-1)}\Big|_{\bar{x}} \end{Bmatrix} \quad (5.15)$$

$$[B_1] = \begin{Bmatrix} \frac{\partial f_1()}{\partial \vartheta_1(t-2)}\Big|_{\bar{x}} & \frac{\partial f_1()}{\partial \vartheta_2(t-2)}\Big|_{\bar{x}} & \frac{\partial f_1()}{\partial \vartheta_3(t-2)}\Big|_{\bar{x}} \\ \frac{\partial f_2()}{\partial \vartheta_1(t-2)}\Big|_{\bar{x}} & \frac{\partial f_2()}{\partial \vartheta_2(t-2)}\Big|_{\bar{x}} & \frac{\partial f_2()}{\partial \vartheta_3(t-2)}\Big|_{\bar{x}} \\ \frac{\partial f_3()}{\partial \vartheta_1(t-2)}\Big|_{\bar{x}} & \frac{\partial f_3()}{\partial \vartheta_2(t-2)}\Big|_{\bar{x}} & \frac{\partial f_3()}{\partial \vartheta_3(t-2)}\Big|_{\bar{x}} \end{Bmatrix} \quad (5.16)$$

Per ottenere il modello locale occorre includere l'informazione dell'equilibrio. Il modello lineare finale è il seguente [10], [19]:

$$\vec{\vartheta}_{NN}(t) = [A_1]\vec{\dot{\vartheta}}(t-1) + [A_2]\,\vec{\dot{\vartheta}}(t-2) + [B_0]\,\vec{\vartheta}(t-1) + [B_1]\,\vec{\vartheta}(t-2) - [C] - [D] + I\bar{\dot{\vartheta}}, \quad (5.17)$$

$$\text{con } [C] = \begin{cases} \sum_{k=0}^{3}[A1_{1,k}\ A2_{1,k}]\,\bar{\vartheta} \\ \sum_{k=0}^{3}[A1_{2,k}\ A2_{2,k}]\,\bar{\vartheta} \\ \sum_{k=0}^{3}[A1_{3,k}\ A2_{3,k}]\,\bar{\vartheta} \end{cases}, \quad [D] = \begin{cases} \sum_{k=0}^{3}[B0_{1,k}\ B1_{1,k}]\,\bar{\dot{\vartheta}} \\ \sum_{k=0}^{3}[B0_{2,k}\ B1_{2,k}]\,\bar{\dot{\vartheta}}. \\ \sum_{k=0}^{3}[B0_{3,k}\ B1_{3,k}]\,\bar{\dot{\vartheta}} \end{cases} \quad (5.18)$$

È stato creato uno script che esegue i passi riportati per la linearizzazione attorno ad una generica configurazione.

In Figura 5.16 si riporta uno dei quattro plot relativi alle uscite x_i dei quattro neuroni nascosti, sempre con braccio in configurazione orizzontale.

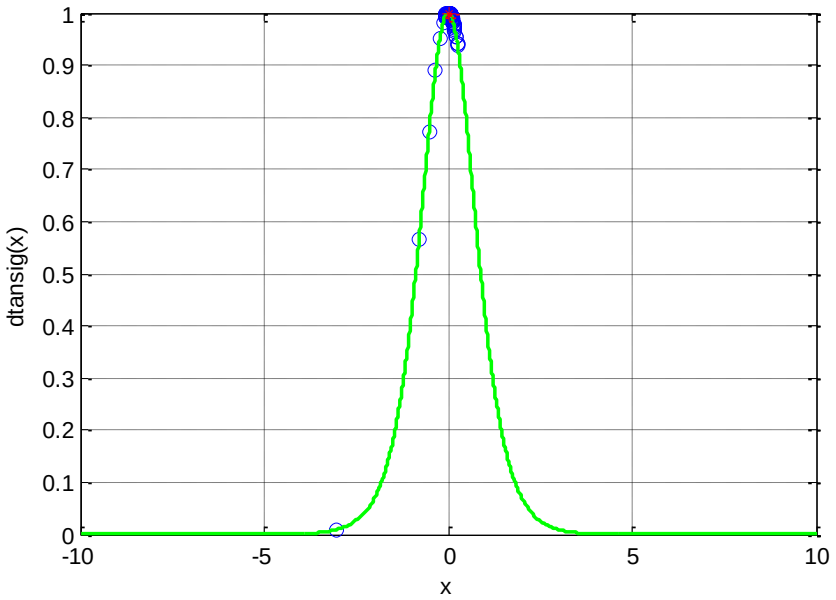

Figura 5.16 Derivata del sigmoide tangente

Eccitando uno per volta gli ingressi del sistema, si scopre che vengono eccitati solo i primi tre modi del sistema, come presentano i grafici degli stimatori H1 relativi a $\dot{\vartheta}_2$ in Figura 5.17. Ciò quindi riconferma che le frequenze proprie contenenti un maggior contributo armonico sono le prime tre, e che le restanti sono dunque trascurabili. Ha poco senso quindi cercare un modello di ordine maggiore del secondo, se si vogliono tenere in conto le dinamiche effettive del sistema.

