UNIVERSITÁ POLITECNICA DELLE MARCHE
FACOLTÁ DI INGEGNERIA

Corso di Laurea Triennale in Ingegneria Biomedica

PIANIFICAZIONE DI TRAIETTORIE E MODELLAZIONE DI UN MANIPOLATORE ANTROPOMORFO

Trajectory planning and modeling of an anthropomorphic manipulator

RELATORE: ***Chiar.mo Prof. Sauro Longhi***
CORRELATORI: ***Dott. Alessandro Freddi, Dott.ssa Federica Verdini***

Tesi di laurea di
Luca A. Pettinari

A.A. 2014-2015

"Everything around you that you call life was made up by people that were no smarter than you. And you can change it, you can influence it. Once you learn that, you'll never be the same again."

(Steve Jobs)

Ringraziamenti

In queste poche righe desidero ringraziare tutti coloro che mi hanno aiutato e sostenuto per l'intero periodo di studio, soprattutto quelli che hanno dato colore e musica al ritmo monotono della vita del pendolare; in particolare dunque Giorgio, Vittoria e Lucilla, le mie sorelle acquisite, Alessio e Gian Marco. E, naturalmente, l'intero team di "Ciuturemoce", che ricorderó sempre con grande affetto: voi tutti siete stati i migliori compagni che io abbia mai potuto sperare di avere. Voglio ricordare anche chi, con un pizzico di esperienza in piú, ha sempre trovato le parole giuste, di sostegno e di conforto tra un esame e l'altro: per questo mi sembra ovvio ringraziare con amore i miei genitori, Alberto e Daniela, mia Nonna, Ines, e la carissima Ottavia.

Un doveroso ringraziamento va all'Ing. Ortenzi che mi ha seguito con scrupolo e accuratezza nello sviluppo di questa tesi, con un continuo scambio di opinioni e pareri, da pari a pari. Ringrazio i miei correlatori, l'Ing. Freddi e l'Ing. Verdini per le numerose revisioni e i preziosi consigli, sia in fase di direzione dei lavori, che nella stesura dell'elaborato finale. Ringrazio anche l'Ing. Monteriú, le cui preziose indicazioni e l'interesse personale hanno contribuito non poco allo sviluppo di questa tesi. Infine ringrazio il mio relatore, il nostro rettore Prof. Longhi per la disponibilitá e l'interesse nei confronti di questo lavoro.

Desidero ringraziare con affetto anche coloro che mi hanno visto crescere, il Prof. Rebichini, i Proff. Mancini e Poggi, per avermi trasmesso qualcosa in piú del mero insegnamento scolastico, e la Professoressa Severini, maestra di vita, la cui figura rimarrá scolpita nella mia memoria piú di quanto la sua umiltá gli abbia permesso di immaginarsi. E poi, come non ricordare i miei amici: Alessandro, Beatrice, Carlo, Chiara, Claudio, Davide e Dave, Elena, Federico, Giampa, i due Giorgi, le giá citate sorelle Burzi, Mattia, Mancio, Mauro, Sara, Sofia, Simone, e tutti gli altri.

Alla fine, dedico questo ultimo spazio ad Eleonora, l'unica persona al mondo che sia riuscita a capire, comprendere e perdonare ogni piccola sfumatura della mia personalitá, l'unica che sia stata capace di apprezzare i miei pregi tanto quanto i miei difetti, l'unica al mondo che mi abbia fatto scoprire il vero significato della parola "Amore".

Introduzione

L'obiettivo di questa tesi é lo studio e la pianificazione di traiettorie di un manipolatore antropomorfo, servendosi di un ambiente di simulazione che ne modelli la struttura cinematica. Pertanto lo sviluppo di modelli in simulazione é la parte preponderante di questo elaborato, in quanto fornisce lo strumento ideale per lo studio delle traiettorie con un approccio in spazio operativo, dove la conoscenza delle leggi orarie dei giunti permette di realizzare un determinato movimento che si desidera far compiere al manipolatore. L'analisi di tali leggi, che generalmente osservano precise specifiche, é l'argomento centrale della pianificazione di traiettorie.
Dunque, gli obiettivi posti per questo lavoro di tesi sono lo sviluppo di *modelli cinematici* di manipolatori antropomorfi, presi singolarmente o in coppia (facendo riferimento ad uno in particolare, il manipolatore MANUS), l'implementazione di *algoritmi di pianificazione* di traiettorie e la traduzione di questi aspetti in un *ambiente di simulazione* con annessa *validazione dei risultati* osservati in realtá virtuale.
Nello svolgimento di questa tesi si parte da una panoramica generale riguardante i manipolatori per compiti d'assistenza e le tecniche analitiche sviluppate per essi, conciliando concetti di meccanica e automatica. Dunque si procede con lo studio di un particolare manipolatore antropomorfo, il manipolatore MANUS, concentrando gli sforzi sulla modellazione e simulazione, e rispettando in esse (per quanto possibile) le caratteristiche di questo manipolatore. In ultima istanza, si é affrontato il problema della pianificazione del movimento di due particolari manipolatori antropomorfi (MANUS), disposti in posizione frontale, coordinati su una traiettoria circolare.
I capitoli della tesi saranno strutturati nel seguente ordine:

- Capitolo 1: generalitá sulla robotica assistiva e finalitá di questa branca, riportando una panoramica sulle caratteristiche dei manipolatori antropomorfi e sulle soluzioni adottate tramite l'uso di questi dispositivi in simbiosi con altre apparecchiature.
- Capitolo 2: principali nozioni matematiche che stanno alla base della pianificazione di traiettorie, soffermandosi su alcuni problemi teorici.
- Capitolo 3: breve descrizione dell'oggetto di studio di questa tesi, cioé il manipolatore MANUS, soffermandosi sui protocolli di comunicazione.
- Capitolo 4: descrizione in dettaglio dello sviluppo dei modelli in simulazione e dei relativi profili ingresso-uscita, seguendo gli approcci delineati nel capitolo teorico e con rimando all'implementazione dei codici in appendice.
- Capitolo 5: caratteristiche del simulatore, descrizione dei risultati e delle problematiche che si sono incontrate, commentando la bontá delle simulazioni e giustificando le soluzioni adottate.

Indice

Capitolo 1

Stato dell'arte dei manipolatori assistivi

La *robotica assistiva* é una branca della robotica che si propone di studiare le possibili soluzioni riabilitative e di servizio che nascono dalle interazioni tra i mezzi offerti da questa tecnologia e l'utenza di persone che generalmente ha parzialmente o totalmente perso l'uso di una particolare funzione (robotica riabilitativa) o che desidera effettuare compito automatizzabile o rischioso per l'utente stesso (robotica di servizio). La robotica riabilitativa (che é una sottobranca della robotica medica) studia tutti quei dispositivi che possono parzialmente sostituire i movimenti e le mansioni svolte per mezzo degli arti, specialmente quelli superiori: in parte, questa richiesta é esaudita dall'impiego di *robot manipolatori*, cioé bracci elettromeccanici in genere controllati dall'utente, capaci di svolgere le attivitá principali dell'arto superiore, come lo spostamento di oggetti o la presa di un utensile, in sinergia con unitá di supporto munite di sensori, batterie e appoggio per l'utente. Lo sviluppo di queste tecnologie ha suscitato interesse da parte della ricerca scientifica che mira ad un futuro (non troppo lontano) in cui questi dispositivi interagiranno in modo coordinato e interattivo con l'utente, il quale percepisce un miglioramento netto sulla qualitá della vita potendo svolgere in maniera indipendente le principali mansioni quotidiane.

1.1 Caratteristiche generali

Il requisito piú importante che ogni dispositivo di questa classe, e in generale ogni dispositivo di supporto umano o apparecchiatura biomedicale deve rispettare, é quello di non arrecare pericoli alla salute del paziente nel tempo in cui vi interagisce. Dunque anche per i robot manipolatori occorre garantire con ogni

mezzo un funzionamento sicuro, cioé non dannoso per l'utente e per il manipolatore stesso. Per far sí che ció avvenga e per garantire che i manipolatori svolgano effettivamente la funzione per cui sono stati progettati, é necessario dotarli di opportuni sistemi di controllo di posizione, al fine di adattarsi efficacemente ad ambienti di lavoro non strutturati, come quello domestico. Occorre considerare anche la richiesta di mobilitá per poter seguire il paziente nelle sue attivitá: il dispositivo deve garantire un ridotto ingombro e un'elevata efficienza, in modo da ridurre la richiesta di energia elettrica necessaria al suo funzionamento, tipicamente limitata. Una soluzione adottata é ad esempio l'uso di postazioni a carrozzella (in inglese *wheelchair*), su cui sono montate batterie di durata limitata, ricaricabili. Queste, oltre ad essere la sede di una sorgente di alimentazione mobile, rappresentano il mezzo di supporto sia per il robot (sostegno del manipolatore) sia per l'utente che abbia perso la funzione di locomozione; inoltre per raggiungere il massimo dell'interattivitá uomo-macchina, si richiede che sia munita anche di una *User Interface* (UI) semplificata, che comunichi con l'utente con modalitá visive o vocali e non di meno con l'esterno, per esempio, con centri medici (per chiamate di soccorso) o con le autoritá. Dovrebbe inoltre poter garantire all'utente anche di mantenere i propri contatti con persone conosciute grazie a strumenti come telefono, videoconferenza o Internet. Tuttavia si consideri che un dispositivo capace di soddisfare tutti i requisiti delineati precedentemente in maniera ottimale, aumenta inevitabilmente sia la complessitá di progetto che la sua stessa realizzabilitá. In letteratura sono stati delineati cinque criteri e i loro relativi indici di valutazione di un robot manipolatore [6]. Essi sono:

1. *Sicurezza*: la definizione qualitativa di questo criterio di valutazione si basa sulla misura della forza necessaria a generare qualsiasi tipo di infortunio ad una persona durante l'interazione umano-robot. L'incorporazione di vincoli elettromeccanici ha un'elevata incidenza sull'aumento di questo fattore.

2. *Resistenza all'urto*: si basa sulla stima dell'ammontare dei danni causati da un urto con l'ambiente esterno, rappresentato in genere da forze impulsive non previste nelle regolari mansioni del manipolatore. Predisporre un sistema di disaccoppiamento temporaneo attuatore-giunto puó aiutare a migliorare la valutazione del dispositivo sotto questo aspetto.

3. *Errore di posizionamento*: misura l'accuratezza della posizione raggiunta dal manipolatore rispetto alla posizione desiderata imposta dal controllore. In genere per i robot manipolatori la banda del segnale di posizione dei giunti, cioé l'intervallo di tutte quelle frequenze a cui si puó riscontrare una minima variazione della posizione angolare, é ampia, pertanto é possibile

effettuare azioni di controllo per un'ampia gamma di movimenti (e.g. spostamenti piccoli e lenti fino a movimenti rapidi ed impulsivi). L'aggiunta di un anello di retroazione nel controllore corregge eventuali dislocazioni dell'organo terminale rispetto alla posizione desiderata aumentando la ripetibilitá e la precisione del controllo.

4. *Dispendio energetico*: é l'indice qualitativo della capacitá del manipolatore di accumulare energia proveniente dall'esterno, di effettuare compiti con piccola attuazione e minor consumo e di essere indipendente da fonti di energia esterne per un intervallo di tempo considerevole.

5. *Adattabilitá*: rappresenta il compromesso tra sicurezza e performance dell'interazione. Un dispositivo del genere é capace di adattarsi sia a situazioni di routine con massima sicurezza e interoperabilitá, sia a modificare le velocitá e le accelerazioni del manipolatore in ambienti non convenzionali o situazioni particolari in cui si richiede maggiore potenza a sfavore della sicurezza.

Tra questi, é da tenere fortemente in considerazione l'indice dell'errore di posizione, dal momento che l'accuratezza e la ripetibilitá sono fattori fondamentali tanto quanto la sicurezza del braccio robotico nei confronti dell'utente. Bisogna preoccuparsi infatti che quest'ultimo, oltre a sentirsi a proprio agio con il dispositivo, possa controllarlo come se fosse "la naturale estensione del suo braccio", cioé che si trovi, in ogni instante, nel punto in cui egli vuole.

1.2 Fonti di errore

Le proprietá cinematico-dinamiche di un manipolatore valutate sul piano teorico possono risultare differenti all'atto pratico a causa di errori fisici, errori di assemblaggio e deformazioni elastiche. In questi casi il robot non si comporta secondo quello che é il suo progetto. A tale scopo sono definite le seguenti proprietá di un robot [2]:

- *Risoluzione*: é la misura della piú piccola variazione di posizione dell'organo terminale del robot manipolatore che puó essere rilevata o comandata. Questo parametro dipende principalmente dalla qualitá dei sensori utilizzati e dal loro posizionamento in una struttura robotica.

- *Ripetibilitá*: definisce la capacitá del robot manipolatore di ritornare su uno stesso punto dello spazio con identici movimenti effettuati in tempi diversi; puó essere intesa come una misura convenzionale della variabilitá dei risultati della misurazione della stessa quantitá o caratteristica nel caso

venga effettuata nelle stesse condizioni. Questo parametro dipende sia dalla risoluzione dei sensori, sia dall'accuratezza complessiva di costruzione della catena cinematica.

- *Accuratezza*: é definita come il massimo errore di posizione che si ottiene quando si muove il terminale del robot in un punto assegnato dello spazio.

Parallelamente alle proprietá di controllo di moto (che ruotano tutte su processi di misurazione, in quanto é la rivelazione dell'errore che ne permette tale controllo), si definiscono le categorie di errori riguardati questi sistemi meccanici, che dipendono dalle imprecisioni di lavorazione della struttura meccanica e delle componenti, attraverso cui avviene la trasmissione del moto. Essi sono:

- *Errori dinamici*: si verificano quando il robot é in movimento e sono causati dalle forze di giunto e dalle risonanze meccaniche indotte dal moto. Questa tipologia di errori si produce solo durante le fasi del moto e diventano trascurabili una volta esauritosi il transitorio.

- *Errori termici*: sono dovuti all'espansione termica dei componenti metallici in presenza di variazioni di temperatura, che possono essere indotte da cause interne come motori, cuscinetti, surriscaldamento, o da cause esterne come condizioni ambientali e sorgenti di calore.

- *Errori di sistema*: Questa categoria di errori comprende tutti quelli generati da cattiva calibrazione, imprecisioni dei sensori, bande morte negli organi di trasmissione del moto e servosistemi mal tarati.

Un calcolo attraverso la funzione cinematica, diretta o inversa, dá un risultato solo approssimato, dato che questi errori di varia natura non vengono modellati da questo approccio in quanto si basa solo su parametri nominali (modello ideale), trascurando altri parametri reali. Questi errori possono essere corretti solo identificando i difetti che li generano. Per aumentare la precisione e l'accuratezza del robot quindi si procede ad effettuare una compensazione degli errori, altrimenti detta *calibrazione*, il cui scopo é quello di trovare un'accurata relazione funzionale tra le letture dei trasduttori di giunto e la reale posizione dell'organo terminale del robot nello spazio di lavoro. Tale procedura ha un suo punto chiave nella fase di misurazione, che puó far variare i costi e la durata di tutto il processo in maniera considerevole.

1.3 Soluzioni adottate e sviluppi

L'applicazione della robotica nella riabilitazione e, in particolare, nell'assistenza alle persone disabili e agli anziani, é stata oggetto di studio approfondito

da parte di svariati gruppi di ricerca, soprattutto negli ultimi anni. Questo tentativo di fornire una risposta ai bisogni crescenti della societá moderna ha portato allo sviluppo di numerosi progetti che si differenziano tra loro per molti aspetti: primo tra tutti l'aspetto della complessitá tecnologica. Il primo tentativo di applicare la robotica al servizio delle persone affette da disabilitá motoria é stato l'impiego di *postazioni fisse*, caratterizzate dall'utilizzo di un manipolatore cartesiano, le cui guide erano tipicamente ancorate ad un tavolo, e dunque utilizzato per compiti di trasporto di oggetti in un ben delimitato spazio di lavoro [9], [13]. Normalmente l'utente controlla il manipolatore tramite un'interfaccia grafica con la quale é possibile assegnare un compito al robot, scegliendo tra alcuni giá predefiniti da eseguire. Per poter operare con le postazioni fisse, nasce l'esigenza di strutturare opportunamente l'ambiente circostante in modo da renderlo noto al sistema, limitando la mobilitá e l'adattabilitá del dispositivo alle molteplici situazioni. Queste apparecchiature, derivanti in modo piú o meno diretto da quelle utilizzate nel campo dell'automazione industriale, sono le meno diffuse tra quelle offerte dalla robotica assistiva, in quanto le proprietá di controllo di questi manipolatori non conciliano adeguatamente i criteri di valutazione esposti precedentemente (elevata inerzia dei bracci, velocitá elevate nell'azione di controllo, bassa adattabilitá ad ambienti poco strutturati). Un notevole miglioramento delle prestazioni richieste da questi dispositivi si é ottenuto dalle *postazioni a carrozzella* (*Wheelchair*), sulle quali é montato un robot manipolatore. Queste postazioni permettono all'utente di utilizzare il robot in contesti differenti, sia in ambienti chiusi che non, grazie al fatto di poter sfruttare il manipolatore unitamente alla mobilitá offerta dalla carrozzella elettrica. Questa soluzione presenta alcuni svantaggi tecnici dovuti soprattutto alla scarsa accuratezza del sistema a causa del fatto che esso non conosce a priori l'ambiente circostante e, in questo modo, risulta impossibile utilizzare il robot in modo autonomo. Il moto del manipolatore viene controllato passo dopo passo dall'utente con la conseguente diminuzione di prestazioni soprattutto dal punto di vista della precisione e della velocitá. In parallelo allo sviluppo di carrozzelle elettriche su cui montare i robot manipolatori, ma comunque sempre pilotate da un utente, la ricerca nel campo della robotica ha dedicato numerosi sforzi anche per studiare lo sviluppo di carrozzelle "intelligenti". Tra i principali vantaggi di questo tipo di carrozzelle vanno rilevati quelli offerti dal ricorso a sensori capaci di aiutare l'utente nel movimento oppure di muoversi autonomamente, anche in presenza di ostacoli. Come evidenziato dagli esempi precedenti, i manipolatori utilizzati su carrozzelle, pur essendo meno precisi a causa del moto relativo derivante dal movimento della base, godono di tre gradi di libertá in piú (due di traslazione e uno di rotazione) rispetto ai precedenti, forniti appunto dalla mobilitá della carrozzella. In questo modo risulta possibile per il braccio rag-

giungere uno spazio di lavoro maggiore e, di conseguenza, ci si puó affidare a robot manipolatori di dimensioni minori. Esistono comunque alcune limitazioni: in questo caso, infatti, si deve cercare di mantenere l'ingombro creato dal manipolatore il piú limitato possibile, per evitare che ci siano problemi in ambienti chiusi e stretti (ad esempio le porte o i corridoi in casa). Infine, la soluzione che offre maggiore flessibilitá nell'ambito della robotica per assistenza é sicuramente rappresentata dai *robot mobili autonomi* equipaggiati con un manipolatore e con sistemi di sensori per eseguire compiti di manipolazione e trasporto di oggetti. Questo sistema appare la soluzione piú promettente per tutti quegli utenti che sono affetti da gravi disabilitá fisiche o che devono passare la maggior parte del loro tempo sdraiati su un letto. Fornendo ad essi un'interfaccia adeguata, risulta possibile per l'utente interagire con il sistema anche a livello vocale. Questo tipo di sistema supera i problemi imposti dalla configurazione a carrozzella, ma rappresenta ovviamente la soluzione dotata di maggior complessitá. Essa si deve tuttavia scontrare con problemi tecnici tutt'ora irrisolti soprattutto nell'ambito dell'assistenza a persone con disabilitá, dove la precisione del sistema, come si puó ben immaginare, risulta fortemente critica.

Capitolo 2

Compendi Teorici

In questo capitolo saranno descritti i principali risultati teorici necessari alla descrizione di un manipolatore antropomorfo, tratti principalmente dal testo di riferimento per la disciplina della robotica, citato in bibliografia [13].
Un *mapilatore* é per definizione una catena di *corpi rigidi*, detti *bracci*, connessi tra loro mediante *giunti* che conferiscono i *gradi di libertá* alla catena, di cui un estremo é connesso ad una *base* dove viene fatto coincidere un riferimento assoluto, e l'altro ad un *organo terminale* per la presa degli oggetti. I giunti considerati sono solo di due categorie: *prismatici*, che consentono la traslazione lungo l'asse di giunto e *rotoidaili* che consentono la rotazione lungo tale asse. Il moto complessivo della struttura é realizzato mediante la composizione di moti elementari di ogni braccio rispetto al precedente. Per poter manipolare un oggetto nello spazio viene richiesta la descrizione di *posizione* e *orientamento* dell'organo terminale. Tale descrizione puó avvenire in primo stadio con un approccio basato sulla **cinematica**, pertanto, mediante l'algebra matriciale, é possibile ricavare l'equazione della *cinematica diretta*, che consente di descrivere posizione e orientamento dell'organo terminale nello spazio in funzione delle *variabili di giunto*. La descrizione della configurazione di un braccio manipolatore dunque é strettamente legata alla sua cinematica diretta, che a sua volta é legata alla convenzione usata per ricavarla: in questo ambito dunque saranno approfondite le *trasformazioni omogenee* e la convenzione adottata per il calcolo di queste, cioé la convenzione di *Denavit-Hartenberg*. Inoltre si analizzano anche i legami tra le velocitá dell'organo terminale e quelle di ogni relativo giunto, entrando cosí nello studio della **cinematica differenziale**, il cui principale strumento di analisi é lo *Jacobiano*, utilizzato per lo studio delle *singolaritá cinematiche* e di centrale importanza negli algoritmi di *inversione cinematica*, dove si studiano i legami tra *spazio operativo*, e *spazio di lavoro*. Inoltre si analizzeranno alcune *tecniche di interpolazione* usate per generare i riferimenti di

traiettoria, cioé quegli ingressi che una volta tradotti in leggi orarie in spazio operativo devono poter garantire l'inseguimento della traiettoria pianificata con una buona fedeltá.