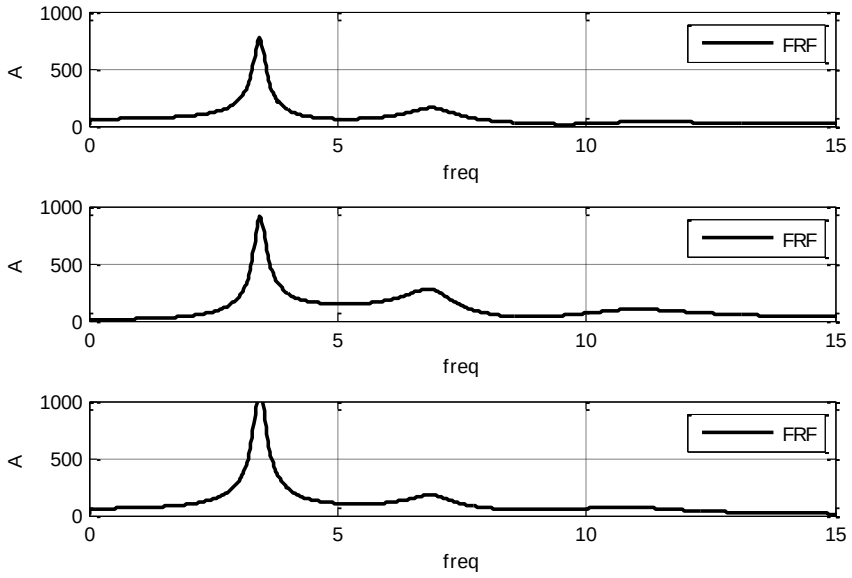

Figura 5.17 Stimatori H1 rispetto a $\dot{\vartheta}_2$ ottenuti forzando un ingresso alla volta

In Figura 5.18 si riporta il confronto tra la funzione di risposta in frequenza ricavata dalla cosimulazione e quella ottenuta in seguito all'identificazione tramite rete neurale nel caso del manipolatore in configurazione orizzontale.

Le frequenze proprie identificate tramite questo metodo sono le seguenti:

3.69 Hz, 7.47 Hz e 11.17 Hz,

molto vicine a quelle del sistema di studio.

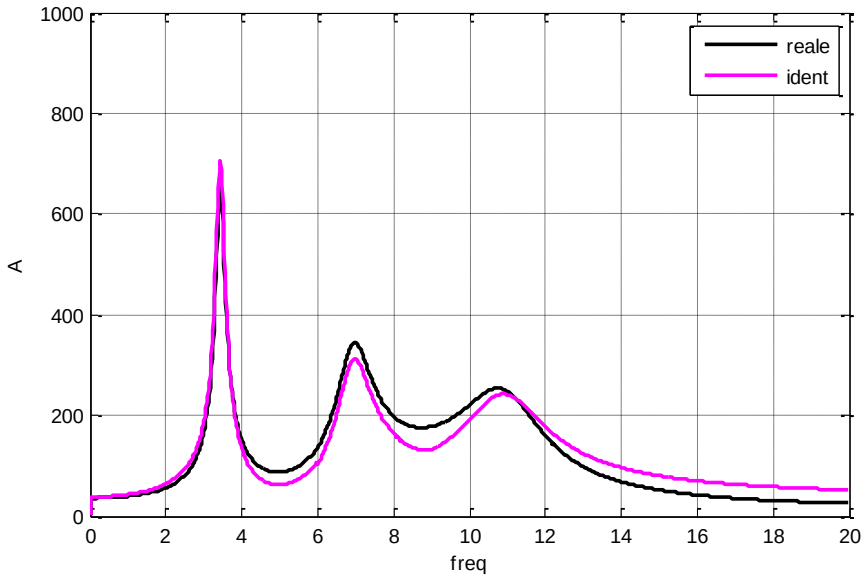

Figura 5.18 Funzione di risposta in frequenza della prima configurazione

Anche tramite una semplice analisi visiva si può notare come l'identificazione sia andata a buon fine. Si può ora scrivere il modello in forma di stato avente come stati le coordinate modali (posizione e velocità angolare di ogni link: in totale 6 stati), e tramite questo passaggio giungere al controllo ottimo in posizione, sfruttando l'informazione sulle velocità angolari fornita dalla rete neurale.

Per completezza si riportano anche le funzioni di risposta in frequenza delle altre due configurazioni di Figura 5.11 e anche in questi due casi si nota come ci sia una buona corrispondenza tra identificazione e reale.

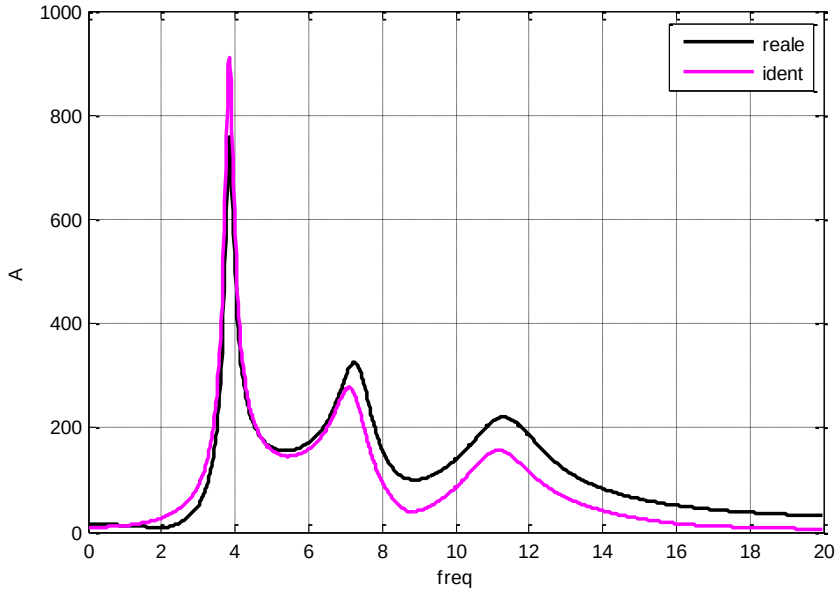

Figura 5.19 Funzione di risposta in frequenza della seconda configurazione

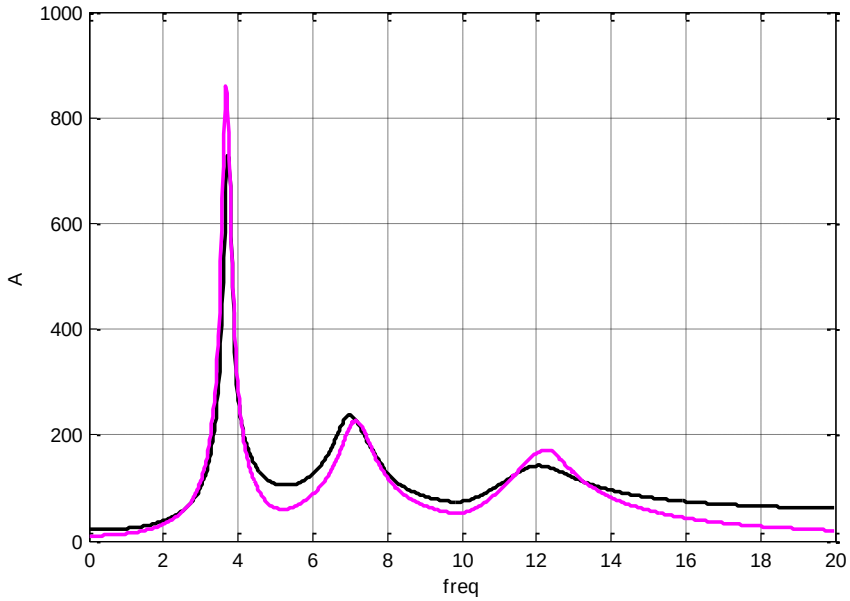

Figura 5.20 Funzione di risposta in frequenza della terza configurazione

5.5.1 Controllo ottimo

In Figura 5.21 si può vedere lo schema di controllo impiegato. Dato che si tratta di un controllo modale, gli stati sono coordinate modali (non note a priori e solitamente non ricollegabili a reali grandezze fisiche). Lo schema necessita quindi di un blocco osservatore (in Figura 5.21) che però ha come ingressi non direttamente le ϑ e le $\dot{\vartheta}$ ma bensì i *delta* corrispondenti. Questo perché l'obiettivo del controllo è effettuare l'inseguimento di una certa storia temporale diversa da zero, che necessita di una legge di controllo non del tipo $u = -Kx$ ma bensì $\Delta u = -K\Delta x$, con K la matrice dei guadagni del controllore. Sempre in Figura 5.21, si mostra come la rete neurale adatti sia l'osservatore che la matrice K di controllo. Questa operazione non è fatta online ma precalcolando le matrici A, B, C, D relative alle diverse configurazioni mostrate in Figura 5.11 ed aggiornando lo schema di controllo ogni 10 secondi.

Rispetto al caso più semplice del robot a singolo braccio, occorre effettuare più prove per la scelta delle matrici Q e R, a causa del maggior numero di stati. Dopo numerose prove si è optato per le seguenti:

$$\text{Q} = diag(1100,900,900,1,1,1),\ \text{R} = diag(0.1,0.1,0.1).$$

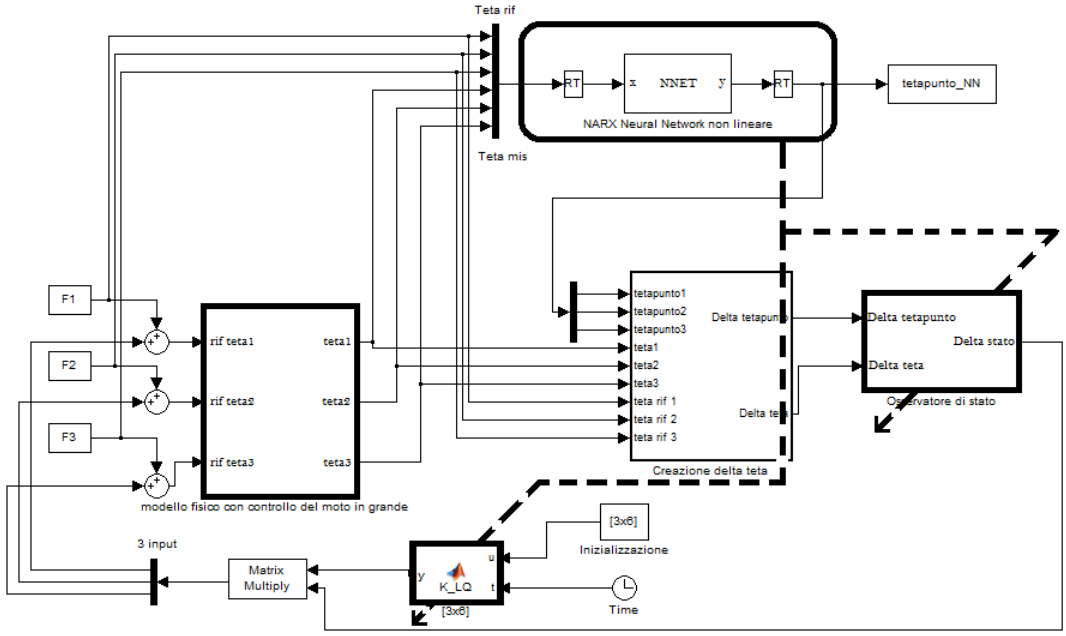

Figura 5.21 Controllo di vibrazioni per il caso di studio a 3 gdl con legge di controllo adattativa

La matrice Q non ha tutti i valori uguali a causa del fatto che nell'anello di controllo del PD la risposta del primo link era peggiore rispetto alle altre due. Per smorzare le oscillazioni eccessive generate da una taratura approssimata dell'anello interno si è necessariamente pesata di più la coordinata modale corrispondente. La scelta di questo tipo di controllo è dovuta proprio al fatto che esso garantisce performance decisive anche nel caso di strutture che rispondono con oscillazioni importanti. Questo è uno dei motivi per cui questa tecnica di controllo non è da considerarsi nuova nell'ambito del controllo di vibrazioni.

Come primo set di grafici si propongono le risposte nelle Figure 5.22, 5.23 e 5.24, per cui la storia temporale scelta, secondo la quale fare evolvere il manipolatore prevede che i primi due link eseguano un passaggio da 90° a 95°, mentre il terzo resti in posizione orizzontale. In questo caso non è stato effettuato nessun tipo di aggiornamento nell'anello di controllo.

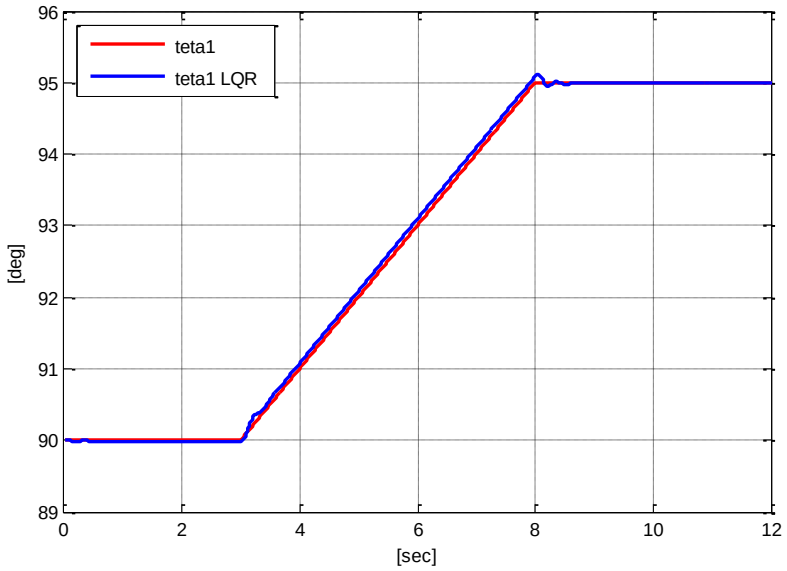

Figura 5.22 Confronto uscite ϑ_1 desiderata e del controllo LQR senza azione adattativa

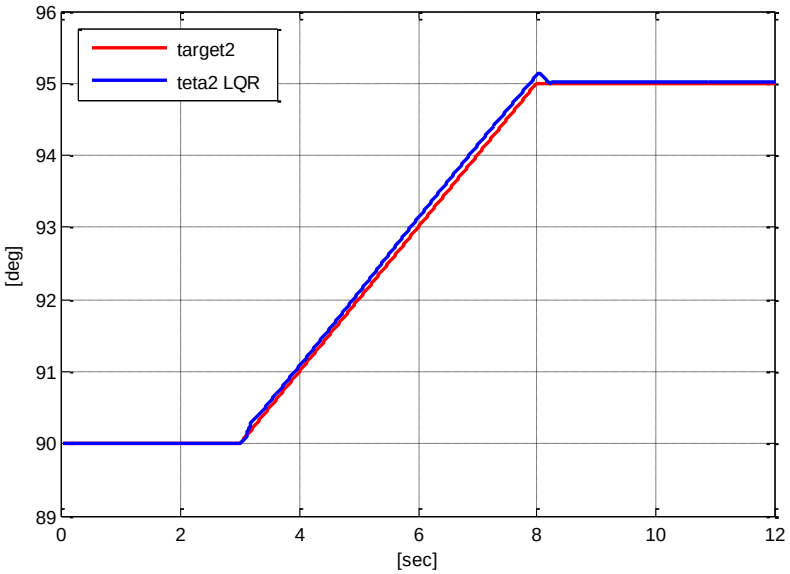

Figura 5.23 Confronto uscite ϑ_2 desiderata e del controllo LQR senza azione adattativa