2.1 Matrici di rotazione

Un *punto materiale* é completamente descritto nello spazio in termini delle sue coordinate rispetto ad una terna di riferimento ortonormale *O-xyz*, dove **x**, **y**, **z** siano i versori degli assi[1]:

$$^0\mathbf{P} = p_x\mathbf{x} + p_y\mathbf{y} + p_z\mathbf{z}$$

Si consideri un *corpo rigido* Ω come un insieme di punti materiali nello spazio, tutti riferiti ad un altro riferimento *O'-x'y'z'*, detto *solidale al corpo*, di centro *O'*. Ogni punto riferito a questo riferimento non cambia mai le sue coordinate, proprio perché solidali con tale sistema, mentre le sue coordinate cambiano se espresse rispetto ad un riferimento assoluto. Dunque per un punto $P \in \Omega$ dal momento che $(P-O) = (P-O') + (O'-O)$, dove rispettivamente *O,O',P* sono il centro del riferimento assoluto, il centro del riferimento solidale, e il punto P, si ha:

$$^0\mathbf{P} =^1 \mathbf{P} + (O' - O) \tag{2.1}$$

$$p_x\mathbf{x} + p_y\mathbf{y} + p_z\mathbf{z} = p'_x\mathbf{x}' + p'_y\mathbf{y}' + p'_z\mathbf{z}' + o'_x\mathbf{x} + o'_y\mathbf{y} + o'_z\mathbf{z} \tag{2.2}$$

dove con 1 si é indicata la terna solidale a Ω. Moltiplicando scalarmente la precedente per i versori del riferimento assoluto é possibile esprimere singolarmente i valori di ogni componente nel riferimento assoluto rispetto a quello solidale, ricordando che $\mathbf{x}\cdot\mathbf{x} = 1$, $\mathbf{x}\cdot\mathbf{y} = 0$ e $\mathbf{x}\cdot\mathbf{z} = 0$, per ortogonalitá dei versori del riferimento assoluto. Si ha:

$$p_x = p'_x(\mathbf{x}'\cdot\mathbf{x}) + p'_y(\mathbf{y}'\cdot\mathbf{x}) + p'_z(\mathbf{z}'\cdot\mathbf{x}) + o'_x \tag{2.3}$$

$$p_y = p'_x(\mathbf{x}'\cdot\mathbf{y}) + p'_y(\mathbf{y}'\cdot\mathbf{y}) + p'_z(\mathbf{z}'\cdot\mathbf{y}) + o'_y \tag{2.4}$$

$$p_z = p'_x(\mathbf{x}'\cdot\mathbf{z}) + p'_y(\mathbf{y}'\cdot\mathbf{z}) + p'_z(\mathbf{z}'\cdot\mathbf{z}) + o'_z \tag{2.5}$$

Si osservi che nelle equazioni precedenti compaiono 9 prodotti scalari piú 3 componenti che indicano le coordinate del punto *O'* rispetto il riferimento assoluto. I nove prodotti si possono riassumere in una matrice, che in algebra lineare pren-

[1]Nella formula seguente, il pedice a sinistra indica il riferimento assoluto che si é definito, indicato convenzionalmente con '0'. Ogni riferimento definito in seguito verrá denominato con i numeri naturali successivi.

de il nome di *matrice di cambiamento di base*, portando una notazione compatta delle equazioni precedenti:

$$ {}^0\mathbf{P} = (\mathbf{R}_1^0){}^1\mathbf{P} + {}^0\mathbf{O}' \tag{2.6} $$

La matrice $\mathbf{R}_1^0$ in questo ambito prende il nome di *matrice di rotazione* e la (2.6) é lo strumento che permette di descrivere un generico *movimento rigido*, definito come un movimento che non causa all'oggetto nessuna *deformazione*, e descrivibile come la composizione di rotazioni e traslazioni. In particolare le rotazioni rigide vanno distinte dalle rotazioni applicate al singolo punto materiale dello spazio: infatti queste si possono vedere come trasformazioni lineari di punti dello spazio, mentre per i corpi rigidi ció che cambia é il riferimento solidale (e con esso tutti i punti di Ω); pertanto lo strumento piú adatto per descrivere orientamenti reciproci tra sistemi di riferimento é una matrice di cambiamento di base e non una matrice di trasformazione. Dovendo descrivere rotazioni, i centri dei riferimenti considerati finora devono coincidere, e le (2.2), (2.6) si semplificano in:

$$ p_x\mathbf{x} + p_y\mathbf{y} + p_z\mathbf{z} = p'_x\mathbf{x}' + p'_y\mathbf{y}' + p'_z\mathbf{z}' \tag{2.7} $$

$$ {}^0\mathbf{P} = (\mathbf{R}_1^0){}^1\mathbf{P} \tag{2.8} $$

Le quali confrontate insieme permettono di concludere che la matrice di rotazione contiene lungo le sue colonne i versori del riferimento solidale:

$$ \mathbf{R} = \begin{bmatrix} \mathbf{x}' & \mathbf{y}' & \mathbf{z}' \end{bmatrix} $$

Si osservi anche che é possibile esprimere cambiamenti di coordinate successivi, sempre in funzione di *nuovo* centro di *nuovo* sistema di riferimento solidale a Ω, semplicemente osservando che:

$$ \begin{aligned} {}^0\mathbf{P} &= (\mathbf{R}_1^0){}^1\mathbf{P} \\ {}^1\mathbf{P} &= (\mathbf{R}_2^1){}^2\mathbf{P} \\ &\vdots \\ {}^n\mathbf{P} &= (\mathbf{R}_n^{n-1})^{n-1}\mathbf{P} \end{aligned} \tag{2.9} $$

con una sostituzione a ritroso si ottiene l'espressione che permette di esprimere il punto $\mathbf{P}$ in funzione delle coordinate (per questo dette anche *controvarianti*) nel sistema di riferimento corrente all'indice n, cioé:

$$ {}^0\mathbf{P} = (\mathbf{R}_1^0 \cdots \mathbf{R}_{n-1}^{n-2}\mathbf{R}_n^{n-1})^n\mathbf{P} \tag{2.10} $$

La regola che ne discende é: *la matrice di rotazione equivalente a n rotazioni composte si ottiene* ***postmoltiplicando*** *ogni singola matrice di rotazione alla prima.* Un'altra proprietá notevole per la matrice di rotazione é la seguente: *la matrice di rotazione é una matrice ortogonale.* Si ha infatti:

$$\mathbf{R}^T\mathbf{R} = \begin{bmatrix} \mathbf{x}'^T \\ \mathbf{y}'^T \\ \mathbf{z}'^T \end{bmatrix} \begin{bmatrix} \mathbf{x}' & \mathbf{y}' & \mathbf{z}' \end{bmatrix} = \begin{bmatrix} \mathbf{x}'^T\mathbf{x}' & \mathbf{x}'^T\mathbf{y}' & \mathbf{x}'^T\mathbf{z}' \\ \mathbf{y}'^T\mathbf{x}' & \mathbf{y}'^T\mathbf{y}' & \mathbf{y}'^T\mathbf{z}' \\ \mathbf{z}'^T\mathbf{x}' & \mathbf{z}'^T\mathbf{y}' & \mathbf{z}'^T\mathbf{z}' \end{bmatrix} = \mathbf{I}$$

ricordando che tali versori per definizione sono ortonormali tra loro. Da questa proprietá discende che:

$$\mathbf{R}^T = \mathbf{R}^{-1} \tag{2.11}$$

che dimostra l'ortogonalitá di $\mathbf{R}$. Da quanto detto in precedenza si puó definire la matrice di rotazione nel seguente modo: *movimento rigido avente come punti fissi un punto detto centro (in due dimensioni), o una retta detta asse di rotazione (in tre dimensioni). Questo movimento sposta tutti i punti intorno al centro, o asse, di un angolo fissato che ne cambia l'orientamento* [18]. Fissato un angolo θ, eventualmente dipendente dal tempo secondo una determinata legge oraria, e considerando come assi principali di rotazione le direzioni dei versori di un sistema di riferimento ortonormale, sviluppando alcuni calcoli trigonometrici, le matrici di rotazioni *fondamentali* si possono scrivere come:

$$\mathbf{R}_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \tag{2.12}$$

$$\mathbf{R}_y = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{2.13}$$

$$\mathbf{R}_z = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.14}$$

Con queste matrici di rotazione fondamentali é possibile descrivere qualsiasi posa ottenibile da piú rotazioni consecutive con la (2.10). Le matrici di rotazione forniscono una descrizione ridondante dell'orientamento di una terna; esse sono infatti caratterizzate da nove elementi che non sono indipendenti, ma legati tra di loro da sei vincoli, dovuti alle condizioni di ortogonalitá, espresse dalla (2.11). Dunque sono sufficienti tre parametri indipendenti per rappresentare un qualsiasi orientamento nello spazio, usando una terna di angoli che costituiscono

una *rappresentazione minima*. Questi tre angoli, riferiti ad assi principali, devono essere tali che non vi siano rotazioni consecutive ad assi paralleli, e vengono chiamati *angoli di Eulero*, di cui esistono 12 versioni, dette *convenzioni*. Un *polso sferico* (presente anche nella configurazione del MANUS) riproduce una di queste terne di Eulero, precisamente la *convenzione ZYZ*, ottenibile dalla composizione delle seguenti rotazioni elementari:

- si ruota la terna origine dell'angolo φ intorno all'asse z: tale rotazione é descritta della matrice di rotazione $\mathbf{R}_z(\varphi)$.
- si ruota la terna dell'angolo θ intorno all'asse y' corrente: tale rotazione é descritta dalla rotazione $\mathbf{R}_{y'}(\theta)$.
- si ruota la terna dell'angolo ψ intorno all'asse z'' corrente: tale rotazione é descritta dalla rotazione $\mathbf{R}_{z''}(\psi)$

L'orientamento finale della terna che si ricava con la composizione di rotazioni definite rispetto alla terna corrente, é caratterizzato dalla matrice di rotazione ottenuta moltiplicando da sinistra verso destra le matrici rappresentative delle rotazioni elementari effettuate:

$$\mathbf{R} = \mathbf{R}_z(\varphi)\mathbf{R}_{y'}(\theta)\mathbf{R}_{z''}(\psi) =$$

$$\begin{bmatrix} \cos\varphi\cos\theta\cos\psi - \sin\varphi\sin\psi & -\cos\varphi\cos\theta\sin\psi - \sin\varphi\cos\psi & \cos\varphi\sin\theta \\ \sin\varphi\cos\theta\cos\psi + \cos\varphi\sin\psi & -\sin\varphi\cos\theta\sin\psi + \cos\varphi\cos\psi & \sin\varphi\sin\theta \\ -\sin\theta\cos\psi & \sin\theta\sin\psi & \cos\theta \end{bmatrix}$$

Una rappresentazione minima dell'orientamento é ricca di conseguenze: oltre ad usare un minor numero di giunti per ottenere una qualsiasi orientazione, con questa scelta é possibile studiare la cinematica diretta scomponendo il problema in due, cioé prima ricercando la configurazione della struttura portante, poi quella del polso sferico. Un altro vantaggio derivante da questa scelta é descritto nei prossimi paragrafi.

2.1.1 Trasformazioni omogenee e catena aperta

Le trasformazioni omogenee consentono di compattare due informazioni disgiunte: la posizione del centro del riferimento solidale al corpo e la sua orientazione rispetto al riferimento assoluto, incluse nell'equazione (2.6). Allo scopo di ottenere una rappresentazione compatta del legame esistente tra le rappresentazioni delle coordinate di uno stesso punto rispetto due terne differenti, si puó introdurre la *rappresentazione in coordinate omogenee* di un vettore, ponendo semplicemente:

$$\tilde{\mathbf{P}} = \begin{bmatrix} \mathbf{P} \\ 1 \end{bmatrix}$$

Con questa scelta é possibile scrivere una relazione del tipo (2.8) come legame diretto tra il punto espresso rispetto al riferimento assoluto e quello espresso nel sistema di riferimento solidale. Le componenti dei vettori salgono a 4, e la matrice di trasformazione, detta appunto *matrice omogenea*, é una (4×4), e diviene:

$$\mathbf{A}_1^0 = \begin{bmatrix} \mathbf{R}_1^0 & {}^0\mathbf{O}' \\ \mathbf{0}^T & 1 \end{bmatrix}$$

il legame diretto si riscrive come:

$${}^0\tilde{\mathbf{P}} = (\mathbf{A}_1^0)^1\tilde{\mathbf{P}} \tag{2.15}$$

con un po' di algebra delle matrici a blocchi considerando queste nuove quantitá, espresse in *coordinate omogenee*, si puó ritornare alla notazione iniziale. La matrice di trasformazione omogenea eredita naturalmente dalla matrice di rotazione tutte le proprietá finora descritte (eccetto quella di essere ortogonale).

A questo punto si hanno tutti gli elementi necessari per studiare i manipolatori: questi si possono considerare come catena di corpi rigidi, collegati tra loro tramite giunti. Si consideri un manipolatore a *catena aperta* costituito da $n+1$ bracci connessi tramite n giunti, ove il braccio 0 é convenzionalmente fisso a terra. Si assuma che ogni giunto fornisca un *grado di libertá* alla struttura meccanica, corrispondente alla *variabile di giunto*. La costruzione di una procedura operativa per il computo della *cinematica diretta* scaturisce naturalmente dalla struttura a catena aperta del manipolatore. Infatti, dal momento che ciascun giunto connette due e solo due bracci consecutivi, é ragionevole considerare dapprima isolatamente il problema della descrizione dei legami cinematici tra bracci consecutivi e successivamente risolvere in maniera ricorsiva il problema della descrizione complessiva della cinematica del manipolatore. A tale scopo é opportuno definire una terna di coordinate solidale a ciascun braccio, dal braccio 0 al braccio n, e definire tra due terne consecutive il nuovo orientamento, conferito dal giunto che connette i due corpi. Indicando con $\mathbf{T}_0^b$ e $\mathbf{T}_e^n$, (indipendenti da variabili di giunto) le matrici di cambiamento di base che rispettivamente permettono di passare dalla base al primo braccio, e dall'ultimo braccio all'organo terminale, si puó scrivere, ricordando la (2.15) e iterandola n volte sostituendo la precedente:

$${}^0\tilde{\mathbf{P}} = (\mathbf{T}_0^b\mathbf{A}_1^0(q_1)\mathbf{A}_2^1(q_2)\cdots\mathbf{A}_n^{n-1}(q_n)\mathbf{T}_e^n)^e\tilde{\mathbf{P}} \tag{2.16}$$

dove la quantitá ${}^{e}\tilde{\mathbf{P}}$ é costante perché solidale con l'ultimo riferimento. La trasformazione che descrive la rotazione dell'organo terminale rispetto alla terna base e in funzione delle variabili di giunto é la seguente quantitá:

$$\mathbf{T}_0^n(\mathbf{q}) = \mathbf{A}_0^1(q_1)\mathbf{A}_1^2(q_2)\cdots\mathbf{A}_{n-1}^n(q_n)$$

dove si puó osservare che ogni trasformazione omogenea apporta un *solo* grado di libertá, conferito dalla variabile di giunto. In quest'ultima sono raggruppate tutte le informazioni necessarie per definire posizionamento e orientamento dell'organo terminale dato un vettore di (dimensione n), contenente il valore delle variabili di giunto. É dunque possibile definire la *cinematica diretta* semplicemente analizzando i vari blocchi della matrice di trasformazione omogenea.

2.1.2 Convenzione di Denavit-Hartemberg

La convenzione di Denavit-Hartenberg fornisce un metodo generale e sistematico per esprimere la posizione e l'orientazione relativi di due bracci consecutivi, ed é dunque un metodo standard per ricavare la matrice omogenea $\mathbf{T}_n^0(\mathbf{q})$ per il computo della cinematica diretta di un manipolatore.

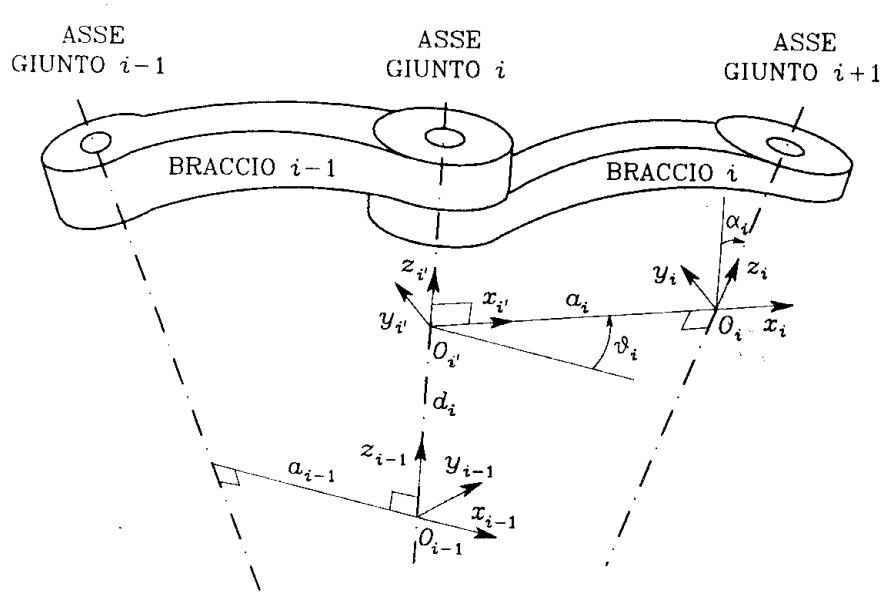

Figura 2.1: Convenzione per due bracci consecutivi.

Considerando come *asse di giunto* quell'asse lungo cui ha luogo il moto, con riferimento alla figura 2.1, si opera nel seguente modo:

- si sceglie l'asse z_i giacente lungo l'asse del giunto $i+1$;

- si individua O_i all'intersezione dell'asse z_i con la normale comune agli assi z_{i-1} e z_i, e con O'_i si indica l'intersezione della normale comune con z_{i-1};
- si sceglie l'asse x_i diretto lungo la normale comune agli assi z_{i-1} e z_i con verso positivo dal giunto i al giunto $i+1$;
- si sceglie l'asse y_i in modo da completare una terna levogira.

Nei seguenti casi la convenzione di Denavit-Hartenberg non fornisce una definizione univoca della terna:

- nella *terna 0* solo la direzione dell'asse z_0 risulta specificata, O_0 e x_0 possono essere scelti arbitrariamente;
- non essendovi un giunto $n+1$ nella *terna n* l'asse z_n non é univocamente definito, mentre l'asse x_n deve essere normale all'asse z_{n-1};
- quando due assi consecutivi sono paralleli, in quanto la normale comune tra di loro non é univocamente definita;
- quando due assi consecutivi si intersecano, in quanto il verso di x_i é arbitrario;
- quando il giunto i é prismatico, in questo caso l'unica direzione determinata é quella dell'asse z_i.

Nei casi di indeterminazione si puó semplificare la procedura ricercando ad esempio delle condizioni di allineamento tra gli assi delle terne. A questo punto la posizione e l'orientazione della terna i rispetto alla terna $i-1$ risultano completamente definite dai seguenti parametri, detti **parametri di Denavit-Hartenberg**:

- a_i distanza di O_i da O'_i;
- d_i coordinata su z_{i-1} di O'_i;
- α_i angolo intorno all'asse x_i tra l'asse z_{i-1} e l'asse z_i valutato positivo in senso antiorario;
- θ_i angolo intorno all'asse z_{i-1} tra l'asse x_{i-1} e l'asse x_i valutato positivo in senso antiorario.

dove a_i e α_i sono sempre *costanti* e dipendono solamente dalla geometria di connessione dei bracci consecutivi tramite i giunti. Tra d_i e θ_i solo uno é variabile in base al tipo di giunto utilizzato nella connessione dei bracci i e $i-1$; in particolare:

- se il giunto é rotoidale la variabile é θ_i;
- se il giunto é prismatico la variabile é di d_i;

A questo punto si puó ricavare la trasformazione di coordinate che lega la terna i ed $i-1$, secondo i seguenti passi:

- si parte da una terna coincidente con la terna $i-1$, per arrivare alla terna i passando per una terna intermedia i';
- si trasla la terna di d_i lungo l'asse z_{i-1} e la si ruota di θ_i attorno allo stesso asse; questa operazione la porta a sovrapporsi alla terna i' ed é descritta dalla trasformazione omogenea:

$$\mathbf{A}_{i'}^{i-1} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & 0 \\ \sin\theta_i & \cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

  dove si puó notare il blocco della matrice di rotazione come rotazione elementare lungo l'asse z_{i-1} e nel blocco del vettore di posizione, una traslazione lungo il medesimo asse.