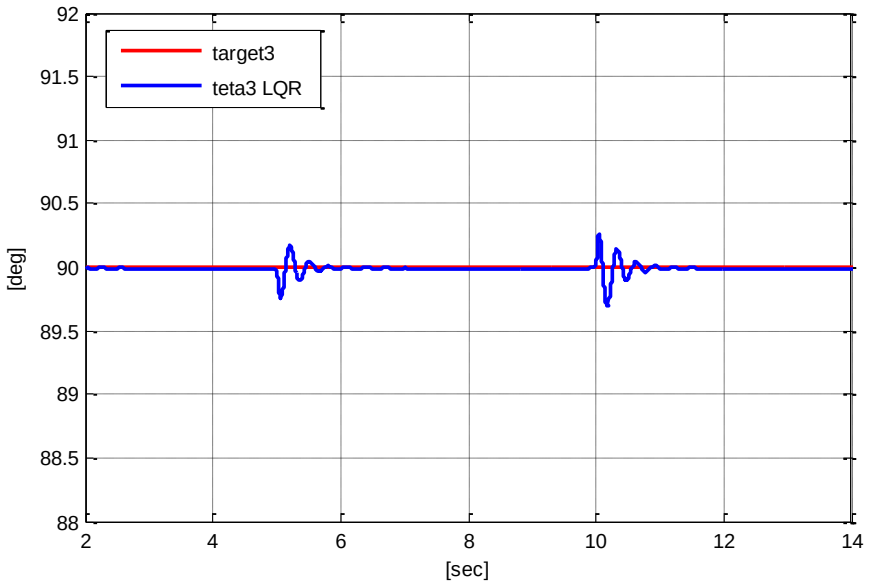

Figura 5.24 Confronto uscite ϑ_3 desiderata e del controllo LQR senza azione adattativa

Nonostante sia garantito un buon inseguimento, si nota come l'uscita del sistema controllato a 95° sia di poco peggiore a quello dei 90° (visibile soprattutto in Figura 5.23). Questo aspetto non si sarebbe verificato se si fosse deciso di adattare l'azione di controllo alla configurazione relativa ai 95°. I 95° si trovano infatti al di fuori dell'intorno di 90°, in quanto il manipolatore compie quasi l'undici percento del range di lavoro scelto. L'idea di far passare il manipolatore da 90° a 95° è quindi il primo passo all'approccio dell'adattamento.

Risulta utile applicare il controllo LQR senza adattamento al sistema che evolve secondo la manovra di Figura 5.11 e confrontarlo con la risposta del controllo LQR con l'adattamento. Il confronto è riportato per una sola grandezza nelle Figure 5.25 e 5.26 ed è evidente come sia migliore il controllo con l'adattamento.

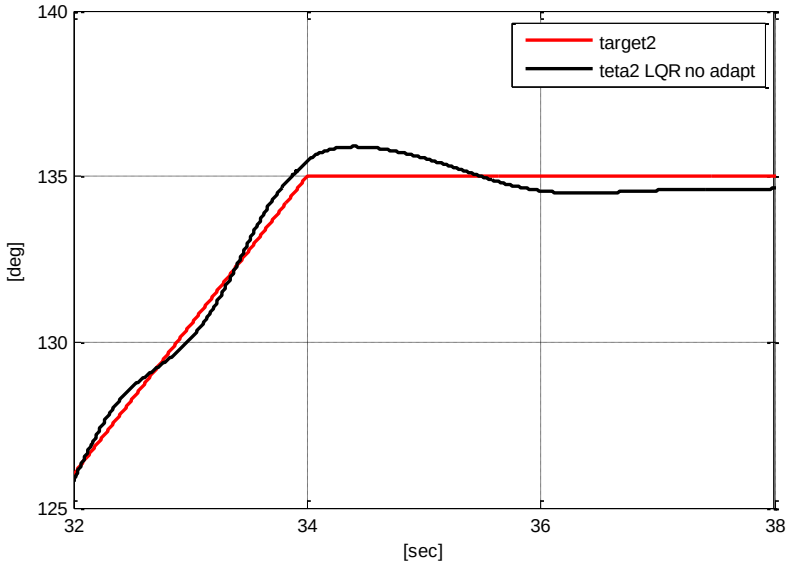

Figura 5.25 Dettaglio confronto uscite $\vartheta_1$desiderata e del controllo LQR senza azione adattativa, seguendo la manovra di studio