- si trasla ora la terna ottenuta nel passaggio precedente di a_i lungo l'asse $x_{i'}$ e la si ruota di α_i attorno allo stesso asse portando alla sovrapposizione fra la terna in questione e la terna i come rappresentato dalla matrice di trasformazione omogenea:

$$\mathbf{A}_{i}^{i'} = \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & \cos\alpha_i & -\sin\alpha_i & 0 \\ 0 & \sin\alpha_i & \cos\theta_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- la trasformazione di coordinate complessiva si ottiene moltiplicando le singole trasformazioni come segue:

$$\mathbf{A}_{i}^{i-1}(q_i) = \mathbf{A}_{i'}^{i-1}\mathbf{A}_{i}^{i'} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\theta_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

  dove con q_i si assume la variabile di giunto, identificata con θ_i se il giunto é rotoidale, con d_i se prismatico.

Si puó riassumere la procedura appena vista basata sulla convenzione di Denavit-Hartenberg nel seguente algoritmo che consente la derivazione della funzione cinematica diretta per qualsiasi manipolatore a catena aperta.

1. Individuare e numerare consecutivamente gli assi dei giunti ed assegnare rispettivamente le direzioni agli assi $z_0, \ldots, z_n$.

2. Posizionare la terna 0 sull'asse z_0 e scegliere x_0 e y_0 in modo da formare una terna levogira. Eseguire i passi 3, 4 e 5 per $i = 1, \ldots, n-1$:

3. Individuare l'origine O_i all'intersezione di z_i con la normale comune agli assi z_{i-1} e z_i. Se gli assi in questione sono paralleli ed il giunto i é rotoidale, posizionare O_i in modo da annullare d_i; Se invece il giunto é prismatico sceglier O_i in corrispondenza di una posizione di riferimento della corsa del giunto.

4. Fissare l'asse x_i diretto lungo la normale comune agli assi z_{i-1} e z_i con verso positivo dal giunto i al giunto $i+1$.

5. Fissare y_i in modo da ottenere una terna levogira.

6. Fissare la terna n allineando z_n lungo la direzione di z_{n-1} se il giunto é rotoidale, scegliere invece z_n in modo arbitrario nel caso di giunto prismatico.

7. Costruire per ciascun giunto la tabella dei parametri a_i, d_i, α_i, θ_i e calcolare sulla base di questi parametri le matrici di trasformazione omogenea $\mathbf{A}_i^{i-1}(q_i)$.

Una volta assegnate la terna di base $\mathbf{T}_0^b$ e la terna dell'organo terminale $\mathbf{T}_e^n$ si hanno tutti gli elementi per il corretto calcolo della funzione di cinematica diretta studiando i blocchi della matrice omogenea:

$$\mathbf{T}(\mathbf{q}) = \mathbf{T}_0^b \mathbf{A}_0^1(q_1)\mathbf{A}_1^2(q_2)\cdots\mathbf{A}_{n-1}^n(q_n)\mathbf{T}_n^e \tag{2.17}$$

Per quanto riguarda gli aspetti computazionali della cinematica diretta, il carico maggiore deriva dal calcolo delle funzioni trascendenti, che tra l'altro conferiscono una natura non lineare alle equazioni del modello. Infine per il calcolo del blocco 3×3 della matrice di rotazione, dato che contiene i versori del sistema corrente di cui ne descrive l'orientamento, puó essere calcolata determinando due versori della terna dell'organo terminale e calcolando il terzo come prodotto vettoriale dei due.

2.2 Algoritmi di inversione

In questo paragrafo si introduce la *cinematica differenziale* che caratterizza i legami esistenti tra le *variabili di giunto* e la *posa* dell'organo terminale. Ció perché spesso si preferisce gestire il controllo dei manipolatori nello *spazio operativo* (definito come il sottospazio vettoriale di dimensione pari al numero di gradi di libertá del manipolatore, contenente tutte le *configurazioni* possibili che conferiscono una determinata posa al manipolatore) anziché nello *spazio di lavoro*, che é definito come la porzione di spazio tridimensionale raggiungibile da un robot manipolatore. Lo strumento attraverso cui si affronta il problema *dell'inversione cinematica* per la determinazione delle legge orarie da imporre ai vari giunti affinché l'organo terminale segua una certa traiettoria nello spazio di lavoro é lo *Jacobiano*, cioé la matrice di trasformazione che lega le velocitá lineari e angolari dell'organo terminale alle singole velocitá di giunto.

2.2.1 Jacobiano

Si consideri un manipolatore a n gradi di libertá. Nei precedenti paragrafi si é visto come é possibile costruire la matrice di trasformazione omogenea che contiene tutte le informazioni su posizione e orientamento dell'organo terminale, nella forma:

$$\mathbf{T}_e(\mathbf{q}) = \begin{bmatrix} \mathbf{R}_e(\mathbf{q}) & \mathbf{p}_e(\mathbf{q}) \\ \mathbf{0}^T & 1 \end{bmatrix}$$

in cui $\mathbf{q}$ indica il vettore delle variabili di giunto. Al variare di queste, variano sia la posizione che l'orientamento dell'organo terminale del manipolatore. L'obiettivo é esprimere in funzione di queste variazioni la velocitá lineare e angolare dell'organo terminale. Queste sono esprimibili rispettivamente in funzione di $\mathbf{p}_e(\mathbf{q})$ e $\mathbf{R}_e(\mathbf{q})$. Infatti ricordando il significato fisico della velocitá lineare come derivata temporale del vettore posizione del riferimento mobile e la velocitá angolare come derivata del cambiamento dell'orientazione dei versori della terna solidale rispetto la terna base [2], si ha:

$$^{(0)}\mathbf{v}_e = \dot{\mathbf{p}}_e(\mathbf{q}) \tag{2.18}$$

$$\mathbf{\Psi}(^{(0)}\boldsymbol{\omega}_e) = \dot{\mathbf{R}}_e\mathbf{R}_e^T \tag{2.19}$$

dove gli apici a sinistra indicano grandezze vettoriali espresse rispetto alla terna di riferimento assoluta, indicata con 0, e la lettera $\mathbf{\Psi}$ indica la *matrice assiale* che contiene al di fuori della diagonale (é antisimmetrica) le componenti della velocitá angolare $\boldsymbol{\omega}_e$. La (2.18) permette di giungere subito a un risultato:

[2] Si veda il teorema di Poisson, che permette di definire la velocitá angolare come $\dot{\mathbf{e}}' = \boldsymbol{\omega} \times \mathbf{e}'$.

ricordando che la derivata di una funzione vettoriale a piú variabili si ottiene dal prodotto scalare del gradiente per il vettore contenente le derivate delle variabili indipendenti rispetto alla variabile temporale t, si puó scrivere:

$$\mathbf{v}_e = \nabla \mathbf{p}_e \cdot \dot{\mathbf{q}} \tag{2.20}$$

Ricordando che il gradiente di una funzione vettoriale é una matrice, si puó definire il legame differenziale tra velocitá lineare e velocitá dei giunti come il seguente prodotto di matrici (che sostituisce il prodotto scalare):

$$\mathbf{v}_e = \mathbf{J}_p(\mathbf{q})\dot{\mathbf{q}} \tag{2.21}$$

$$\text{dove} \quad \mathbf{J}_p = \nabla \mathbf{p}_e = \begin{bmatrix} \frac{\partial p_{e,1}}{\partial q_1} & \cdots & \frac{\partial p_{e,1}}{\partial q_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial p_{e,n}}{\partial q_1} & \cdots & \frac{\partial p_{e,n}}{\partial q_n} \end{bmatrix}$$

Analogamente il calcolo dello Jacobiano riferito alla velocitá angolare si esprime (sviluppando calcoli laboriosi) rispetto al vettore $\mathbf{q}$ con la stessa forma della (2.21):

$$\boldsymbol{\omega}_e = \mathbf{J}_o(\mathbf{q})\dot{\mathbf{q}} \tag{2.22}$$

La notazione puó essere resa piú compatta ponendo:

$$\begin{bmatrix} \mathbf{v}_e \\ \boldsymbol{\omega}_e \end{bmatrix} = \begin{bmatrix} \mathbf{J}_p \\ \mathbf{J}_o \end{bmatrix} \dot{\mathbf{q}} \tag{2.23}$$

Si osservi che le componenti necessarie a descrivere la posizione dell'organo terminale non concorrono anche all'orientazione e viceversa, pertanto vi sono delle derivate parziali identicamente nulle. In particolare indicando con $\hat{\boldsymbol{\theta}}$ il vettore delle componenti che concorrono al raggiungimento della posizione desiderata e con $\tilde{\boldsymbol{\theta}}$ il vettore delle componenti che concorrono invece all'orientamento, si possono estrarre dalle matrici i blocchi $\mathbf{J}_p$, $\mathbf{J}_o$ con componenti non nulle, come segue:

$$\begin{bmatrix} \mathbf{v}_e \\ \boldsymbol{\omega}_e \end{bmatrix} = \begin{bmatrix} \mathbf{J}_p & \mathbf{O} \\ \mathbf{O} & \mathbf{J}_o \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{\theta}} \\ \tilde{\boldsymbol{\theta}} \end{bmatrix}$$

Indicando con $\boldsymbol{\nu}$, $\mathbf{J}$ e $\mathbf{q}$ le quantitá matriciali e vettoriali espresse nella precedente relazione, si ottiene una forma ancora piú compatta, ossia:

$$\boldsymbol{\nu} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}} \tag{2.24}$$

dove la matrice a blocchi precedente contiene blocchi $(3 \times p)$ per la posizione, con p che indica il numero di parametri usati per la descrizione della posizione

e $(3 \times m)$ per l'orientamento, con m numero di parametri per l'orientamento $(p + m = DOF)$. Con ció si é ottenuta la relazione diretta ricercata tra le velocitá nello spazio di lavoro e le velocitá di rotazione dei singoli giunti nello spazio dei giunti: questa rappresenta un vantaggio rispetto la (2.16), in quanto si é definito un legame diretto tra le grandezze sopracitate. Nonostante ció il procedimento usato per il suo calcolo non é univocamente determinato: sfruttando la cinematica dei corpi rigidi é infatti possibile sviluppare una costruzione algoritmica dello Jacobiano procedendo braccio per braccio con risultati diversi, per quanto riguarda il blocco $\mathbf{J}_o$, da quelli finora visti. Cosí si ottiene il cosidetto *Jacobiano geometrico* che differisce da quello definito in questo paragrafo, detto a buon diritto *Jacobiano analitico*: l'uno riveste una maggiore importanza quando si vuole esprimere una grandezza fisica del manipolatore partendo dalla sua configurazione fisico-geometrica, mentre l'altro é piú utile quando si vuole istituire il legame diretto tra spazio operativo e di lavoro e dunque riveste una notevole importanza nella risoluzione del problema inverso, mediante gli algoritmi di inversione. Inoltre, lo Jacobiano risulta lo strumento per determinare le *configurazioni singolari*: queste sono le pose per cui si ha una perdita di gradi di mobilitá da parte del manipolatore. Infatti, a paritá di velocitá assegnata nell'equazione (2.24), nelle configurazioni in cui si ha una perditá di mobilitá, il moto é concentrato in un minor numero di giunti, che richiedono una maggiore attuazione: ció porta spesso a velocitá molto elevate nello spazio operativo, che spesso superano le tolleranze ammesse. É pertanto vitale conoscere e studiare tali configurazioni, in modo che l'organo terminale non si trovi mai in un loro intorno per i motivi di cui sopra. Analiticamente, una configurazione singolare é quella per cui il rango dello Jacobiano non é piú pieno. Ció si traduce semplicemente nel discutere la seguente:

$$\det(\mathbf{J}(\mathbf{q})) = 0$$

che si annullerá in corrispondenza di determinate configurazioni ai giunti del manipolatore.

2.2.2 Algoritmi di inversione cinematica

Gli algoritmi di inversione si basano sulla conoscenza dello Jacobiano, che come abbiamo visto é dipendente dalla posa (cioé da $\hat{\boldsymbol{\theta}}$ e $\tilde{\boldsymbol{\theta}}$) e descrive il contributo delle velocitá di ogni giunto a quella lineare e angolare dell'organo terminale. Assumendo che il problema sia ben posto (ossia la traiettoria non passi nell'intorno di punti in cui si hanno pose singolari) e che l'inversione dello Jacobiano sia dunque computabile, i valori delle variabili di giunto ad ogni iterazione sono

dati, per differenziazione discreta della (2.24), da:

$$\mathbf{q}(t_{k+1}) = \mathbf{q}(t_k) + \mathbf{J}^{-1}(\mathbf{q}(t_k))\boldsymbol{\nu}(t_k)\Delta t \tag{2.25}$$

$$\text{dove} \quad \Delta t = t_{k+1} - t_k$$

essendo un algoritmo a tempo discreto, l'integrazione é di tipo discreta e ció produce fenomeni di *deriva* della soluzione, ovvero alle variabili di giunto calcolate corrisponde, nello spazio operativo, una posa dell'organo terminale diversa da quella desiderata. É possibile ovviare a tale inconveniente ricorrendo ad uno schema di soluzione che tenga conto dell'*errore* commesso nello spazio di lavoro, e imponendo a questo una dinamica asintoticamente stabile allo zero. Per fare questo si definisca, a tempo continuo, l'errore e la sua derivata temporale come:

$$\mathbf{e} := \mathbf{x}_d - \mathbf{x}_u$$

$$\dot{\mathbf{e}} := \dot{\mathbf{x}}_d - \dot{\mathbf{x}}_u$$

dove $\mathbf{x}_d$ é la posizione desiderata imposta dalla traiettoria da inseguire e $\mathbf{x}_u$ é la posizione calcolata del manipolatore una volta ottenute le variabilili di giunto. Affinché l'errore tenda asintoticamente a zero si deve imporre la seguente equazione differenziale:

$$\dot{\mathbf{e}} + \mathbf{K}\mathbf{e} = \mathbf{0} \tag{2.26}$$

che ha come soluzione un esponenziale tendente allo zero a seconda degli autovalori della *matrice dei guadagni* [3] $\mathbf{K}$, con una velocitá di convergenza che aumenta piú saranno grandi questi autovalori. Mentre le velocitá $\dot{\mathbf{x}}_d$ sono calcolabili per derivazione diretta delle equazioni parametriche che descrivono un cammino nello spazio di lavoro, le velocitá $\dot{\mathbf{x}}_u$ sono calcolabili sfruttando la (2.24). Dunque la derivata dell'errore é altresí esprimibile come:

$$\dot{\mathbf{e}} = \dot{\mathbf{x}}_d - \mathbf{J}(\mathbf{q})\dot{\mathbf{q}} \tag{2.27}$$

Sostituendo la derivata dell'errore cosí espressa nella (2.26) si arriva alla seguente:

$$\dot{\mathbf{x}}_d - \mathbf{J}(\mathbf{q})\dot{\mathbf{q}} = -\mathbf{K}\mathbf{e}$$

da cui, si giunge finalmente alla:

$$\dot{\mathbf{q}} = \mathbf{J}^{-1}(\mathbf{q})(\mathbf{K}\mathbf{e} + \dot{\mathbf{x}}_d) \tag{2.28}$$

[3] che generalmente é diagonale.

effettuando un'integrazione della (2.28) si possono ottenere infine i valori da imporre ai giunti del manipolatore per inseguire la traiettoria desiderata con un errore accettabile. Dunque si ottiene l'algoritmo discretizzando il tutto con un intervallo di campionamento temporale definito da un Δt. In realtá essendo un algoritmo a tempo discreto, non sará sufficiente fornire al manipolatore solamente questo insieme di punti (che generalmente non sará denso per non appesantire la computazione che cresce al diminuire del Δt), ma si dovrá ricorrere a tecniche di interpolazione per fornire un numero di punti di piú denso. Per ogni passo k, si puó tradurre l'equazione (2.28) nel seguente schema a blocchi:

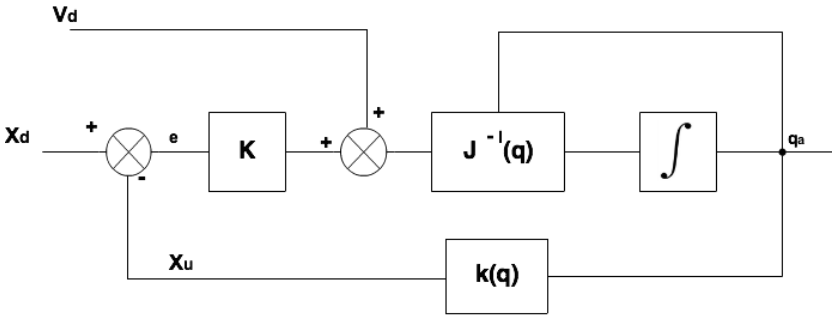

Figura 2.2: Schema a blocchi della (2.28).

dove la funzione $k(\cdot)$ é la funzione di cinematica diretta. Tale schema puó essere rivisitato in termini degli usuali schemi di *controllo in retroazione*: la presenza di un blocco integratore garantisce *astatismo* per riferimenti costanti, e l'azione in avanti fornita dal blocco sommatore con l'apporto di $\dot{\mathbf{x}}_d$ garantisce un *errore a regime* nullo lungo tutta la traiettoria [13].

2.2.3 Algoritmo di inversione per inseguimento di due traiettorie a distanza fissa

Nel capitolo introduttivo si é menzionato che uno degli obiettivi finali di questa tesi é quello di realizzare un modello in simulazione del moto, coordinato e vincolato su una stessa circonferenza, di due manipolatori antropomorfi disposti in posizione frontale. Ció crea la necessitá di definire un algoritmo *ad hoc* in modo che le traiettorie vengano inseguite con una certa fedeltá di asservimento del percorso.
Quando due manipolatori vengono usati in combinazione per realizzare un movimento sincronizzato, si rende necessaria la nozione di *Jacobiano relativo*. Si considerino allora due manipolatori, definendo i seguenti 4 vettori con le relative orientazioni rispetto il riferimento assoluto: P_1 é l'origine della base del manipolatore di sinistra (indicato con la lettera L), P_2 é il punto solidale all'organo

terminale del manipolatore sinistro, P_3 é il punto solidale all'organo terminale del manipolatore destro (indicato con la lettera R), e infine P_4 l'origine della base del manipolatore di destra. Come dimostrato in [7] da Rodrigo S. Jamisola Jr. e Rodney G. Roberts, si ricava la seguente formula, che descrive la derivata del *vettore posizione* del manipolatore destro P_3 rispetto al punto P_2 nei seguenti termini:

$$^2\dot{\mathbf{P}}_3 = \mathbf{\Psi}(^2\mathbf{P}_3)\mathbf{R}_2^1\mathbf{J}_{oA}\dot{\boldsymbol{\theta}}_A + (\mathbf{R}_4^2)^T\mathbf{J}_{pB}\dot{\boldsymbol{\theta}}_B - \mathbf{R}_2^1\mathbf{J}_{pA}\dot{\boldsymbol{\theta}}_A \tag{2.29}$$

dove con la lettera $\mathbf{\Psi}$ si é indicata la *matrice assiale* e le matrici $\mathbf{J}_{oA}$, $\mathbf{J}_{pB}$ e $\mathbf{J}_{pA}$ sono i blocchi dello Jacobiano, che come abbiamo visto é una matrice diagonale a blocchi, contenente lungo la diagonale, nell'ordine, il blocco riferito alle velocitá lineari e quello riferito alle velocitá angolari. Le due matrici $\mathbf{R}_2^1$ e $\mathbf{R}_4^2$ descrivono rispettivamente l'orientamento delle basi dei manipolatori rispetto all'organo terminale del manipolatore di sinistra. Supponendo che nella realizzazione della configurazione dei due manipolatori le basi dei due si trovino ad una distanza fissa lungo una direzione con lo stesso orientamento, e che le terna solidale dell'organo terminale abbia in ogni istante lo stesso orientamento delle rispettive basi, si ha che $\mathbf{R}_2^1 = \mathbf{R}_4^2 = \mathbf{I}$. Inoltre, restringendo la pianificazione della traiettoria in spazio operativo dei due manipolatori solamente ai primi tre giunti della catena cinematica, gli ultimi tre giunti di entrambi i manipolatori manterrano costanti i loro valori, ed in particolare per la configurazione del polso sferico del manipolatore di sinistra, si avrá un contributo del blocco $\mathbf{J}_{oA}$ pari alla matrice nulla. Con queste osservazioni, indicando il vettore posizione $^2\dot{\mathbf{P}}_3$ con la quantitá $\dot{\mathbf{VP}}$, la formula si semplifica nel seguente modo:

$$\dot{\mathbf{VP}} = \mathbf{J}_{pB}\dot{\boldsymbol{\theta}}_B - \mathbf{J}_{pA}\dot{\boldsymbol{\theta}}_A \tag{2.30}$$

Da questa equazione é possibile ricavare i valori da imporre ai primi tre giunti del manipolatore di destra data in ingresso la derivata del vettore posizione determinato da entrambi gli organi terminali in ogni istante per inseguire entrambe le traiettorie. Anche in questo caso occorre ragionare come nel paragrafo precedente, definendo una variabile d'errore che tenga conto del discostamento del valore del vettore posizione desiderato, determinato a partire dalle traiettorie da inseguire, da quello calcolato usando (2.30). Pertanto si defiscono:

$$\mathbf{VP_d} := \mathbf{xl}_d - \mathbf{xr}_d$$

$$\mathbf{e}_{vp} := \mathbf{VP_d} - \mathbf{VP_u}$$

Queste grandezze vengono adottate come indice di controllo dell'errore di posizionamento reciproco, cioé tra la distanza degli organi terminali dei due manipo-

latori, che, dal momento che si richiede di asservire un cammino su circonferenza, deve essere costante. Pertanto la quantitá $\mathbf{e}_{vp}$ é la variabile di errore, che naturalmente deve tendere a zero. Per questa si definisce, in analogia a quanto fatto nel paragrafo precedente, la seguente dinamica secondo l'equazione differenziale:

$$\dot{\mathbf{e}}_{vp} + \mathbf{K}\mathbf{e}_{vp} = \mathbf{0} \tag{2.31}$$

che garantisce un errore a regime nullo, in maniera tanto piú marcata quanto piú elevati saranno gli autovari della matrice $\mathbf{K}$. Similmente a quanto fatto nei paragrafi precendenti, la (2.31) si puó riarrangiare con i seguenti passaggi:

$$\dot{\mathbf{V}}\mathbf{P_d} - \dot{\mathbf{V}}\mathbf{P_u} = -\mathbf{K}\mathbf{e}_{vp}$$

Sostituendo in $\dot{\mathbf{V}}\mathbf{P_u}$ la (2.30):

$$\dot{\mathbf{V}}\mathbf{P_d} - \mathbf{J}_{pB}\dot{\theta}_B + \mathbf{J}_{pA}\dot{\theta}_A = -\mathbf{K}\mathbf{e}_{vp}$$

che riordinando e isolando $\dot{\boldsymbol{\theta}}_B$, identificato come vettore delle variabili di giunto rotoidali da $\dot{\mathbf{q}}_B$, fornisce finalmente:

$$\dot{\mathbf{q}}_B = \mathbf{J}_{pB}^{-1}(\dot{\mathbf{V}}\mathbf{P_d} + \mathbf{J}_{pA}\dot{\mathbf{q}}_A + \mathbf{K}\mathbf{e}_{vp}) \tag{2.32}$$

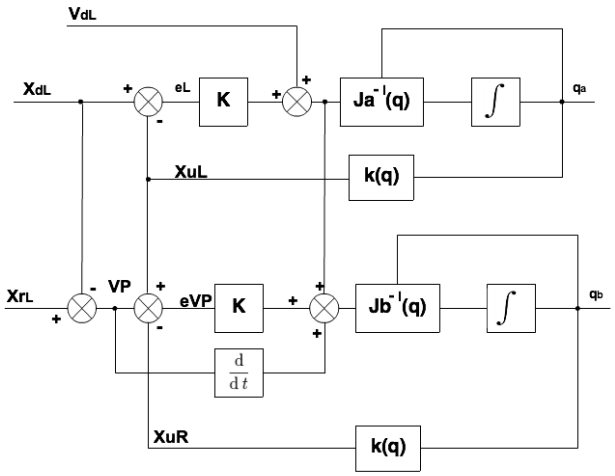

Figura 2.3: Schema a blocchi della (2.28) e della (2.32) combinate.