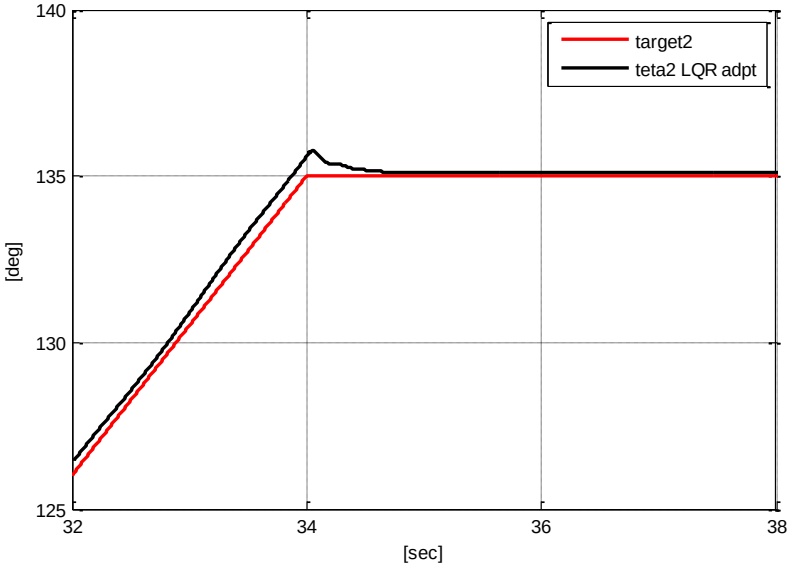

Figura 5.26 Dettaglio confronto uscite $\vartheta_1$desiderata e del controllo LQR con azione adattativa, seguendo la manovra di studio

Nelle Figure 5.27, 5.28 e 5.29 si riportano le risposte delle tre posizioni angolari alla traiettoria di Figura 5.11 sovrapposte alla risposta del sistema senza l'anello di controllo LQR adattativo, cioè semplicemente sotto l'azione del PD.

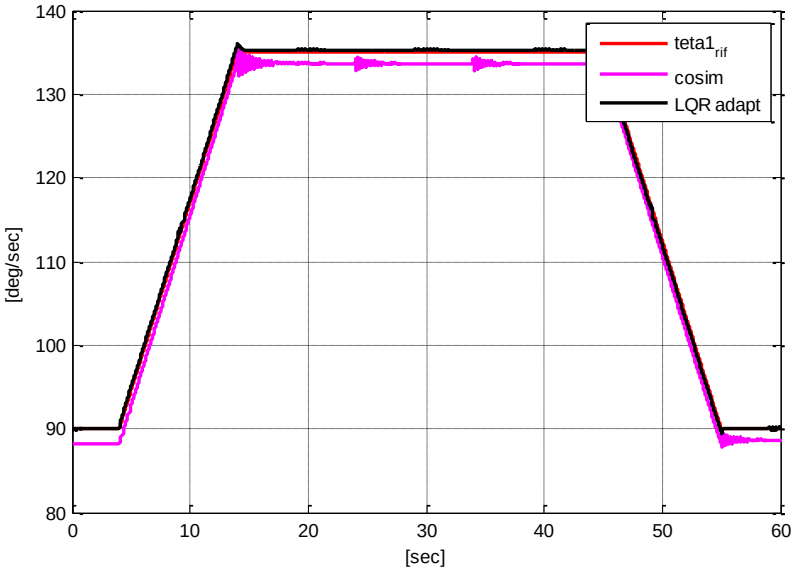

Figura 5.27 Confronto uscite ϑ_1 desiderata, del controllo LQR con azione adattativa e del regolatore PD, seguendo la manovra di studio

Già dal primo grafico si nota come l'inseguimento sia migliore, infatti la cosimulazione presentava un offset poco riducibile e oscillazioni poco smorzate al cambio di legge di moto.

Le oscillazioni presenti nella risposta del controllo LQR adattativo ogni 10 secondi (a partire dai 5 secondi iniziali) sono relative al cambio di matrici. Se avessero avuto una dinamica eccessiva si sarebbero potute smorzare cercando di attenuare l'azione di controllo proprio negli istanti di cambio matrice.

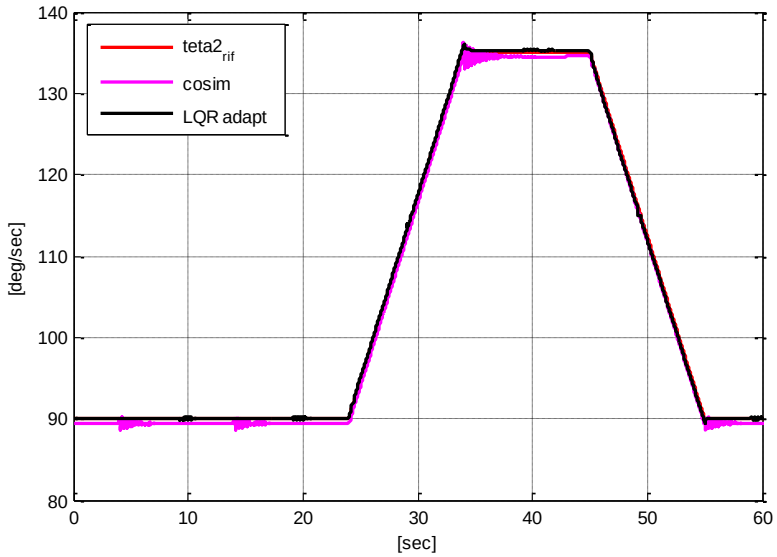

Figura 5.28 Confronto uscite ϑ_2 desiderata, del controllo LQR con azione adattativa e del regolatore PD, seguendo la manovra di studio