Effettuando un'integrazione é possibile ottenere i valori da imporre ai giunti del secondo manipolatore affiché segua la traiettoria desiderata. Si osservi che que-

sti valori oltre che dalla quantitá $\dot{\mathbf{V}}\mathbf{P_d}$, calcolabile direttamente una volta note le equazioni parametriche delle traiettorie desiderate, dipendono da $\dot{\mathbf{q}}_A$, cioé dal vettore delle configurazioni ai giunti calcolato usando l'algoritmo descritto nel paragrafo precedente per inseguire la traiettoria del manipolatore di sinistra. Dalla (2.32) é possibile ricavare uno schema a blocchi (fig. 2.3): dal momento che per il calcolo di $\dot{\mathbf{q}}_B$ si usano grandezze dello schema a blocchi trattato nel paragrafo precedente si puó pensare di unire i due in un unico schema, che fornisca come uscite i valori $\dot{\mathbf{q}}_A$ e $\dot{\mathbf{q}}_B$, da imporre rispettivamente al manipolatore di sinistra e al manipolatore di destra. Naturalmente, questo eredita tutte le proprietá del precedente, tra cui *l'anello di retroazione* con la funzione di cinematica diretta per determinare la variabile d'errore e l'*azione in avanti*, fornita in questo caso dalla derivata del vettore di posizione tra le traiettorie desiderate e la quantitá $\mathbf{J}_{pA}\dot{\mathbf{q}}_A$.

2.3 Pianificazione di traiettorie

L'obiettivo della pianificazione di traiettorie é quello di generare gli ingressi di riferimento per il sistema di controllo del moto, che assicuri l'esecuzione da parte del manipolatore delle traiettorie pianificate, a partire da un numero finito di parametri (*punti di raccordo* o *funzionali di costo* da rispettare) utili a caratterizzare la traiettoria desiderata. Qui vengono presentate le tecniche per generare traiettorie nello spazio operativo, distinguendo da una parte il moto *punto-punto*, che prevede il raggiungimento di una posa finale del giunto a partire dalle condizioni iniziali, disinteressandosi del percorso che l'organo terminale traccia nello spazio di lavoro, e dall'altra la generazione di traiettorie a partire da un insieme di punti di raccordo (*via points*) mediante uso di *spline* cubiche [13]. In genere questo insieme di punti di raccordo viene fornito dagli algoritmi di pianificazione del movimento appena visti: questi sono infatti algoritmi *a tempo discreto*, e quindi forniscono un insieme *finito* di punti, che una volta raccordati andranno a costituire ognuna delle leggi in spazio operativo da imporre per la realizzazione di una data traiettoria.
Questa parte riguardante la pianificazione, é solitamente implementata nell'elettronica del manipolatore, mentre per i modelli in simulazione é possibile considerare una delle due possibilitá, in quanto tutta la parte di pianificazione viene effettuata a parte, portando in ingresso al modello il risultato degli algoritmi di inversione e di interpolazione.

2.3.1 Moto punto-punto

Questa tecnica viene presentata per descrivere lo spostamento che un manipolatore puó effettuare tra due punti nel minimo intervallo di campiomento di pacchetti d'informazione che l'elettronica di controllo riesce a decodificare ed inviare ai motori: infatti, assegnata una traiettoria, essa non verrá mai interpretata come grandezza analogica, ma campionata in modo da disporre di un insieme finito di dati. Tra due campioni, il profilo di velocitá é costante, e qualsiasi traiettoria, per quanto complessa sia, viene comunque discretizzata e percorsa come una spezzata di punti [4], [5], [11]. Allora ci si potrebbe chiedere che senso abbia definire una tecnica di interpolazione polinomiale, quando l'asservimento di una traiettoria si puó scomporre in ultima analisi come moto punto-punto. La risposta sta nel fatto che il manipolatore, quando si trova in modalitá di pianificazione (e non in *pilotazione diretta*, ad esempio mediante *joystick* per il MANUS [14]), deve poter disporre comunque di un insieme uniformemente denso di informazioni, ottenuto appunto attraverso l'interpolazione *spline* che collega i punti di passaggio, anche distanti tra loro, fornendo con continuitá l'informazione che sará digitalizzata e inviata dall'elettronica di controllo. Infine un'interpolazione punto-punto puó rivestire un ruolo importante nel momento in cui si vuole raggiungere un punto preciso dello spazio di lavoro senza tener conto delle traiettorie e degli sforzi di controllo. Un'interpolazione punto-punto prevede il raggiungimento di un valore finale, indicato con q_f partendo da un valore iniziale q_0 con una traiettoria che agli estremi é parabolica, con tratto intermedio lineare come si vede in figura:

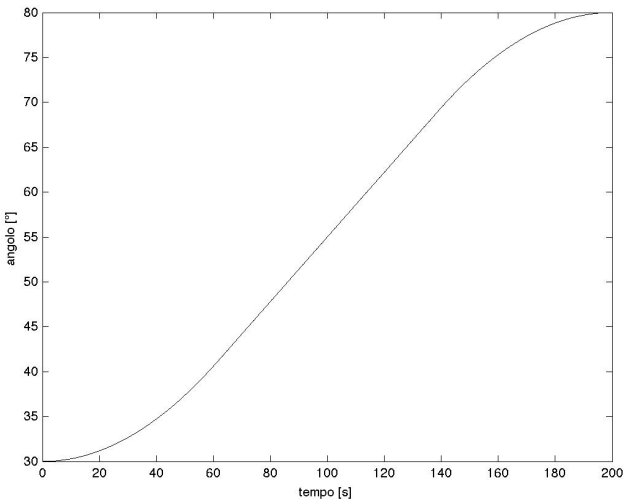

Figura 2.4: Profilo parabolico-lineare.

Il profilo di velocitá, ottenuto per derivazione diretta dei tratti della curva, viene chiamato profilo di velocitá *trapezioidale*, da cui prende anche il nome questo tipo di interpolazione, come si puó vedere nella figura seguente:

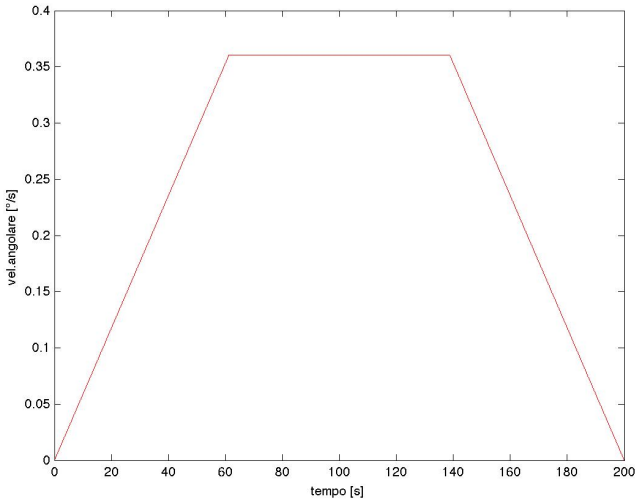

Figura 2.5: Profilo trapezoidale di velocitá.

I tratti lineari costanti si estendono negli intervalli temporali $[0, t_c]$ e $(t_f - t_c, t_f]$, mentre il tratto costante nell'intervallo di mezzo, con t_f che esprime il vincolo temporale in cui deve essere percorsa la curva e t_c l'ampiezza dell'intervallo di tempo in cui si hanno i due tratti parabolici; t_m é il *tempo medio*, cioé la metá di t_f. I valori corrispondenti ad ognuno di questi istanti sono q_c, che é il valore dell'ordinata nel primo punto di raccordo, q_i e q_f, rispettivamente i valori iniziale e finale, e q_m, il valore assunto dalla curva nell'istante t_m, pari alla media tra q_i e q_f. L'equazione che descrive il primo tratto parabolico si esprime semplicemente con la scelta:

$$q(t) = q_i + \frac{1}{2}\ddot{q}_c t^2 \tag{2.33}$$

dove $\ddot{q}_c$ é il coefficiente di concavitá uguale in ogni punto della parabola, e pertanto espressa come la derivata seconda di $q(t)$ nel punto t_c. La velocitá alla fine del tratto parabolico deve equagliare la velocitá (costante) del tratto lineare. Dunque derivando la (2.33) nell'istante t_c in cui si deve avere il raccordo, si ottiene:

$$\frac{dq}{dt}\Big|_{t=tc} = \ddot{q}_c t_c \tag{2.34}$$

questa quantitá deve essere uguale al coefficiente angolare del tratto lineare che si esprime, noti due punti (t_c, q_c) e (t_m, q_m), come:

$$m = \frac{(q_m - q_c)}{(t_m - t_c)} = \frac{(\frac{q_f+q_i}{2} - q_c)}{(\frac{t_f}{2} - t_c)} = \frac{q_f + q_i - 2q_c}{t_f - 2t_c} \tag{2.35}$$

Uguagliando la (2.34) con la (2.35) si ottiene:

$$\ddot{q}_c t_c(t_f - 2t_c) = (q_f + q_i - 2q_c)$$

$$2\ddot{q}_c t_c^2 - \ddot{q}_c t_f t_c + (q_f + q_i) - 2q_c = 0$$

Sostituendo in q_c la (2.33) calcolata per $t = t_c$, si ottiene la seguente equazione di secondo grado rispetto la variabile t_c:

$$\ddot{q}_c t_c^2 - \ddot{q}_c t_f t_c + q_f - q_i = 0 \tag{2.36}$$

che risolta fornisce:

$$t_c = \frac{t_f}{2} \pm \frac{1}{2}\sqrt{\frac{t_f^2\ddot{q}_c - 4(q_f - q_i)}{\ddot{q}_c}} \tag{2.37}$$

dove la soluzione con il segno positivo deve essere esclusa in quanto si otterrebbe un valore di t_c superiore a metá del tempo che si deve impiegare per raggiungere la soluzione finale: ció é tecnicamente errato perché un tratto lineare deve essere sempre presente. Infatti t_c é l'ampiezza che deve essere ricoperta dai tratti parabolici per ottenere una curva interpolante che rispetti le proprietá di continuitá e regolaritá richieste all'inizio. Generalmente $\ddot{q}_c$ é assegnata, con il vincolo che il segno di questa quantitá sia lo stesso di $(q_f - q_i)$ e con il fatto che l'argomento del radicale deve essere maggiore o uguale dello zero, il ché impone:

$$|\ddot{q}_c| \geq \frac{4|q_f - q_i|}{t_f^2} \tag{2.38}$$

Assegnati q_i, q_f, t_f, é possibile assegnare $\ddot{q}_c$ con il vincolo precedente. Si calcola poi il valore dell'ampiezza dei tratti parabolici t_c con la (2.37). La funzione ottenuta é definita per casi nel seguente modo:

$$q(t) = \begin{cases} q_i + \frac{1}{2}\ddot{q}_c t_c^2 & 0 \leq t \leq t_c \\ q_i + \ddot{q}_c t_c(t - \frac{1}{2}t_c) & t_c < t \leq t_f - t_c \\ q_f - \frac{1}{2}\ddot{q}_c(t_f - t)^2 & t_f - t_c < t \leq t_f \end{cases} \tag{2.39}$$

2.3.2 Interpolazione *spline*

L'interpolazione mediante *spline* cubiche é una tecnica per generare polinomi passanti per un insieme di punti, raccordati in corrispondenza di ognuno di questi. In genere con un unico polinomio occorre aumentare il grado all'aumentare dei punti, generando traiettorie con caratteristiche oscillanti molto accentuate (il cosidetto *fenomeno di Runge* [18]). Si preferisce allora costruire una serie di polinomi raccordati tra loro nei punti di passaggio, le *spline* cubiche appunto, cioé polinomi di terzo grado su cui si possono imporre condizioni di continuitá e regolaritá in tali punti.

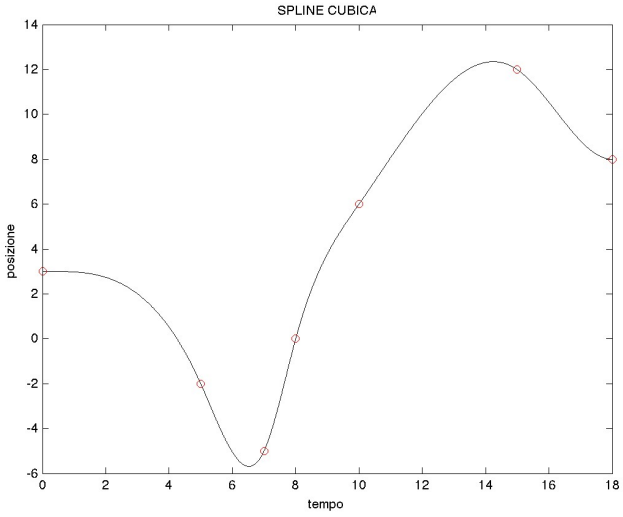

Figura 2.6: Grafico di una spline che interpola un set di punti.

Nella figura 2.6 si puó vedere come i punti siano piú densi in zone che presentano *ostacoli*, mentre si assottigliano laddove la pianificazione non riveste un ruolo di primaria importanza (ossia laddove si ha arbitrarietá di percorrimento di un cammino).
Indicando le derivate successive con velocitá e accelerazioni, una *spline* cubica si puó definire matematicamente come:

$$s(t) = \{p_k(t),\, t \in [t_k, t_{k+1}], k = 0, \ldots, n-1\}$$

$$\text{dove} \quad p_k(t) = a_{k,0} + a_{k,1}(t - t_k) + a_{k,2}(t - t_k)^2 + a_{k,3}(t - t_k)^3 \tag{2.40}$$

In questo modo si hanno 4 coefficienti per ciascuno degli $n-1$ polinomi: in totale $4(n-1)$ parametri da definire. D'altra parte si hanno i seguenti vincoli:

- $2(n-1)$ condizioni di passaggio per i punti, in quanto per ogni polinomio

si deve garantire la continuitá della *spline* nei due punti estremi.

- $n-2$ condizioni di continuitá delle velocitá nei punti intermedi, per assicurare la regolaritá della curva.
- $n-2$ condizioni di continuitá delle accelerazioni nei punti intermedi, per assicure che ogni punto intermedio non sia di flesso.

Si hanno dunque:

$$4(n-1) - 2(n-1) - 2(n-2) = 2$$

gradi di libertá che possono essere utilizzati per imporre opportune condizioni sulla velocitá iniziale e finale. Dal momento che nella trattazione di questa tesi saranno disponibili dagli algoritmi precedentemente descritti sia le posizione che le velocitá nello spazio operativo [4], la trattazione si semplifica, senza dover assegnare le condizioni sulle accelerazioni. Assumendo nulla la velocitá iniziale e finale di ogni giunto del manipolatore, per ciascun polinomio cubico si hanno quattro condizioni al contorno su posizione e velocitá (assegnate come $\{q_1, \ldots, q_n\}$ e $\{v_1, \ldots, v_{n-1}\}$), le quali, partendo dalla (2.40), danno luogo a un sistema di quattro equazioni:

$$\begin{cases} p_k(t_k) = a_{k,0} = q_k \\ \dot{p}_k(t_k) = a_{k,1} = v_k \\ p_k(t_{k+1}) = a_{k,0} + a_{k,1}T_k + a_{k,2}T_k^2 + a_{k,3}T_k^3 = q_{k+1} \\ \dot{p}_k(t_{k+1}) = a_{k,1} + 2a_{k,2}T_k + 3a_{k,3}T_k^2 = v_{k+1} \end{cases} \tag{2.41}$$

essendo $T_k = t_{k+1} - t_k$. Il sistema (2.41) puó essere riscritto usando le prime due equazioni sostituite nelle ultime due, ottenendo due equazioni in due incognite. In forma matriciale si ha:

$$\begin{bmatrix} T_k^2 & T_k^3 \\ 2T_k & 3T_k^2 \end{bmatrix} \begin{bmatrix} a_{k,2} \\ a_{k,3} \end{bmatrix} = \begin{bmatrix} q_{k+1} - q_k - v_k T_k \\ v_{k+1} - v_k \end{bmatrix} \tag{2.42}$$

Le incognite possono essere agevolmente ricavate usando il *metodo di Kramer*. Raggruppando le soluzioni, si ha infine che i coefficienti sono univocamente

[4] generalmente ció non accade, e le velocitá dei punti intermedi non sono note, elevando la complessitá computazionale dell'algoritmo.

determinati come:

$$\begin{cases} a_{k,0} = q_k \\ a_{k,1} = v_k \\ a_{k,2} = \frac{1}{T_k}\left[\frac{3(q_{k+1}-q_k)}{T_k} - 2v_k - v_{k+1}\right] \\ a_{k,3} = \frac{1}{T_k^2}\left[\frac{2(q_k-q_{k+1})}{T_k} + v_k + v_{k+1}\right] \end{cases} \tag{2.43}$$

che forniscono pertanto i quattro parametri necessari per definire per gli $n-1$ polinomi raccordati che formano la *spline* interpolante.
Dunque, se da una parte l'interpolazione punto-punto modella il movimento realizzabile dal manipolatore nell'intervallo di campionamento che passa tra due pose calcolate dall'elettronica di controllo, dall'altra l'interpolazione con *spline* cubiche permette di fornire *a monte* i riferimenti al controllore quando viene indicato un numero ristretto di punti che il manipolatore deve seguire (come si vede in 2.6), riferimenti che poi saranno campionati ad un certo intervallo di tempo e realizzati con movimenti che seguano la prima tecnica descritta. Per questo motivo le traiettorie, spesso complesse ed articolate, descritte in spazio di lavoro da manipolatori assistivi (che usano motori elettrici, a basso consumo di potenza) risultano spesso piuttosto scattose e imprecise [4][5].

Capitolo 3

Il manipolatore MANUS

Il MANUS é un prodotto realizzato e commercializzato dall'azienda olandese *Exact Dynamics*: si tratta di un manipolatore antropomorfo non ridondante a 6 gradi di libertá conferiti da altrettanti giunti rotoidali, con l'aggiunta di un settimo grado rappresentato dal *gripper* in figura 3.1 [14]:

Figura 3.1: L'organo terminale del MANUS.

A questi sette gradi di libertá é possibile aggiungerne un ottavo rappresentato da un supporto mobile in grado di variare l'altezza della base del manipolatore (*lift*). I 6 gradi di libertá consentono al MANUS di poter posizionare con orientazione arbitraria il *gripper* in un punto qualsiasi all'interno del proprio *spazio di lavoro destro*, che é rappresentato da una sfera di raggio di $80cm$ circa.

3.1 Caratteristiche meccaniche

Dal momento che il MANUS é stato concepito per essere collocato su una sedia a rotelle, un'attenzione particolare in fase di progettazione é stata rivolta al problema degli ingombri e dei pesi, in modo da non incidere eccessivamente sulla mobilitá dell'utente. In quest'ottica il manipolatore é dotato di una procedura automatica di chiusura su se stesso quando non é utilizzato, a cui si affianca una procedura che provvede a riportarlo in posizione operativa nel momento del riutilizzo. Per quanto riguarda il peso, grazie ai materiali impiegati, al dimensionamento dei motori elettrici e alle tecniche utilizzate per la trasmissone del moto dagli attuatori ai giunti, si é potuto limitare a $13Kg$ la massa del dispositivo, fissando peró allo stesso tempo a soli $1,5Kg$ il massimo carico utile, sostenibile dal *gripper* (pinza) [4], [11]. Una delle particolaritá costruttive che distingue il MANUS dalla maggior parte degli altri manipolatori antropomorfi é il fatto che tutti gli attuatori sono collocati nella base verticale, che viene fatta ruotare dal primo giunto della catena cinematica. Tale scelta é stata fatta con l'intento di ridurre il peso complessivo della struttura: il posizionamento degli attuatori nella base ed il conseguente alleggerimento degli altri bracci garantisce infatti la possibilitá di azionare il manipolatore con una minor richiesta di potenza. Inoltre tramite l'impiego di motori elettrici piú piccoli e piú leggeri, si consentono consumi energetici inferiori e una maggior durata delle batterie che alimentano il sistema. Un ulteriore vantaggio derivante dalla posizione degli attuatori é la totale assenza di cablaggi all'interno dei bracci, che offre la possibilitá di far compiere ai giunti (ad eccezione del quinto) piú di un giro completo intorno all'asse di rotazione del giunto. La trasmissione del moto dagli attuatori ai 6 giunti ed al gripper avviene tramite cinghie collegate direttamente, o connesse fra loro in modo da poter raggiungere i giunti piú distanti dalla base. In particolare questo discorso vale per il secondo e il terzo giunto, che non sono indipendenti ma connessi in modo tale che, quando al secondo giunto viene imposto uno spostamento angolare $\Delta\theta_2$, il terzo risponde con una rotazione opposta e di pari ampiezza, per mantenere costante la direzione dell'asse del terzo braccio [5]. Tali soluzioni, oltre che garantire una diminuzione dei pesi delle parti mobili del manipolatore, forniscono un primo meccanismo di sicurezza, consentendo di limitare la massima forza di rotazione applicata su ciascun giunto tramite lo slittamento delle cinghie. L'adozione di ginghie comporta peró l'introduzione di giochi e non linearitá nella catena di trasmissione del moto, che, soprattutto per i giunti piú lontani dalla base (giunti 4, 5 e 6), limitano la precisione di controllo. A questo va aggiunto che le informazioni disponibili sui valori degli angoli di rotazione dei giunti, sono quelle fornite dagli *encoder* posti sugli alberi dei motori, che non necessariamente coincidono con i valori reali a causa delle

imprecisioni introdotte proprio dalle cinghie. All'estremitá opposta del manipolatore rispetto alla base, si trova il *gripper*, che consente al manipolatore di afferrare oggetti ed interagire attivamente con l'ambiente circostante. Le due estremitá sono state progettate in modo da allontanarsi ed avvicinarsi rimanendo sempre parallele fra loro, cosí da garantire una presa costante. L'apertura e la chiusura del *gripper* sono realizzate da un attuatore collocato come tutti gli altri nella base del manipolatore, e collegato tramite cinghia, che consente un'apertura massima del *gripper* di 9 cm [5].

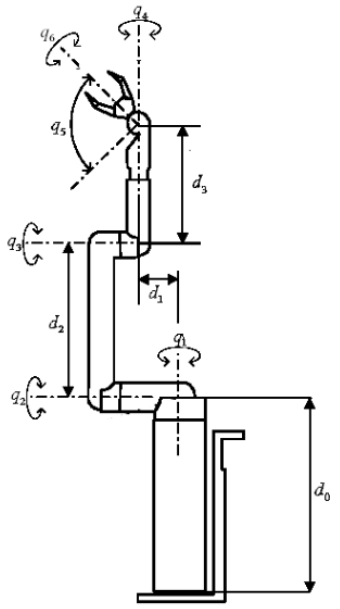

Figura 3.2: Parametri geometrici del MANUS.

3.2 Protocollo di comunicazione

Nel campo della robotica si ha la convergenza delle discipline tipiche dell'ingegneria, ossia: elettronica, meccanica, informatica e automazione. Pertanto, di pari passo con una descrizione della caratteristiche meccaniche, é necessario trattare brevemente anche l'architettura delle componenti hardware e le modalitá di comunicazione tra queste. Riportando principalmente quanto detto in [8], [11], il sistema di controllo del manipolatore (ossia l'unitá che traduce l'nformazione digitale in segnali analogici al manipolatore e viceversa) permette di muovere il manipolatore sia nello spazio dei giunti, impostando cioé singolarmente il valore di ogni giunto, sia nello spazio cartesiano, muovendo cioé il

manipolatore lungo gli assi cartesiani nello spazio.

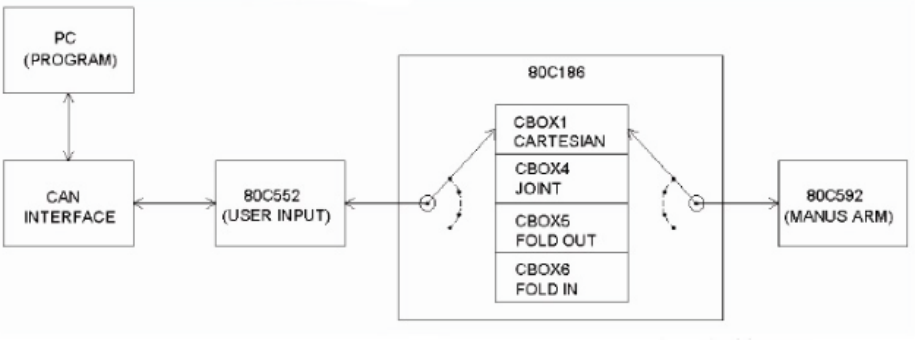

Figura 3.3: Sistema di controllo.

Analizzando in dettaglio il sistema di controllo, mostrato in figura 3.3, si individuano tre microprocessori:

- *Mathematical processor* (INTEL 80C186): si occupa di eseguire le trasformazioni algoritmiche quando il manipolatore opera in *Cartesian Mode*, di verificare le traiettorie per evitare eventuali autocollisioni e, inoltre, di calcolare con un ciclo di $10ms$ (é il *clock* di ciclo-istruzione) le coppie richieste dai motori, con un controllo di tipo PI (proporzionale-integrativo) in base alla retroazione sulle velocitá.

- *User I/O processor* (80C552): gestisce la comunicazione da un utente esterno per quanto riguarda il posizionamento che il MANUS deve assumere. Dunque il *set point* viene comunicato al *Mathematical processor* dall'utente esterno tramite un secondo processore, appunto lo *User I/O processor*.

- *Control I/O processor* (80C592): una volta conclusa l'elaborazione da parte del *Mathematical processor*, i dati di coppia vengono trasmessi ai motori tramite questo terzo microprocessore, il *Control I/O processor* appunto. La comunicazione tra questi due viene realizzata tramite il protocollo CAN-bus (*Controller Area Network*) ed é identificata dall' *Internal* CAN-bus che si distingue dall'*External* CAN-bus. Il primo realizza la comunicazione tra il *control box* (CBOX in figura 3.3) che il *Mathematical processor* mette a disposizione per le diverse modalitá di lavoro opzionabili, e gli attuatori dei giunti, mentre il secondo realizza la comunicazione tra l'utente e il control box selezionato. L'*External* CAN-bus viene utilizzato, infatti, quando si richiede che il manipolatore venga comandato diretta-

mente da un utente, mentre non é possibile, connettere direttamente un elaboratore all'*Internal* CAN-bus.

Per quanto riguarda il *control box*, é possibile selezionare tra i possibili stati di funzionamento del manipolatore:

- *Stati preprogrammati* (CBOX5, CBOX6): permettono di accedere a due procedure preprogrammate dal costruttore che fanno sí che il manipolatore, partendo dalla posizione di riposo, si posizioni in quella operativa e, viceversa, da qualsiasi posizione operativa vada a posizionarsi nella posizione di riposo, in modo totalmente automatico.
- *Stato cartesiano* (CBOX1): il *gripper* del manipolatore MANUS in questa modalitá puó effettuare una traslazione singolarmente lungo i tre assi cartesiani x, y, z (*Cartesian Mode*).
- *Stato dei giunti* (CBOX4): ogni singolo asse di rotazione relativo ad ogni giunto rotoidale del manipolatore puó essere orientato liberamente; questo stato di funzionamento risulta essere quello maggiormente libero da vincoli.

Nello stato precedente infatti il sistema di controllo si occupa di effettuare, oltre alla *cinematica inversa*, anche alcuni controlli di sicurezza per evitare eventuali autocollisioni. Ció non avviene in quest'ultimo stato di funzionamento, poiché si agisce direttamente sul valore angolare della variabile di giunto.
Come ultima osservazione, alla luce di come é stato sviluppato il manipolatore, questo é stato realizzato per ridurre il consumo di potenza, ottimizzando la tensione di lavoro e la corrente assorbita [11]. Infatti l'alimentazione necessita di una tensione di $24V$ e di una corrente continua di $2A$, per un consumo che non supera dunque i $50W$. Infine, come accennato nel paragrafo precedente, per poter comunicare con il sistema di controllo del manipolatore, risulta necessario inviare messaggi su di una rete CAN per la quale é necessario utilizzare un altro *controller*, implementato nell'*hardware* da una scheda ISA (*Instruction Set Architecture*) a 16-bit. Il problema essenziale di questa scheda peró é che essa é poco adattabile ai sistemi odierni, dato che il bus ISA risulta ormai sorpassato e sostituito dal piú moderno bus PCI. Questa considerazione ha spinto dunque a ricercare una scheda PCI che fornisca un'interfaccia CAN-bus, con la caratteristica di potersi interfacciare con sistemi operativi *Linux* mediante i *driver* forniti dal produttore.

3.2.1 Comunicazione CAN-bus

Come giá detto, il manipolatore comunica con l'utente esterno attraverso un *controller* CAN-bus, che nello specifico é un bus di tipo *multi-master* nel quale

i *messaggi*, ossia gli *oggetti* della comunicazione, non sono diretti a destinazioni specifiche, ma istante per istante ogni dispositivo (o *endpoint*) puó assumere il controllo del bus e spedire un messaggio. Ogni messaggio ha un ID che lo identifica e ogni *endpoint* decide autonomamente se il messaggio é di sua competenza o meno. Questo implica che si possono spedire messaggi, con differenti *prioritá*, anche a piú *endpoint* contemporaneamente. L'informazione sul CAN-bus viene inviata attraverso *pacchetti*, in cui ogni messaggio ha un contenuto informativo diviso nei seguenti campi (vd. tabella esemplificativa 3.1):

- ID: l'identificativo del messaggio, codificato con valore esadecimale.
- EXT: con valore uguale ad 1 indica che il messaggio é un esteso, con valore 0 se si tratta di un messaggio standard.
- RTR: indica al sistema in ascolto se il messaggio necessita di una risposta: pari a 1 se il *flag* RTR é stato settato e il sistema mittente richiede un messaggio di risposta, 0 viceversa.
- DLC: lunghezza del campo dati del codice, dato dal numero di byte di dati in questo messaggio, con un massimo di 8 byte disponibili.
- DATA: contenuti in un campo di 8 byte, ognuno dei quali contiene la codifica di un'informazione precisa. In particolare queste stringhe sono indicate con A0, ..., A7 e contengono gli ingressi poi processati dal *Mathematical processor*.

Campo	**Tipo di dato**	**Esempi**
ID	numero esadecimale	0x350, 0x37F
RTR	variabile booleana	0, 1
DCL	numero decimale	0, ..., 8
DATA	stringa a 8-bit	11011100

Tabella 3.1: Campi di un messaggio nella comunicazione CAN-bus

Il contenuto di ogni messaggio dipende dal *control box* che si vuole selezionare per la modalitá di lavoro del manipolatore. Infatti quando si controlla il braccio utilizzando la *trasparent mode* che permette di controllare i movimenti del MANUS direttamente da PC, bisogna selezionare una delle modalitá di controllo (figura 3.3), in cui gli ingressi forniti dagli utenti vengono trattati in maniera diversa: il *control box* che viene selezionato determina il modo in cui gli I/O vengono interpretati dal processore matematico e inoltre determina il significato delle informazioni in *feedback* di posizione. In particolare bisogna definire

la struttura della rete, il formato dei messaggi, il significato da attribuire agli identificatori e ai dati. Come giá detto, il manipolatore MANUS puó essere controllato inviando i comandi dal PC allo *User I/O processor* attraverso l'*External* CAN-bus: infatti la comunicazione tra *I/O control processor*, i *drivers* dei motori e gli *encoders* avviene sull'*Internal* CAN-bus che non é accessibile all'utente. Dunque tale unitá é l'unica che permette di trasferire direttamente i comandi al processore matematico. La comunicazione tramite *External* CAN-bus tra il PC e *User I/O processor* avviene attraverso un protocollo del tipo *domanda-risposta*, caratterizzato da una tempistica ben definita. Si considerino il seguente schema e la seguente tabella:

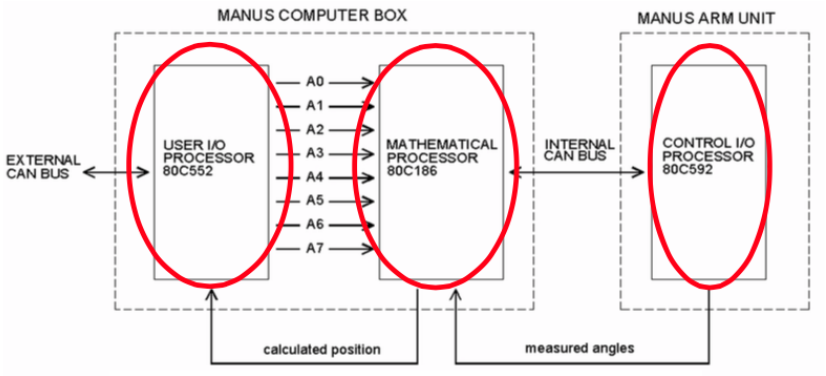

Figura 3.4: Livelli nella comunicazione CAN-bus.

ID	DLC	interpretazione messaggio
computer box $\rightarrow$ *PC*		
0x350	8	Stato del sistema e valore delle prime tre variabili di giunto
0x360	8	Ultime tre variabili di giunto + grado di libertá del *lift*
0x37F	0	Interrogazione per comando da PC
computer box $\leftarrow$ *PC*		
0x371	8	Selezione del CBOX1 e assegnazione *posizione* desiderata
0x374	8	Selezione del CBOX4 e assegnazione *configurazione* desiderata
0x375	0	Selezione del CBOX5 (Fold out)
0x376	0	Selezione del CBOX6 (Fold in)

Tabella 3.2: Campi di un messaggio nella comunicazione CAN-bus

In riferimento a questi: quando l'unitá *computer box* del MANUS (composta da *User I/O processor* e *mathematical processor* in figura 3.4) inviano due *data frame* (ID 0x350, 0x360) e un *remote frame* (ID 0x37F) spaziati tra loro

ognuno di $20ms$, al termine di questo intervallo di tempo il PC ha un tempo di $20ms$ per inviare a sua volta un *data frame* a sua disposizione; se, trascorso questo tempo, L'*External* CAN non ha ancora risposto con alcun comando, il *computer-box* continua ad eseguire i movimenti secondo i comandi precedenti. Pertanto lo *User I/O processor* invia un pacchetto al PC ogni $20ms$: per primi vengono inviati due pacchetti di informazione e il terzo pacchetto inviato serve a richiedere una risposta al PC. Pertanto il PC puó inviare i comandi al MANUS ogni $60ms$, che diventa dunque l'intervallo minimo in lettura da parte del sistema di controllo di un informazione inviata da utente [8], [11]. Un esempio di comunicazione secondo questo protocollo é identificata nella seguente tabella, dove l'abbreviazione "CB" sta per *computer box*:

Tempo	**TX/RX**	**Descrizione messaggio**
$20ms$	CB/PC	ID0x350: stato del sistema e valore attuale delle variabili
$40ms$	CB/PC	ID0x360: assegnazione delle variabili
$60ms$	CB/PC	ID0x37F: *remote frame* di richiesta comando
$< 80ms$	PC/CB	ID0x371 e dati nulli: selezione del CBOX0
$80ms$	CB/PC	ID0x350: stato del sistema e valore attuale delle variabili
$100ms$	CB/PC	ID0x360: assegnazione delle variabili
$120ms$	CB/PC	ID0x37F: *remote frame* di richiesta comando
$< 120ms$	PC/CB	ID0x371 e byte2=1: movimento in direzione dell'asse x

Tabella 3.3: Esempio di comunicazione CAN-bus

3.3 Analisi cinematica del Manus

Dal punto di vista cinematico il MANUS presenta una configurazione *non ridondante* a sei gradi di libertá: ció significa che per il raggiungimento di una generica posa nello spazio, concorrono tutti e sei i gradi di libertá del manipolatore, perdendo la mobilitá di uno o piú giunti solo in corrispondenza delle configurazioni singolari. Gli ultimi tre giunti sono disposti in modo da formare un *polso sferico* a rappresentazione minima, che consente la risoluzione del problema cinematico in forma chiusa tramite la proprietá detta ***disaccoppiamento cinematico***. Questa permette di trattare il problema cinematico inverso scomponendolo in due fasi: nella prima si calcola la configurazione relativa alla posizione del centro del polso, determinando la configurazione di *struttura portante*, mentre nella seconda ci si occupa di calcolare l'orientamento dell'organo terminale, determinando la configurazione di *polso sferico*.

3.3.1 Cinematica diretta

Il calcolo della cinematica diretta si riduce ad una semplice procedura algoritmica una volta noti i parametri di Denavit-Hartenberg, che traducono in una semplice tabella tutte le informazioni riguardanti la natura dei giunti e degli assi di rotazione braccio per braccio. Per il manipolatore MANUS, sono riassunti nella tabella 3.4:

i	a_i	α_i	d_i	θ_i
1	0	$-\frac{\pi}{2}$	0	q_1
2	L_2	0	L_1	q_2
3	0	$-\frac{\pi}{2}$	0	q_3
4	0	$\frac{\pi}{2}$	L_3	q_4
5	0	$-\frac{\pi}{2}$	0	q_5
6	0	0	L_4	q_6

Tabella 3.4: I parametri di Denavit-Hartenber del MANUS.

Questi parametri rendono conto della geometria delle connessioni dei bracci, la dimensione geometrica prevalente di ognuno di questi (lunghezza) e la natura dei giunti. Nella figura 3.5 é mostrato un modello del manipolatore MANUS quando é a riposo (nessun giunto viene attuato e assenza di forze esterne escluso il campo gravitazionale):

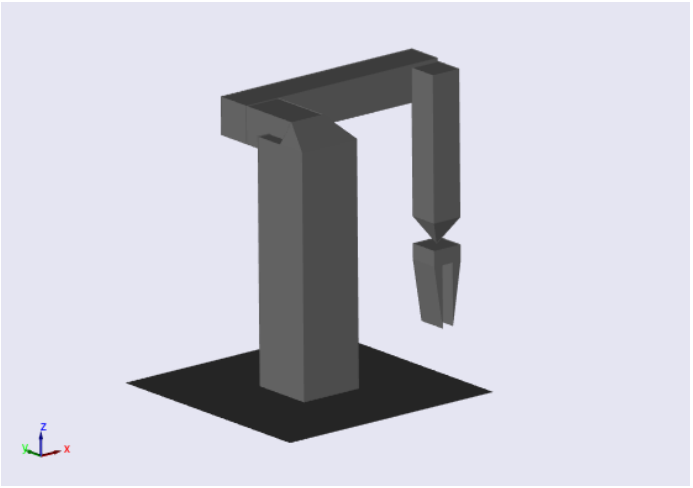

Figura 3.5: Il manipolatore MANUS in posa iniziale.

Le lunghezze di ogni braccio sono fornite in letteratura, e corrispondono a:

L_1	105 mm
L_2	400 mm
L_3	320 mm
L_4	160 mm

Tabella 3.5: Valori dei parametri geometrici.