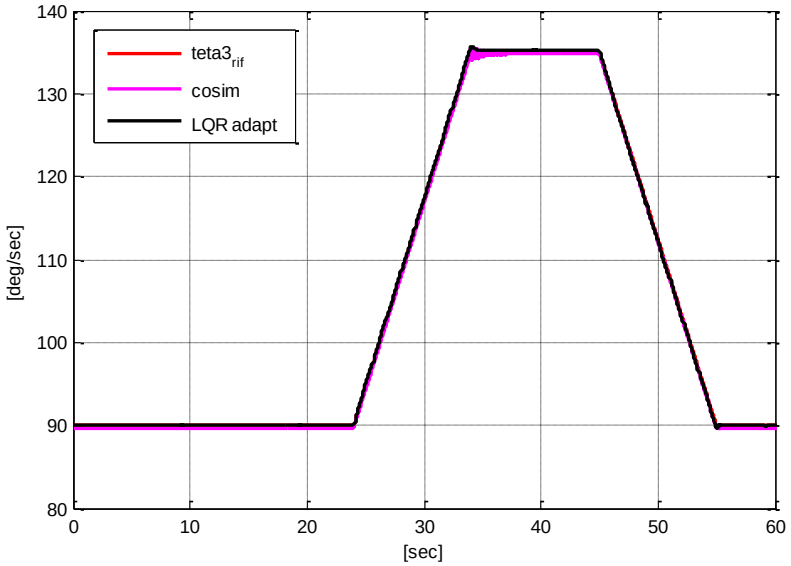

Figura 5.29 Confronto uscite ϑ_3 desiderata, del controllo LQR con azione adattativa e del regolatore PD, seguendo la manovra di studio

6. Conclusioni e sviluppi futuri

Il presente lavoro ha portato a una logica di controllo ottimo classica impiegata per il controllo di vibrazioni, ma con connotati innovativi. L'idea di base rilevante riguarda l'impiego delle reti neurali artificiali per la modellazione del sistema, tenendo conto degli aspetti fortemente non lineari che definiscono la sua natura senza però dover far ricorso ad un numero di variabili estremamente grande. Rispetto alle altre tecniche di controllo già presenti ed impiegate da tempo, la presenza di un modello non lineare di partenza rappresenta un grande vantaggio, perché permette di lasciare offline l'addestramento della rete neurale e la linearizzazione a tratti, che comportano un dispendio di tempo e di capacità computazionale non eccessivi.

Riguardo il modello, la rete neurale rappresenta un ottimo strumento per modellare un sistema non lineare, in quanto l'espressione tra ingresso e uscita è molto semplice ed esistono diversi modi per linearizzare il sigmoide, anche se lo sviluppo secondo il polinomio di Taylor offre un risultato più che accettabile. È chiaro come sia fondamentale scegliere un livello di complessità della rete che sia adeguato al problema esaminato, per evitare quindi un sovradimensionamento che può portare ad un overfitting di dati. Occorre però anche prestare attenzione al gruppo di dati da collezionare per addestrare la rete, che devono essere scelti in funzione di quello che si vuole ottenere. Questo

aspetto critico è però comune a tutti i modelli per cui è necessaria una fase di apprendimento.

L'addestramento è svolto usando un'evoluzione dei minimi quadrati ricorsivi, quindi permette di velocizzare quella che sarebbe l'operazione più lunga e potrebbe portare a scartare un modello come questo rispetto ad altre tipologie più tradizionali di modelli. È chiaro però come sia determinante non sovraccaricare l'apprendimento evitando di fornire un numero eccessivo di campioni, che sovraccarica inutilmente la computazione.

Si è voluto inoltre dare un significato fisico all'ordine della rete neurale NARX collegando direttamente l'ordine del modello con il numero di frequenze proprie realmente interessanti per l'applicazione in questione, in seguito alla risposta del sistema a fronte di eccitazione esterna.

Sempre dal punto di vista dell'identificazione, viene dato risalto soprattutto alle velocità angolari del sistema, in quanto sul banco prova non sono stati installati dei sensori in tal senso, è sembrato quindi ragionevole interessarsi nel creare un modello a rete neurale che potesse fornire questa grandezza all'osservatore, impiegato per la realizzazione della legge di controllo.