Con questi parametri, applicando la procedura per il computo della matrice di trasformazione che esprime l'orientamento del sistema solidale con l'organo terminale si ottengono le sequenti matrici omogenee per ognuna delle 6 variabili di giunto:

$$\mathbf{A}_1^0 = \begin{bmatrix} \cos q_1 & 0 & -\sin q_1 & 0 \\ \sin q_1 & 0 & \cos q_1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{A}_2^1 = \begin{bmatrix} \cos q_2 & -\sin q_2 & 0 & L_2 \cos q_2 \\ \sin q_2 & \cos q_2 & 0 & L_2 \sin q_2 \\ 0 & 0 & 1 & L_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{A}_3^2 = \begin{bmatrix} \cos q_3 & 0 & -\sin q_3 & 0 \\ \sin q_3 & 0 & \cos q_3 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{A}_4^3 = \begin{bmatrix} \cos q_4 & 0 & \sin q_4 & 0 \\ \sin q_4 & 0 & -\cos q_4 & 0 \\ 0 & 1 & 0 & L_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{A}_5^4 = \begin{bmatrix} \cos q_5 & 0 & -\sin q_5 & 0 \\ \sin q_5 & 0 & \cos q_5 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{A}_6^5 = \begin{bmatrix} \cos q_6 & -\sin q_6 & 0 & 0 \\ \sin q_6 & \cos q_6 & 0 & 0 \\ 0 & 0 & 1 & L_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Considerando che il riferimento *base* coincide con la terna indicata con 1 e il riferimento dell'organo terminale con la terna n, usando la (2.17) si puó scrivere la matrice di trasformazione finale come:

$$\mathbf{T}(\mathbf{q}) = \mathbf{A}_0^1(q_1)\mathbf{A}_1^2(q_2)\mathbf{A}_2^3(q_2)\mathbf{A}_3^4(q_2)\mathbf{A}_4^5(q_2)\mathbf{A}_5^6(q_n)$$

$$= \begin{bmatrix} T_{11} & T_{12} & T_{13} & T_{14} \\ T_{21} & T_{22} & T_{23} & T_{24} \\ T_{31} & T_{32} & T_{33} & T_{34} \\ T_{41} & T_{42} & T_{43} & T_{44} \end{bmatrix} \tag{3.1}$$

Esaminando i blocchi di quest'ultima (ponendo $c_i = \cos q_i$ e $s_i = \sin q_i$), ricordando che ad ogni indice in piú corrisponde una somma degli argomenti

delle funzioni a cui si riferiscono gli indici, si ottengono le seguenti equazioni:

$$p_x = T_{14} = -L_4(s_5(s_1s_4 + c_4c_1c_{23}) - c_5c_1s_{23}) - L_3c_1s_{23} + L_2c_1c_2 - L_1s_1$$
$$p_y = T_{24} = L_4(s_5(c_1s_4 - c_4s_1c_{23}) - c_5s_1s_{23}) - L_3s_1s_{23} + L_2c_2s_1 + L_1c_1$$
$$p_z = T_{34} = -L_4(c_5c_{23} + c_4s_5s_{23}) - L_3c_{23} - L_2s_2$$

$$r_{11} = T_{11} = s_6(c_4s_1 - s_4c_1c_{23}) + c_6(c_5(s_1s_4 + c_4c_1c_{23}) - s_5c_1s_{23})$$
$$r_{12} = T_{12} = c_6(c_4s_1 - s_4c_1c_{23}) - s_6(c_5(s_1s_4 + c_4c_1c_{23}) - s_5c_1s_{23})$$
$$r_{13} = T_{13} = -s_5(s_1s_4 + c_4(c_1c_2c_3 - c_1s_2s_3)) - c_5(c_1c_2s_3 + c_1c_3s_2)$$
$$r_{21} = T_{21} = -s_6(c_1c_4 + s_4s_1c_{23}) - c_6(c_5(c_1s_4 - c_4s_1c_{23}) + s_5s_1s_{23})$$
$$r_{22} = T_{22} = s_6(c_5(c_1s_4 + c_4s_1c_{23}) + s_5s_1s_{23}) - c_6(c_1c_4 + s_4s_1c_{23})$$
$$r_{23} = T_{23} = s_5(c_1s_4 - c_4s_1c_{23}) - c_5s_1s_{23}$$
$$r_{31} = T_{31} = s_4s_6s_{23} - c_6(s_5c_{23} + c_4c_5s_{23})$$
$$r_{32} = T_{32} = s_6(s_5c_{23} + c_4c_5s_{23}) + c_6s_4s_{23}$$
$$r_{33} = T_{33} = c_4s_5s_{23} - c_5c_{23}$$

i termini della quarta riga sono tutti nulli, tranne $T_{44} = 1$. I blocchi fondamentali sono il blocco (3×3) della matrice di rotazione e quello del vettore di posizione del centro del riferimento solidale con l'organo terminale. Dunque si ottengono 12 equazioni in 6 incognite per risolvere il problema cinematico. Sará molto utile sfruttare la proprietá di disaccoppiamento cinematico per risolvere solo un sottoinsieme tra queste in funzione di un numero minore di incognite. Infatti per la struttura portante basterá considerare i termini T_{41}, T_{42}, T_{43} azzerando il paramtro L_4 (per indicare che la catena ha fine in corrispondenza del centro del polso). Analogamente per il polso sferico sará sufficiente calcolare la matrice di rotazione della convenzione di Eulero degli ultimi tre giunti (q_1, q_2, q_3 considerati nulli) ed eguagliarla a quella che esprime l'orientamento finale.

3.3.2 Cinematica inversa

Una volta nota la funzione di cinematica diretta k, descritta nella (3.1), il problema della cinematica inversa é quello di ricavare in *forma chiusa* la configurazione ai giunti per cui un certo manipolatore ha una determinata posizione e orientamento dell'organo terminale nello spazio. In sostanza, dati in ingresso un vettore che indichi *la posizione desiderata* $\mathbf{p}$ e una *terna di Eulero* $\mathbf{\Phi}$ che indichi l'orientamento del riferimento solidale all'organo terminale rispetto la base, si desidera trovare l'equivalente configurazione ai giunti.

Il problema viene impostato matematicamente nel seguente modo:

$$\begin{bmatrix} \tilde{\boldsymbol{\theta}} \\ - \\ \hat{\boldsymbol{\theta}} \end{bmatrix} = k^{-1}(\mathbf{p}, \boldsymbol{\Phi})$$

dove in genere con $\tilde{\boldsymbol{\theta}}$ si indica il vettore dei giunti riferiti alla *struttura portante*, mentre con $\hat{\boldsymbol{\theta}}$ quelli del *polso sferico*. Come giá detto in precedenza, il MANUS gode della proprietá di disaccoppiamento cinematico, pertanto in virtú di questa é possibile studiare la posizione del *centro del polso* rispetto ai primi tre giunti, ed in seguito l'orientamento del polso sferico, il cui riferimento non é solidale con il centro del polso ma con l'apice della pinza dell'organo terminale, che ha una propria lunghezza L_4. Da ció si intuisce che la posizione desiderata $\mathbf{p}$ costituisce "l'ingresso" per risolvere la prima parte del problema, mentre la terna di Eulero $\boldsymbol{\Phi}$ per la seconda. Diversamente dal precedente paragrafo, nella risoluzione del problema inverso si incontrano maggiori difficoltá nell'esprimere la soluzione in forma chiusa, soluzione peraltro non univoca: infatti si possono ricavare piú pose per cui si abbia la stessa posizione o lo stesso orientamento.

3.3.3 Centro di polso

Dal momento che, come si é giá detto, il centro del polso non coincide con il centro del riferimento solidale con l'organo terminale (fatto coincidere con la punta della pinza quando é chiusa), la struttura portante dovrá essere studiata a partire dalla posizione del centro del polso $\tilde{\mathbf{p}}$, diversa dalla posizione desiderata, e ricavabile da $\mathbf{p}$. Per riscalare tali coordinate occorre introdurre la specifica di orientamento della terna solidale all'organo terminale espressa dalla terna di Eulero $\boldsymbol{\Phi}$: questa é una delle convenzioni di Eulero, e prende il nome di *angoli aeronautici* o notazione RPY (dall'inglese *Roll*, rollio, *Pitch*, beccheggio, *Yaw*, imbardata). Questa convenzione, che é diversa da quella adottata nel polso sferico (convenzione ZYZ, ad assi correnti), descrive l'orientamento del sistema di riferimento finale rispetto a quello di base con la seguente composizione di rotazioni, ad *assi fissi*:

- si ruota la terna origine di un angolo α lungo l'asse x, detto *angolo di imbardata*.
- si ruota, sempre rispetto la terna origine, di un angolo β lungo l'asse y, detto *angolo di beccheggio*.
- si ruota infine rispetto alla terna origine, di un angolo γ lungo l'asse z, detto *angolo di rollio*.

Da questa successione si puó ottenere una matrice di rotazione *premoltiplicando*[1] le matrici di rotazione, cioé:

$$\mathbf{R}_{RPY} = \mathbf{R}_z(\gamma)\mathbf{R}_y(\beta)\mathbf{R}_x(\alpha) =$$

$$\begin{bmatrix} \cos\gamma\cos\beta & \cos\gamma\sin\beta\sin\alpha - \sin\gamma\cos\alpha & \cos\gamma\sin\beta\cos\alpha - \sin\gamma\sin\alpha \\ \sin\gamma\cos\beta & \sin\gamma\sin\beta\sin\alpha - \cos\gamma\cos\alpha & \sin\gamma\sin\beta\cos\alpha - \cos\gamma\sin\alpha \\ -\sin\beta & \cos\beta\sin\alpha & \cos\beta\cos\alpha \end{bmatrix}$$

Osservando la figura 3.6, si consideri la seguente relazione vettoriale:

$$(P - O) = (P - \tilde{P}) + (\tilde{P} - O) \tag{3.2}$$

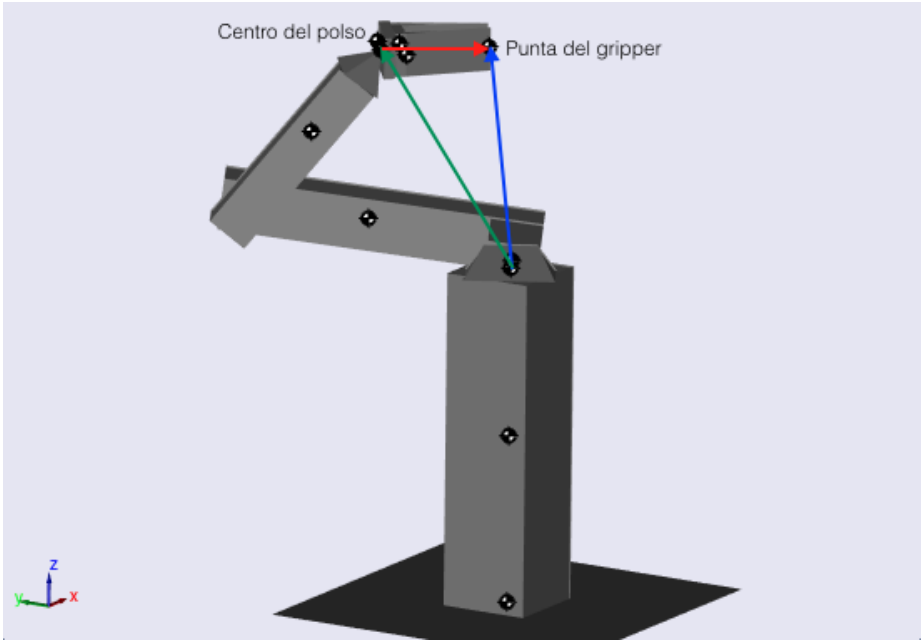

Figura 3.6: interpretazione grafica della (3.2)

dove con $\tilde{P}$ si indica il punto relativo al centro del polso, con O il centro della terna di base e con P il punto che identifica la punta della pinza del manipolatore. Immaginando che in $\tilde{\mathbf{p}}$ siano concentrati tutti e tre i giunti del polso sferico, una volta indicata l'orientazione RPY, risolvendo il problema inverso, il

[1] Si puó dimostrare che la formula per le rotazioni fisse ha la stessa struttura di (2.8), ma con ordine di moltiplicazione delle matrici invertito.

polso assumerá la terna ai giunti, con terna di riferimento orientata rispetto alla base tramite la matrice $\mathbf{R}$. Dato che la pinza dell'organo terminale puó essere stilizzata come un segmento di lunghezza L_4, una volta orientato il polso, essa giace lungo la direzione dell'asse z del riferimento solidale con l'organo terminale: questa direzione é individuata dal versore $\mathbf{z}'$ dell'asse z della terna, espresso rispetto alla terna base dalla terza colonna della matrice $\mathbf{R}_{RPY}$. Riprendendo (3.2), si riconoscono $\mathbf{p}_{des} = (P - O)$, e $\tilde{\mathbf{p}} = (\tilde{P} - O)$, ed esprimendo il vettore $(P - \tilde{P}) = L_4\mathbf{z}'$ si ha:

$$\tilde{\mathbf{p}} = \mathbf{p}_{des} - L_4\mathbf{z}' \tag{3.3}$$

Quest'ultima é molto importante perché permette di riscalare le coordinate di una determinata posizione desiderata a quelle di centro del polso: rispetto a queste é possibile risolvere la prima parte del problema cinematico usando la fondamentale proprietá di disaccoppiamento cinematico, che semplifica enormemente la trattazione del problema inverso. Si osservi che il riscalamento delle coordinate si rende necessario proprio perché, ai fini della manipolabilitá, occorre connettere ai tre giunti di polso ancora un "braccio" che modelli la mano (si tratta pur sempre di modellare un manipolatore *antropomorfo*, dunque ispirato alla mano umana). Questa prima parte, come si puó vedere dalla (3.2), non chiama in gioco le variabili di giunto, ma solo i parametri di orientazione RPY, definiti come ingresso del problema inverso.

3.3.4 Struttura portante

Una volta note le coordinate riscalate del centro del polso, si puó procedere con la risoluzione della configurazione della *struttura portante*. Il problema si risolve considerando le equazioni della cinematica diretta considerate come se la struttura si concludesse con il riferimento solidale al centro del polso (questo deriva dalla proprietá di disaccoppiamento). In questo modo si ottiene il seguente sistema *non lineare*:

$$\begin{cases} \tilde{p}_x = -L_3 \cos q_1 \sin(q_2 + q_3) + L_2 \cos q_1 \cos q_2 - L_1 \sin q_1 \\ \tilde{p}_y = -L_3 \sin q_1 \sin(q_2 + q_3) + L_2 \sin q_1 \cos q_2 - L_1 \cos q_1 \\ \tilde{p}_z = -L_3 \cos(q_2 + q_3) + L_2 \sin q_2 \end{cases} \tag{3.4}$$

La soluzione di questo sistema, avvalendosi delle tecniche descritte dalle fonti [8], [10], [13], [15], sono:

$$q_{3,1} = -\arcsin\left(\frac{\tilde{p}_x^2 + \tilde{p}_y^2 + \tilde{p}_z^2 - L_1^2 - L_2^2 - L_3^2}{2L_2L_3}\right) \tag{3.5}$$

$$q_{3,2} = \pi - q_{3,1}$$

$$q_{2,j} = -2\arctan\left(\frac{-(L_3\sin q_{3,i} + L_2) \pm \sqrt{\tilde{p}_x^2 + \tilde{p}_y^2 - L_1^2}}{L_3\cos q_{3,i} - \tilde{p}_z}\right) \tag{3.6}$$

$$con \quad i = 1, 2 \quad e \quad j = 1, \ldots, 4$$

$$\begin{aligned} q_{1,1} &= \arctan\left(\frac{\tilde{p}_y(L_3\sin(q_{2,1} + q_{3,1}) + L_2\cos q_{2,1} - \tilde{p}_x L_1}{\tilde{p}_x(L_3\sin(q_{2,1} + q_{3,1}) + L_2\cos q_{2,1} + \tilde{p}_y L_1}\right) \\ q_{1,2} &= \arctan\left(\frac{\tilde{p}_y(L_3\sin(q_{2,2} + q_{3,1}) + L_2\cos q_{2,2} - \tilde{p}_x L_1}{\tilde{p}_x(L_3\sin(q_{2,2} + q_{3,1}) + L_2\cos q_{2,2} + \tilde{p}_y L_1}\right) \end{aligned} \tag{3.7}$$

soluzioni valide sotto le seguente condizioni:

$$\begin{cases} \tilde{p}_x^2 + \tilde{p}_y^2 \geq L_1^2 \\ \tilde{p}_x^2 + \tilde{p}_y^2 + \tilde{p}_z^2 \geq L_1^2 + (L_2 - L_3)^2 \\ \tilde{p}_x^2 + \tilde{p}_y^2 + \tilde{p}_z^2 \leq L_1^2 + (L_2 + L_3)^2 \end{cases} \tag{3.8}$$

Tutte le soluzioni sono considerate nell'intervallo $(-\pi, \pi]$, espresse in radianti. Si osservi che l'unica soluzione indipendente da una variabile di giunto é la soluzione del terzo giunto q_3, che ammette due soluzioni per via della non iniettivitá della funzione seno nell'intervallo considerato. Da questa discende q_2, che ammette quattro soluzioni, dipendenti dalla precedente e dalla scelta del segno (piú o meno). Infine la soluzione del primo giunto q_1 dipende da entrambe, e si distinguono due soluzioni, una dipendente da $q_{2,1}$, l'altra da $q_{2,2}$. Dunque si possono indicare le 4 possibili terne che risolvono il problema inverso come:

$$\tilde{\boldsymbol{\theta}}_1 = \begin{bmatrix} q_{1,1} \\ q_{2,1} \\ q_{3,1} \end{bmatrix} \qquad \tilde{\boldsymbol{\theta}}_2 = \begin{bmatrix} q_{1,2} \\ q_{2,2} \\ q_{3,1} \end{bmatrix} \qquad \tilde{\boldsymbol{\theta}}_3 = \begin{bmatrix} q_{1,1} \\ q_{2,3} \\ q_{3,2} \end{bmatrix} \qquad \tilde{\boldsymbol{\theta}}_4 = \begin{bmatrix} q_{1,2} \\ q_{2,4} \\ q_{3,2} \end{bmatrix}$$

Pertanto bisogna determinare un *criterio di scelta* della soluzione, basato su condizioni di minimo consumo energetico, in genere ottenibili attuando preferibilmente giunti piú piccoli a favore di quelli piú grandi. A questo punto é giá possibile dare un'idea dello spazio di lavoro raggiungibile dalla struttura portante del MANUS (il grafico 3.7 é stato ottenuto dal codice A.2 in appendice):

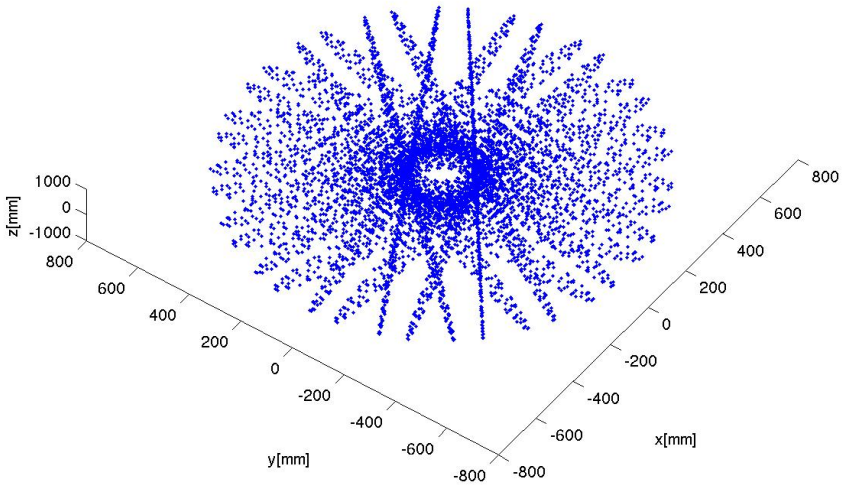

Figura 3.7: Spazio di lavoro raggiungibile

Come si puó vedere, lo spazio di lavoro é una sfera forata, di raggio 80 cm circa: in particolare questo fatto é riscontrabile in figura 3.7, dove si vede chiaramente (e come d'altronde ci si poteva aspettare dalla condizione espressa nel sistema di disuguaglianze (3.8)) che la porzione di spazio nell'intorno dell'asse centrale non é raggiungibile, con un raggio pari esattamente a L_1.

3.3.5 Polso sferico

L'ultima parte alla risoluzione del problema inverso riguarda (una volta posizionato il centro del polso e trovato il vettore $\tilde{\boldsymbol{\theta}}$) l'orientamento della pinza e finalmente il raggiungimento della posizione desiderata. Ció implica che il blocco (3×3) nella matrice omogenea data dalla (3.1) deve essere uguale a quella che si ottiene dalla convenzione RPY. Cioé si deve imporre che:

$$\mathbf{R}_{RPY} = \mathbf{R}_6^0 = \mathbf{R}_3^0(\tilde{\boldsymbol{\theta}})\mathbf{R}_6^4(\hat{\boldsymbol{\theta}}) \tag{3.9}$$

Da cui si ottiene la relazione che lega $\hat{\boldsymbol{\theta}}$, ancora incognita, alla matrice di rotazione riferita alla convenzione RPY e alla matrice di rotazione riferita a $\tilde{\boldsymbol{\theta}}$, calcolata nel paragrafo precedente. Ricordando le proprietá di ortogonalitá della matrice di rotazione, si ha:

$$\mathbf{R}_6^4(\hat{\boldsymbol{\theta}}) = (\mathbf{R}_3^0)^T \mathbf{R}_{RPY} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{3.10}$$

I nove paramtri r_{ij} derivano dal calcolo del prodotto della matrice di rotazione trasposta e riferita alla struttura portante con la matrice di rotazione riferita dalla convenzione RPY. Per esprimere i valori degli angoli degli ultimi tre giunti bisogna considerare la matrice $\mathbf{R}_6^4$: considerando che essa descrive l'orientamento di un polso sferico con convenzione di Eulero ad assi correnti ZYZ, si possono eguagliare i nove parametri r_{ij} con quelli della matrice di rotazione della convenzione ZYZ (nota, §2.1) del polso sferico:

$$\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} =$$

$$\begin{bmatrix} \cos q_4 \cos q_5 \cos q_6 - \sin q_4 \sin q_6 & -\cos q_4 \cos q_5 \sin q_6 - \sin q_4 \cos q_6 & \cos q_4 \sin q_5 \\ \sin q_4 \cos q_5 \cos q_6 + \cos q_4 \sin q_6 & -\sin q_4 \cos q_5 \sin q_6 + \cos q_4 \cos q_6 & \sin q_4 \sin q_5 \\ -\sin q_5 \cos q_6 & \sin q_5 \sin q_6 & \cos q_5 \end{bmatrix}$$

La precedente definisce un insieme di 9 equazione di cui tre sono sufficienti alla determinazione degli ultimi tre giunti. Calcolati i nove parametri r_{ij}, si possono determinare nel modo seguente, ricordando che la funzione $atan2$ fornisce una soluzione univoca a partire dagli argomenti:

$$\begin{gathered} \frac{-r_{23}}{-r_{13}} = \frac{r_{23}}{r_{13}} = \frac{\sin q_4 \sin q_5}{\cos q_4 \sin q_5} = \tan q_4 \Rightarrow \\ q_{4,1} = atan2(-r_{23}, -r_{13}) \quad \text{oppure} \quad q_{4,2} = atan2(r_{23}, r_{13}) \\ \text{(2 soluzioni)} \end{gathered} \tag{3.11}$$

$$\begin{gathered} \pm \frac{\sqrt{r_{31}^2 + r_{32}^2}}{r_{33}} = \frac{\pm \sin q_5}{\cos q_5} = \pm \tan q_5 \Rightarrow \\ q_{5,1} = atan2(\sqrt{r_{31}^2 + r_{32}^2}, r_{33}) \quad \text{oppure} \quad q_{5,2} = atan2(-\sqrt{r_{31}^2 + r_{32}^2}, r_{33}) \\ \text{(2 soluzioni)} \end{gathered} \tag{3.12}$$

$$\begin{gathered}\frac{-r_{32}}{r_{31}} = \frac{r_{32}}{-r_{31}} = \frac{\sin q_5 \sin q_6}{\sin q_5 \cos q_6} = \tan q_6 \Rightarrow \\ q_{6,1} = atan2(-r_{32}, r_{31}) \quad \text{oppure} \quad q_{6,2} = atan2(r_{32}, -r_{31}) \\ \text{(2 soluzioni)}\end{gathered} \tag{3.13}$$

Pertanto le due soluzioni si possono raggruppare cosí:

$$\hat{\boldsymbol{\theta}}_1 = \begin{bmatrix} q_{4,1} \\ q_{5,1} \\ q_{6,1} \end{bmatrix} \qquad \hat{\boldsymbol{\theta}}_2 = \begin{bmatrix} q_{4,2} \\ q_{5,2} \\ q_{6,2} \end{bmatrix}$$

La soluzione anche in questo caso non é univoca e sará necessario considerare un criterio di scelta. Ogni valore é compreso nell'intervallo $(-\pi, \pi]$, eccetto per il quinto giunto (che ha uno spostamento assoluto massimo di 126 gradi [15]), e gli angoli sono espressi in radianti.

Capitolo 4

Implementazione dei modelli

L'obiettivo della tesi é lo sviluppo di modello cinematici e la loro successiva implementazione in ambiente di simulazione: ció permette di visualizzare il risultato della pianificazione del moto nello spazio di lavoro di un manipolatore antropomorfo in realtá virtuale. Tuttavia occorre sottolineare che la fedeltá della simulazione é intrinsecamente connessa alla conoscenza del sistema: piú essa é dettagliata e piú la simulazione riesce a predirne esattamente il comportamento, mentre se si ha una conoscenza solamente "esterna" del sistema, senza le dovute specifiche di come é stato realizzato, allora é probabile che la simulazione trascuri molti aspetti effettivamente riscontrabili nella realtá. Nel nostro caso ci troviamo nella seconda ipotesi, e pertanto si é dovuto abbassare il livello di analisi del modello, da dinamico a cinematico. Le principali fonti consultate per la modellazione in *Simulink* sono state principalmente [1] e [12], oltre ai *datasheet* forniti da MathWorks [17].
L'ambiente software piú adatto allo sviluppo é l'ambiente di calcolo numerico MATLAB®, usato per la stesura degli algoritmi, trasportabili in *Simulink*, che é un ambiente appunto integrato in MATLAB, rivolto in particolare alla simulazione di modelli cinematici/dinamici e di sistemi *embedded*. Riguardo a questo, la sua facilitá di utilizzo sta nell'*editor* grafico, che permette di trascinare dalle varie librerie i *blocchi* necessari, collegati tra loro in uno schema in cui si definiscono ingressi, uscite, blocchi che implementano funzioni MATLAB, sottosistemi, grafici etc. Una volta definito il modello, il codice viene generato automaticamente ed é possibile avviare la simulazione. In questo capitolo si discuteranno tre modelli implementati in questo ambiente: il modello *punto-punto* [13], che descrive appunto il moto punto-punto descritto nel paragrafo 2.3.1, il modello *a inseguimento di traiettoria* per la simulazione di alcuni particolari cammini nello spazio di lavoro, e infine il modello *Master-Slave* che, ereditando gran parte dai primi due, simula in realtá virtuale il movimento coordinato di

due manipolatori nel tracciare due archi di circonferenza partendo da posizioni diametralmente opposte. Per quanto riguarda gli algoritmi (i codici sono riportati in appendice), sono state implementate la cinematica inversa (seguendo i passi del paragrafo 3.3), le tecniche di interpolazione (§§2.3.1,2.3.2), e gli algoritmi di inversione cinematica (§2.2). In particolare per l'implementazione del problema cinematico inverso, gli algoritmi necessari sono parte integrante del modello *Simulink*, mentre per gli altri due si é preferito effettuare a parte tutta la parte riguardante la pianificazione mandando direttamente in ingresso i risultati al simulatore dei modelli (riducendo la complessitá).

4.1 Modello punto-punto

Il modello punto-punto è il primo modello in simulazione sviluppato per questa tesi, ed è quello da cui vengono ereditate le principali caratteristiche nei successivi: ció che si propone di simulare è il moto punto-punto. É possibile lavorare in due *modalitá*: nel caso in cui siano forniti gli ingressi direttamente come *posa finale* definendo $\tilde{\boldsymbol{\theta}}$ e $\hat{\boldsymbol{\theta}}$ siamo in modalitá cinematica diretta; nel caso in cui si forniscano *posizione desiderata* e *orientamento RPY* invece ci troviamo in modalitá cinematica inversa. Nella figura B.1 riportata in appendice, si puó vedere la struttura complessiva del modello: sono definiti 6 ingressi, tra cui se ne scelgono sempre 4 (2 ingressi piú 2 condizioni iniziali per struttura portante e per polso sferico) a seconda della modalitá in cui si vuole eseguire la simulazione; in totale si forniscono in in ingresso 12 parametri scalari alla volta.
Sempre in riferimento alla figura B.1, il primo blocco é il sottosistema che contiene tutta la codifica degli algoritmi del problema della *cinematica inversa*, di *accoppiamento di giunto* e di *generazione di traiettorie*. Le uscite di questo blocco diventano gli ingressi di quello immediatamente successivo, convertendo i valori in uscita del precedente (espressi in gradi) in coppie da assegnare ai motori, modellati da appositi blocchi. Il secondo blocco contiene tutta la *catena cinematica* del manipolatore, con *giunti*, *bracci*, *encoders* e *sensori di posizione assoluta*. Questo modello ha 7 uscite: le prime 6 sono gli angoli "misurati" dagli *encoders* una volta raggiunta la posa finale dal manipolatore (e vengono riportati in ingresso come condizioni iniziali alla simulazione successiva), mentre la settima é la posizione assoluta, misurata dai sensori di posizione, della punta della pinza rispetto la terna base, che confrontata con la posizione desiderata portata in ingresso, fornisce l'*errore di posizione*, valore compreso per buona parte delle simulazioni tra gli ordini di 10^{-13} e 10^{-15}.

4.1.1 Blocco CONTROLLORE

Il blocco CONTROLLORE é il sottosistema (in *Simulink* corrisponde ad un blocco *subsystem*, collocato nella libreria *commonly used blocks*) che processa gli ingressi del sistema (v. fig. B.1), che essendo quantitá scalari sono definiti usando blocchi *constant*, anche questi nella medesima libreria. In questo sottoblocco (riportato in appendice in figura B.2) si possono individuare due parti: la prima, che a partire dagli ingressi, si occupa di effettuare tutte le trasformazioni della modalitá cinematica inversa, e la seconda che si occupa di generare le traiettorie con profilo di velocitá trapezoidale per ciascun giunto. Gli ingressi del sottosistema CONTROLLORE sono i blocchi *constant*, la variabile booleana f usata come *flag* per i blocchi *switch* (viene determinata automaticamente avviando in MATLAB lo script A.1 in appendice) e il valore t_s come vincolo temporale di adempimento della traiettoria; inoltre viene considerato un *clock* di sistema, che suddivide in istanti finiti l'intervallo temporale tra un momento iniziale ($t_i = 0$) e finale (t_s). Oltre a questi sono presenti dei blocchi di uso generale come *input* e *output*, *subsystem*, per modellare ad esempio l'accoppiamento di giunto di cui si é giá parlato nel paragrafo 3.1, *mux* e *demux*, per raggruppare in una stessa linea di collegamento piú informazioni e viceversa, *switch*, per selezionare le linee in funzione della modalitá, *display*, per visualizzare un valore su una linea di connessione, e *gain*, usato dove é resa necessaria una conversione gradi/radianti. In particolare i blocchi piú interessanti peró sono i blocchi *function* e *simout*. Il primo permette di includere nel modello una funzione scritta in MATLAB che processi un dato ingresso u e che restituisca un'uscita y: il numero di ingressi e uscite possono aumentare a piacimento. Il secondo (*simout*) invece salva nel *Workspace* una variabile strutturata in formato *timeseries*, e ció permette di estrapolare e manipolare i dati numerici ottenuti dalla simulazione. In questo sottosistema sono presenti tre importanti blocchi *function* a cui corrispondono i passi indicati per il computo della cinematica inversa nel paragrafo 3.3. Il blocco *WRIST* corrisponde al primo passo del calcolo:

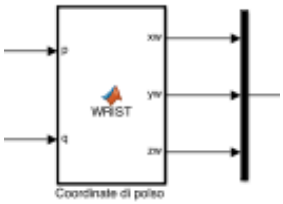

Figura 4.1: Blocco WRIST.

Come si puó vedere in figura 4.1, le linee di ingresso sono semplicemente due: la posizione desiderata (p) e la terna di Eulero (q) per l'orientamento (riportate in radianti per le funzioni trigonometriche) mentre in uscita sono riportate le tre componenti riscalate del centro di polso, riunite tramite un *demux*. Ricordando la formula (3.3), si puó scrivere semplicemente il codice contenuto in questo blocco (v. A.3). Ottenute le coordinate di centro di polso, seguendo il procedimento descritto nel paragrafo 3.3, si definisce un altro blocco *function*, denominato CINV1, che implementa la risoluzione della cinematica inversa per la struttura portante, rappresentato in figura:

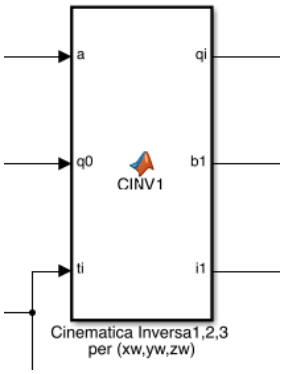

Figura 4.2: Blocco CINV1.

Gli ingressi sono: le coordinate di centro di polso (a) ottenute dal blocco precedente, le condizioni iniziali (q_0) dei primi tre giunti e il clock di simulazione t_i; come uscite si hanno le tre variabili di giunto calcolate da questo blocco, un indice i_1 per indicare (con un blocco *display*) quale delle 4 possibili soluzioni é stata scelta) e una variabile logica b_1 settata al valore 1, che viene portata bassa solo nel caso in cui le coordinate in ingresso non rispettino le condizioni espresse in (3.8). Il codice completo si trova in appendice (v. A.4): in esso é interessante notare il *criterio di scelta* descritto nel paragrafo 3.3.5. Infatti prima si considera il primo giunto, e quale delle due quantitá tra $|q_{1,1} - q_{0,1}|$ e $|q_{1,2} - q_{0,1}|$ sia la piú piccola, poiché ció permette di scegliere tra le soluzione quella che ha minor spunto del motore al primo giunto. Fatto questo resta comunque una coppia con lo stesso valore di q_1, cioé quelle due soluzioni che pur risolvendo il problema cinematico portano l'organo terminale nello stesso punto con due configurazioni differenti, note come configurazioni di *gomito basso* e *gomito alto*. Pertanto tra le due si sceglie quella che ne minimizza il valore in norma, usando la funzione *norm*, presente nelle librerie di MATLAB [15]. Si osservi che all'inizio vi é un costrutto *if* che esprime la condizione (3.8): se questa non é soddisfatta, viene inviato un messaggio d'errore e l'uscita é pari alle condizioni iniziali. Ció indica

sostanzialmente che il manipolatore nella simulazione rimarrá fermo nel punto in cui si trovava. Infine, restano i valori degli ultimi tre giunti del polso sferico, calcolati dal blocco *function* CINV2, che elabora a sua volta i risultati ottenuti dal blocco precedente:

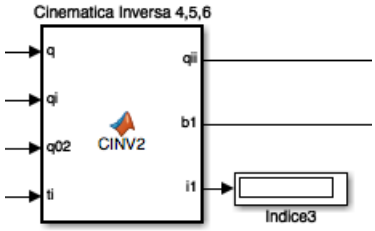

Figura 4.3: Blocco CINV2.

Gli ingressi sono la configurazione di Eulero per l'orientazione (q), le variabili di giunto calcolate per la struttura portante nel blocco precedente(q_i), le condizioni iniziali ($q_{0,2}$) e il tempo di simulazione (t_i); si rimanda al codice completo in appendice (v. A.5). Anche in questo caso si é dovuto implementare un *criterio di scelta* e in questo caso si sceglie direttamente la soluzione che minimizza il suo valore in norma [15]. Tra le uscite, oltre alle tre variabili di polso sferico, é compreso, in analogia a quanto fatto per il blocco CINV1, un indice che riveli quale delle due terne possibili é stata scelta e una variabile logica b_1 settata a 1, che viene mandata bassa se le soluzioni ai giunti non soddisfano le condizioni discusse nel paragrafo 3.3.5. In particolare questa variabile booleana, insieme alla sua corrispettiva in CINV1, sono processate da un blocco AND il cui risultato viene usato per controllare un blocco interruttore. Se nella compilazione tutte le condizioni sono rispettate, poiché questi valori logici sono entrambi settati alti nel software, il modello simulerá normalmente, mentre, se anche una delle due viene mandata bassa, si avrá che, per via del blocco *switch*, al posto dei valori finali ai giunti, saranno selezionate le condizioni iniziali, lasciando di fatto fermo il manipolatore in ogni istante della simulazione.

4.1.2 Generazione di traiettorie punto-punto

Una volta computati gli angoli da assegnare ai giunti (nella modalitá cinematica inversa o definiti direttamente in ingresso se in modalitá cinematica diretta) occorre generare una traiettoria che porti l'organo terminale nella posizione desiderata: per fare ció si é implementato l'algoritmo dell'interpolazione punto-punto descritto del paragrafo 2.3.1. Si osservi che in ognuno dei due sottosistemi per la generazione di traiettorie (visibili sulla destra della figura B.2) vi sono tre ingressi e un'uscita che ingloba, a gruppi di tre, le tre traiettorie per

$\tilde{\boldsymbol{\theta}}$ e $\hat{\boldsymbol{\theta}}$; i tre ingressi per un singolo blocco sono rappresentati dalle tre condizioni iniziali (che rappresentano, per la relativa traiettoria, il punto di partenza), i tre valori finali ai giunti (che analogamente sono il punto di arrivo), e il vincolo temporale t_s, che é il tempo che deve essere impiegato dall'organo terminale per il raggiungimento della posa finale (costante, $t_s = 300$).

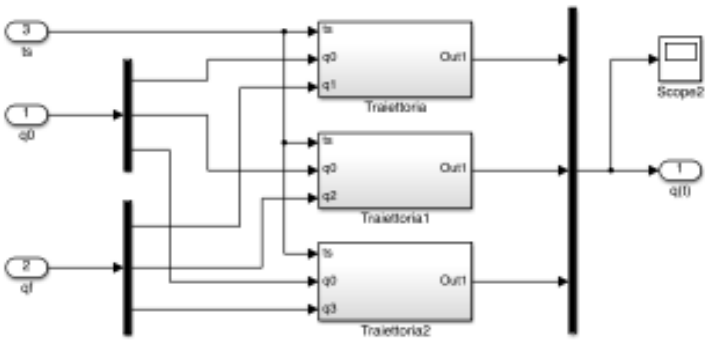

Figura 4.4: Schema dei blocchi di generazione di traiettoria.

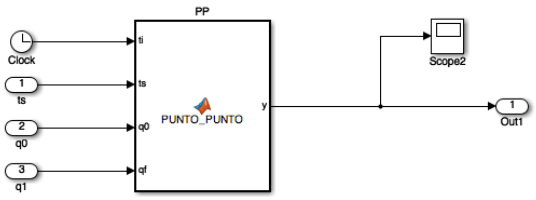

Figura 4.5: Blocco *function* di generazione di una delle 6 traiettorie.

In entrambi questi blocchi, gli ingressi vengono *demuxati* per ciascuna delle 6 traiettorie in un blocco *subsystem* contenente la *function* in figura 4.5 che implementa l'algoritmo *punto-punto* per una singola traiettoria tra valore iniziale e finale. Gli ingressi della *function*, chiamata PUNTO-PUNTO, sono il valore iniziale e finale da imporre al giunto, il vincolo temporale t_s, e il campione temporale t_i, scandito dal *clock* (fig. 4.5). Per via di quest'ultimo, l'intero codice verrá compilato per ogni istante t_i, e allo scorrere del *clock* una delle condizioni logicamente disgiunte di uno tra i costrutti *if* (per chiarezza, v. il codice A.6 in appendice) sará soddisfatta, mentre tutti gli altri segmenti di traiettoria saranno considerati nulli. Alla fine l'uscita y é data dalla somma istante per istante dei

vari segmenti, in cui solo uno di questi contribuirá nel corrispettivo intervallo con un valore diverso da zero, mentre gli altri resteranno nulli, come si puó vedere piú chiaramente nel codice A.6 in appendice. Ad esempio nell'intervallo temporale del primo tratto parabolico, il segmento lineare e il secondo tratto parabolico sono nulli, in modo che alla fine sommando si ottenga solo il profilo desiderato in quell'intervallo. Ripetendo questo ragionamento per ogni intervallo, la traiettoria finale viene ottenuta sommando insieme i singoli segmenti di traiettoria.

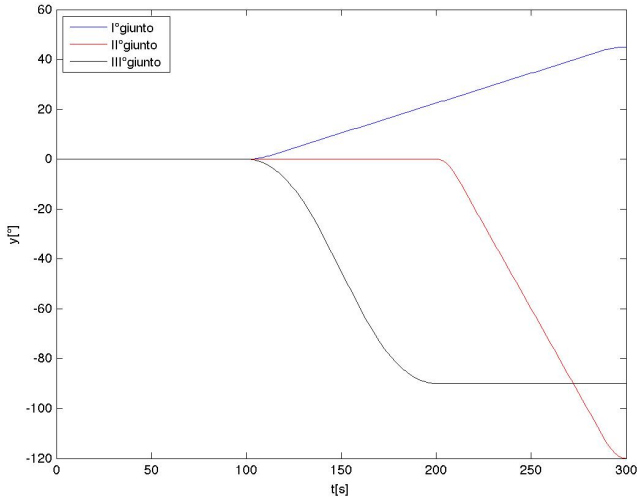

Figura 4.6: Traiettorie per i giunti della struttura portante.

Nel codice vengono calcolati i valori di t_c e $\ddot{q}_c$ rispettivamente usando le (2.37) e la (2.38), e per ogni costrutto *if* sono state usate le equazioni del sistema (2.39). Cosí é possibile ottenere i 6 riferimenti necessari all'attuazione dei giunti in realtá virtuale, come mostrato meglio nei grafici precedenti (figg. 4.6, 4.7). In questi si puó vedere come le traiettorie non siano uniformemente distribuite in tutta l'ampiezza dell'intervallo $[0, t_s]$: per prima cosa vengono attuati i giunti del polso sferico che raggiungono il loro valore finale esattamente a metá dell'intervallo di simulazione. Definendo poi t_f come la terza parte di t_s (per questo si sceglie $t_s = 300s$), per $t = t_f$ incomincia a muoversi anche il primo giunto che raggiunge il suo valore finale in $t = t_s$ e il terzo giunto che invece lo raggiunge in $t = 2t_f$. Non appena quest'ultimo ha raggiunto il suo corrispondente valore finale, incomincia a muoversi immediatamente dopo il secondo giunto, che raggiunge il suo valore finale in $t = t_s$. Quindi il codice in appendice (A.6, valido per la generazione della traiettoria per il quarto giunto) dovrá essere particolarizzato a seconda dei casi, includendo traiettorie a valori costanti

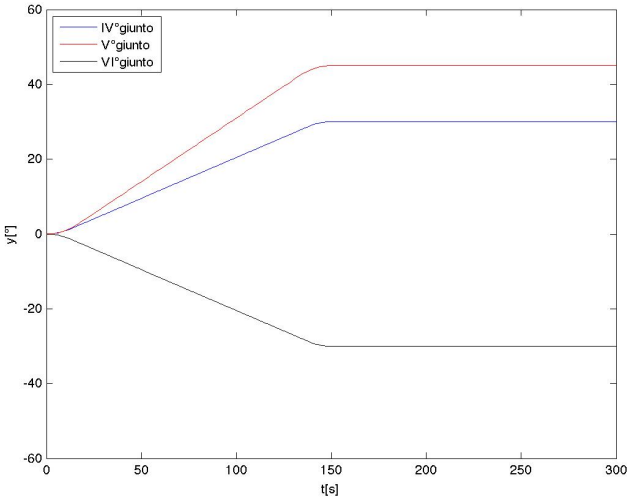

Figura 4.7: Traiettorie per i giunti del polso sferico.

(giunto fermo) quando altri giunti vengono attuati; ció puó essere ottenuto caso per caso semplicemente da costrutti *if* aggiuntivi per ogni tratto in piú, dove verrá specificato il valore costante da imporre al giunto nel'intervallo di tempo desiderato quando t_i scorre all'interno dell'intervallo desiderato.