L'altra particolarità di questo approccio presentato riguarda il controllo di vibrazioni, in particolare l'operazione di adattamento, che avviene in precisi istanti di tempo stabiliti a priori, tenendo conto della traiettoria secondo cui si vuole far evolvere il manipolatore, della velocità della stessa e con l'obiettivo di non sollecitare troppo la struttura. Per rispettare quest'ultimo requisito è risultato fondamentale evitare di imporre il cambio di legge di controllo negli istanti in cui si modifica la legge di moto. Altre tipologie di controllo di vibrazioni che prevedono l'adattamento non seguono il criterio qua esposto, ma piuttosto si basano sulla modifica della legge di controllo in istanti di tempo che sono funzione della configurazione in cui si trova il manipolatore. È naturale

comprendere come in quei casi possa risultare utile un modello identificativo che possa unificare tutte le informazioni derivanti dai modelli linearizzati. Questi risultati possono perciò essere visti come punto di partenza per ottenere una *rete di modelli lineari locali (Local Model Network),* modello a rete neurale costituito da più modelli linearizzati in diverse configurazioni di equilibrio [12], [18], [19].

Come fatto già in questo lavoro, l'idea consiste nell'identificare numerose condizioni di equilibrio (che nel caso di un robot possono essere definite *condizioni di linearizzazione*) da impiegare in un controllore che possa generare l'azione di controllo sulla base del modello in quel momento identificato, facendo evolvere il manipolatore attraverso numerosi stadi linearizzati. La differenza sostanziale rispetto al modello ottenuto in questa tesi sta nel fatto che la rete di modelli lineari locali permette di unire l'informazione derivante da ogni singolo modello linearizzato, pesando ogni uscita appartenente ad ogni sottomodello linearizzato.

Sarebbe interessante che il controllore adattativo ottenuto a valle dell'impiego della rete a modelli locali adattasse l'azione di controllo in funzione della configurazione assunta dal robot in quel preciso istante, e non fissando a priori un intervallo di tempo prestabilito in funzione della traiettoria da descrivere.

L'impiego delle reti a modelli locali potrebbe comportare un maggior bagaglio di matrici precalcolate, ma questo aspetto è chiaramente legato al tipo di lavoro che il robot deve svolgere.

7. Bibliografia

[1] L. Zuo. *Effective and Robust Vibration Control Using Series multiple tuned-mass dampers*. Journal of Vibration and Acoustics (2009).

[2] DongBin Lee and C. Nataraj. *Model-Based Adaptive Tracking Control of Linear Time-Varying System with Uncertainties.* Intech.

[3] R.Guidorzi. *Multivariable system identification*. Bononia University Press.

[4] O.Nelles. *Identification of nonlinear systems.* Springer.

[5] G. Corriga, A. Giua, G. Usai. *An implicit gain-scheduling controller for cranes.* IEEE transational on control systems technology.

[6] P. Korba, R. Babuska, H. Verbruggen, P. M. Frank. *Fuzzy Gain Scheduling: Controller and Observer Design Based on Lyapunov Method and Convex Optimization*. IEEE transactions on fuzzy systems.

[7] J. Novak, P. Chalupa, V. Bobal. *Local model networks for modelling and predictive control of nonlinear systems.* Tomas Bata University, faculty of applied Informatics.

[8] G. Diana, F. Cheli. *Dinamica dei sistemi meccanici*. Polipress.

[9] S. Bittanti. *Identificazione dei modelli e sistemi adattativi.* Pitagora editrice Bologna.

[10] L. V. Fausett. *Fundamentals of neural networks.* Prentice Hall editions.

[11] H. Hassoun. *Fundamentals of artificial neural networks.* The MIT Press.

[12] B. Hartmann. *Recent partitioning strategies for local model networks,* University of Siegen.

[13] K.Madsen, H. O. *Methods for nonlinear least squares problems.* Informatical and Mathematical modeing technical University of Denmark.

[14] S.Bittanti. *Simulazione, identicazione e controllo. Il caso di uno scambiatore di calore.* Pitagora editrice Bologna.

[15] V. Beiu, J. Peperstraete, J. Vandewalle, R. Lauwereins. *Close approximation of sigmoid functions by sum of steps for VSLI implementation of neural networks.* University of Belgium, University of Bucharest.

[16] P. Bolzern, R. Scattolini, N. Schiavoni. *Fondamenti di controllo automatici.* McGrawHill.

[17] B. Siciliano, L. Sciavicco, L. Villani, G. Oriolo. *Robotica: modellistica, pianificazione e controllo.* Mc Graw Hill.

[18] J. Novak, P.Chalupa, V. Bobal. *Local model networks for modelingand predictive control of nonlinear system.* Tomas Bata University, Faculty of Applied Informatics.

[19] Prof F. Previdi*: http://dinamico2.unibg.it/previdi/index.html.*

[20] K. Basterrextea, J. M. Tarela, I. Del Campo. *Approximation of sigmoid function and the derivative for the artificial neurons*. University of Basque Country, Elektronika eta Telecomunikazio Saila.

[21] *http://www.cs.berkeley.edu/~pabbeel/cs287-fa12/slides/NonlinearOptimizationForOptimalControl.pdf*

[22] S. Naidu. *Optimal control systems*. CRC press.

[23] D. Truong. *Neural networks for identification, prediction and control.* Springer.

[24] J. Wu. *Neural networks and simulation methods.* M. Dekker.

[25] *http://www.mathworks.it/it/help/nnet/ug/design-time-series-narx-feedback-neural-networks.html.*