4.1.3 Accoppiamento di giunto

Nel paragrafo 3.1 si era annoverata tra le caratteristiche meccaniche del MANUS la dipendenza tra secondo e terzo; questo dettaglio costruttivo é stato incluso nel modello del manipolatore con un blocco sottrattore tra ingresso 3 e 2 del blocco MANUS (v. fig. B.1). Riportando quanto detto, se il secondo giunto viene attuato di uno spostamento Δq_2, il terzo giunto subisce una rotazione uguale e contraria per mantenere in asse il centro di polso [5], esprimibile in questi termini:

$$\Delta q_3' = \Delta q_3 - \Delta q_2 \tag{4.1}$$

D'altro canto, se il terzo viene attuato il secondo non risente alcun spostamento. Tuttavia con questo legame di dipendenza, chiamato appunto *accoppiamento di giunto*, i valori in ingresso non saranno gli stessi osservati in uscita, proprio a causa del blocco sottrattore, che porterá ad un valore finale di q_3 diverso da quello assegnato in ingresso. Allora occorre effettuare una *compensazione* nel blocco CONTROLLORE per quanto riguarda la terna $\tilde{\theta}$ della struttura portante, sia per i valori finali che per le condizioni iniziali, introducendo un blocco *subsystem* che modelli questo aspetto, contenente il seguente schema:

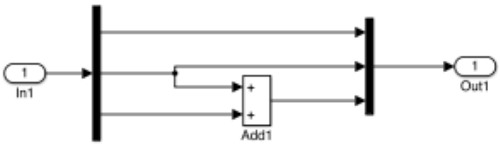

Figura 4.8: Blocco di compensazione.

Questa compensazione giustifica la scelta di attuare per ultimo il secondo giunto: infatti se il terzo giunto si muove mentre il secondo resta costante, allora $\Delta q_2 = 0$ e non vi é un movimento pari e contrario. D'altra parte per la compensazione, il valore finale che viene assegnato al giunto é $q_3' = q_3 + q_2$, il ché non porterebbe l'organo terminale nella posizione desiderata. Tuttavia, con riferimento alla figura 4.6, si puó vedere che nel momento in cui incomincia a muoversi il secondo giunto, il terzo resta costante sulla quota $q_3 + q_2$: ricordando peró che l'ingresso al terzo motore viene scalato di una quantitá $-\Delta q_2$ per la caratteristica intrinseca dell'accoppiamento di giunto (v. fig. B.1), se il secondo giunto si muove, il terzo giunto si riporterá al valore desiderato, muovendosi in opposizione con uno spostamento pari e contrario di ampiezza, esattamente pari a $-q_2$.
Ció che si osserva in simulazione é il movimento del terzo giunto indipendentemente dal secondo, mentre nel momento in cui viene attuato il secondo si osserva il movimento "di richiamo" tra i due bracci con il risultato di mantenere il centro del polso in asse.

4.1.4 Blocco "MANUS"

Questo sottosistema contiene la parte "meccanica" del manipolatore, prendendo in ingresso le traiettorie generate dal blocco descritto nel precedente paragrafo. Pertanto é stata utilizzata la libreria dedicata *SymMechanics* che contiene i seguenti blocchi fondamentali nella modellazione della catena cinematica:

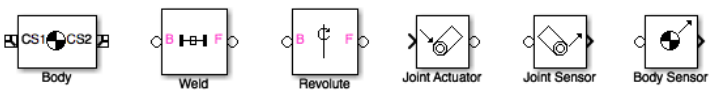

Figura 4.9: Blocchi della struttura meccanica del MANUS.

Il blocco *Body* modella fisicamente un braccio della catena: in esso vanno speci-

ficate *massa*, *tensore d'inerzia* e alcuni punti per definire la geometria del corpo. Questi punti sono definiti rispetto ad un riferimento che deve essere esplicitamente specificato, e a loro volta possono fungere da nuovi riferimenti, intesi come punti connessione tra gli elementi della catena cinematica. Dal momento che si vuole ottenere un modello cinematico, e non dinamico, grandezze come massa e tensore d'inerzia perdono d'importanza. Per quanto riguarda le geometrie, la grandezza piú rilevante da definire é il parametro a_i della convenzione di Denavit-Hartenber, cioé la lunghezza di un braccio, mentre tutte le altre grandezze sono state definite per dare consistenza ai bracci, che altrimenti sarebbero simulati in realtá virtuale come semplici aste. Dunque la catena é stata realizzata collegando i vari riferimenti distanti come indicati nella tabella 3.5, ma la forma dei corpi é abbastanza distante da quella reale, come si puó vedere ad esempio in figura 3.4 (nella simulazione si sono usati principalmente parallelepipedi e solidi prismatici, mentre in realtá sono principalmente cilindrici per ridurre l'inerzia). I blocchi *Weld* e *Revolute* rappresentano le connessioni tra corpi e quindi possono collegare due blocchi *Body* consecutivi: rispettivamente, rappresentano un *vincolo rigido*, ossia due corpi incernierati tra loro con zero gradi di libertá residui, e un *giunto rotoidale*. Per quest'ultimo si puó predisporre una coppia ingresso/uscita che prenda in ingresso il segnale dei blocchi *motori*, e invii in uscita il segnale per i *sensori*, modellati rispettivamente dai blocchi *Joint Actuator* e *Joint Sensor*. Infine si usa un blocco *Body Sensor*, che deve essere direttamente connesso ad un punto del corpo preso in considerazione, per "misurarne" la posizione assoluta nello spazio. Per fare ció é lecito chiedersi come definire un riferimento assoluto nel modello. Per fare ció si dispone dei due seguenti blocchi:

Figura 4.10: Blocchi per definire l'*environment*.

Il blocco *Ground* permette di definire un riferimento assoluto, ed é la terna da cui ha inizio la catena cinematica: in questo modello viene fatta coincidere con la sommitá della colonna del manipolatore, dove si trova il primo giunto rotoidale che fa ruotare tutta la base del manipolatore. Unitamente a questo viene usato il blocco *Machine Enviroment* che permette di definire diversi parametri di analisi e risoluzione del modello oltre ai i quali la presenza del campo gravitazionale: ció é importante perché, anche se il calcolo della coppia é automatico, questo

deve tener conto oltre allo spostamento angolare desiderato anche tutte le forze esterne a cui é soggetto il manipolatore, tra cui anche la forza di gravitá. Viene connesso al blocco *Ground* che ha una porta predisposta per l'*environment*. Il sottosistema MANUS completo di tutte le connessioni e blocchi é rappresentato nell'appendice in figura B.3, dove ritroviamo in ingresso le traiettorie di ogni giunto, processate dai blocchi *Joint Actuators*: a questi sono necessarie anche le derivate successive fino al secondo ordine del segnale per calcolare la coppia da imporre. Inoltre osservando attentamente il modello si puó vedere come tra il secondo e terzo ingresso vi sia un legame espresso dal blocco sottrattore: come giá detto, questo esprime la dipendenza cinematica derivante dall'accoppiamento di giunto come *vincolo intrinseco* del manipolatore, giustificando dunque le scelte di cui si é ampiamente discusso nel paragrafo precedente.

4.2 Modello a inseguimento di traiettoria

Il modello *inseguimento di traiettoria* é il secondo sviluppato in questo lavoro di tesi, ed é il modello cinematico con cui si possono effettuare simulazioni di asservimento di un percorso una volta pianificate le traiettorie in spazio operativo. Lo schema "a macroblocchi" del modello somiglia a quello precedente (si veda la figura B.4 in appendice) e naturalmente eredita gran parte delle caratteristiche del blocco che descrive la struttura meccanica, mentre, come spesso ripetuto, la parte di generazione delle traiettorie é stata separata da questo. Infatti, per questo secondo modello non basta inserire in ingresso un numero ristretto di parametri (come per il primo) per generare a tempo di simulazione i riferimenti inviati ai blocchi motori, ma occorre definire un insieme discreto di punti che individuino un percorso in spazio di lavoro, per cui é difficile conciliare in un unico modello sia la parte computazionale che di simulazione. Per questo, per prima cosa si deve assegnare un *set point*, ottenuto da un'insieme di campioni calcolati dalle equazioni parametriche della curva che si intende seguire con l'organo terminale (nell'appendice sono giá codificate alcune curve note, ma se ne possono generare di qualsiasi tipo cambiando lievemente il codice). Poi occorre processarlo con l'algoritmo a tempo discreto di inversione (implementato nel A.9.1 nell'appendice), il quale traduce l'informazione da spazio di lavoro a spazio operativo, implementando insieme le tecniche di inversione e interpolazione(§§2.2,2.3).
Per generare un qualsiasi riferimento di traiettoria in spazio operativo occorre richiamare dalla *Command Window* la *function* ALGINV1, che prende in ingresso i seguenti parametri: c, che permette di scegliere una delle possibili traiettorie dal set di cammini definiti in appendice, dt, il passo con cui si vogliono discretizzare le traiettorie, dT, il passo con cui saranno fornite le leggi in spazio operativo

usando l'interpolazione *spline* (v. codice A.8, *function* SPLINE) e t_f che identifica l'ampiezza dell'intervallo di parametrizzazione della curva. Restituisce in uscita un vettore contenente le 6 componenti ai giunti della configurazione iniziale q_0, piú 6 diverse variabili strutturate in formato *timeseries* che contengono le leggi orarie che devono essere rispettate dai rispettivi giunti affiché l'organo terminale si muova sul percorso di moto pianificato. Queste, salvate nel *Workspace* vengono automaticamente interpretate dal modello a tempo di simulazione dai 6 blocchi *"from Workspace"* (fig. 4.11) e processate dal primo macroblocco, indicato con DATA: la funzione di questo é semplicemente quella di fornire un segnale "leggibile" dagli attuatori del blocco successivo, convertendo gli ingressi in segnali d'uscite con derivata fino al secondo ordine. Dal momento che la *function* ALGINV1 fornisce le velocitá di giunto per l'inseguimento di una certa traiettoria (per inciso, il MANUS é controllato in velocitá [4], [5], [11]), occorre che questa venga derivata da un blocco *derivatore* per avere il segnale di accelerazione, e un blocco *integratore* per la posizione angolare:

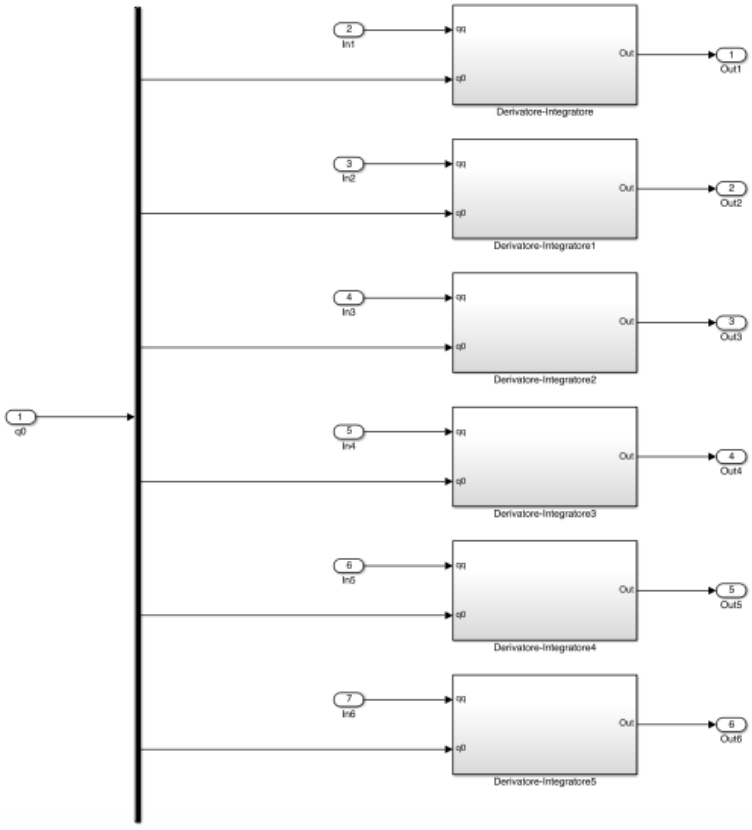

Figura 4.11: Schema interno del blocco DATA.

In particolare, occorre portare in ingresso anche le condizioni iniziali (fornite

da ALGINV1) relative ad ogni giunto, per ottenere correttamente dal segnale di velocitá quello di posizione angolare. Ognuno dei blocchi *subsystem* della figura 4.11 contiene dunque un blocco *derivatore* e un blocco *integratore* con le condizioni iniziali di giunto, connessi naturalmente come segue in questo singolo modulo:

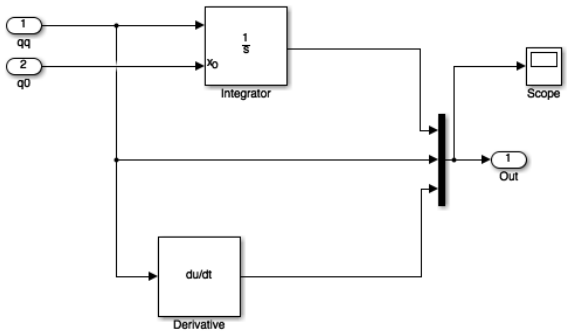

Figura 4.12: Blocco "derivatore-integratore"

Infine il segnale viene portato in ingresso al "macroblocco" della struttura meccanica del manipolatore MANUS, che eredita come giá detto esattamente tutte le caratteristiche dell'analogo blocco del primo modello.

4.3 Modello Master-Slave

Il modello *Master-Slave* é il terzo ed ultimo sviluppato per questa tesi il quale permette di verificare l'obiettivo finale di questo elaborato, cioé la simulazione del movimento coordinato di due manipolatori, posti in posizione frontale, vincolati a muoversi su una circonferenza, partendo da posizioni diametralmente opposte.
Il nome é dovuto al fatto che, con l'algoritmo descritto nel paragrafo 2.2.3 é possibile pianificare l'arco di circonferenza per uno dei due (*Master*) e ottenere le traiettorie in spazio operativo direttamente per l'altro (*Slave*). Anche in questo caso si é adottato l'approccio del modello precedente, ossia di effettuare in separata sede tutta la parte di pianificazione delle traiettorie, dandole direttamente come ingressi da *Workspace* attraverso blocchi *simin*. Sostanzialmente lo schema del modello, in figura 4.13, riporta con qualche aggiustamento (gli ingressi dei segnali di traiettoria sono raggruppati tutti su un'unica linea) lo schema precedente insieme a un suo duplicato.
L'unica differenza la si trova all'interno del blocco della struttura meccanica

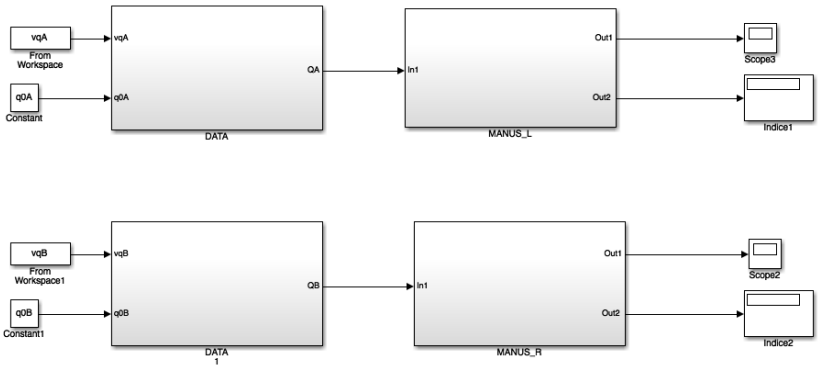

Figura 4.13: Schema del modello Master-Slave

del secondo manipolatore MANUS (*Slave*), nel quale viene modificata la posizione assoluta (blocco *environment*) del sistema di riferimento, posta a un metro di distanza lungo la direzione x rispetto al primo manipolatore: ció per poter osservare in simulazione i due manipolatori in posizione frontale, i cui organi terminali non si toccano reciprocamente. Anche qui, l'implementazione dell'algoritmo per la pianificazione con due manipolatori, che prende il nome di ALGINV2, segue similmente il codice del precedente con alcune differenziazioni: gli ingressi si riducono a due, cioé t_f e dt che rispettivamente indicano l'ampiezza dell'intervallo temporale e del passo usato nella discretizzare tale intervallo. Inoltre nell'implementazione di questo particolate algoritmo di inversione, si é giá predisposta la curva di pianificazione, che é un arco di circonferenza: in virtú di questa scelta, si puó considerare come variabile operativa all'interno del codice il vettore posizione (indicato con **VP**) tra gli organi terminali dei due manipolatori in cui deve essere minimizzato l'errore di posizione per avere un buon inseguimento da entrambi i manipolatori (per il codice completo, v. A.9.2 in appendice).

Capitolo 5

Analisi dei risultati

In questo capitolo verranno presentati i risultati delle simulazioni dei modelli descritti nel capitolo precedente, che trovano qui la *validazione* degli algoritmi di risoluzione del problema cinematico inverso e di pianificazione in spazio operativo. In particolare, per il primo, l'obiettivo principale é la simulazione del moto del manipolatore da un punto ad un altro indipendentemente dal cammino che segue; nel secondo invece, l'inseguimento di una traiettoria pianificata, mediante l'assegnazione di profili in spazio operativo. Infine per l'ultimo, gli obiettivi sono gli stessi del caso precedente, riguardanti peró due manipolatori che devono effettuare un movimento coordinato. Per ognuno di essi l'indice di verifica della bontá del modello é l'*errore*, definito come la differenza tra le grandezze desiderate e quelle osservate in uscita in precise situazioni operative. Va osservato che, come ogni simulazione, anche quelle effettuate sono determinate dalla struttura e dalle caratteristiche del modelli utilizzati e dai vincoli imposti: pertanto non possono avvalersi di generalitá assoluta, né tanto meno considerasi prive di errori se non suggellate da un'analisi condotta sull'oggetto delle simulazioni nella realtá fisica.

5.1 Analisi del modello punto-punto

Seguendo l'ordine del capitolo precedente, il primo modello preso in analisi é il modello punto-punto. Una volta aperto in ambiente *Simulink*, occorre fornire al modello gli ingressi per la simulazione, avviando in MATLAB lo *script* descritto nel codice A.1 in appendice; a questo punto si sono definiti tutti gli ingressi necessari per avviare correttamente la simulazione. Detto questo peró occorre tenere in considerazione alcune impostazioni del *simulatore*. Aprendo il modello ci si troverá di fronte la seguente schermata:

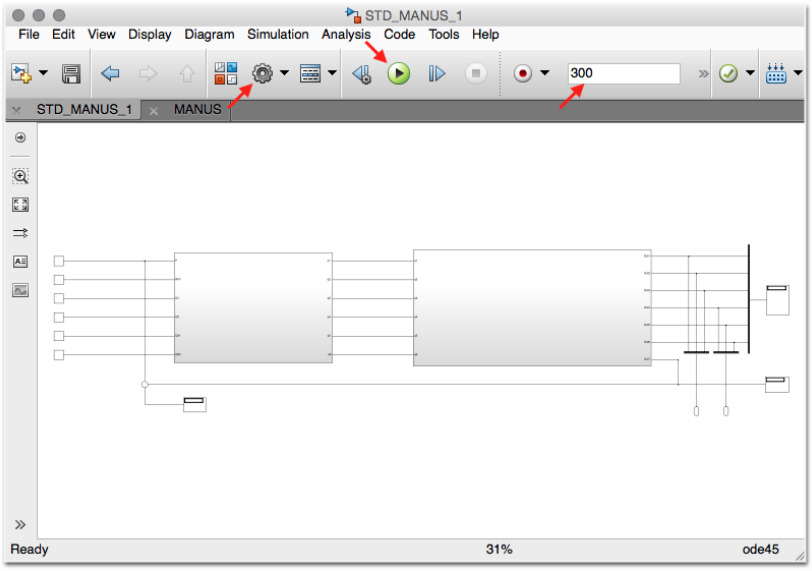

Figura 5.1: Schermata iniziale del modello.

Nella *text-box* a destra (ultima freccia a destra) del menú é possibile impostare il tempo di osservazione del modello in simulazione: puó essere arbitrariamente impostato e ponendo *inf* la simulazione proseguirá finché non verrá manualmente interrotta. Nel paragrafo 4.1.2 é stata giá fornita la motivazione della scelta del tempo di simulazione $t_s = 300s$. La prima freccia a sinistra indica l'icona (raffigurante un ingranaggio) delle impostazioni (*Configuration Parameters*): cliccandovi si potrá accedere ad una serie di impostazioni manuali di parametri di simulazione, di cui i piú importanti corrispondono alle sezioni *Solver* e *Data Import/Export*. Con la prima sezione, si otterrá la schermata rappresentata in figura in figura 5.2, in cui le freccie blu indicano i principali settaggi da impostare. Nella seconda sezione, la schermata che si apre é mostrata in figura 5.3: in tal caso non va spuntata nessuna opzione ad eccezione del campo *Output*, dove, una volta selezionato, é possibile nominare delle variabili su cui verranno salvate le uscite del modello. In questo caso si é deciso di nominare queste variabili con lo stesso denominatore usato per le condizioni iniziali, in modo che, ad ogni simulazione successiva si dispongano di condizioni iniziali pari alle configurazioni finali precedentemente raggiunte. Infine, sempre nella schermata dei *Configuration Parameters*, nella sezione *SimMechanics 1G*, occorre spuntare l'opzione "*Show animation during simulation*" per poter visualizzare in realtá virtuale il risultato della simulazione del modello, che é adesso pronta per essere avviata cliccando sul pulsante di *start* in figura 5.1.

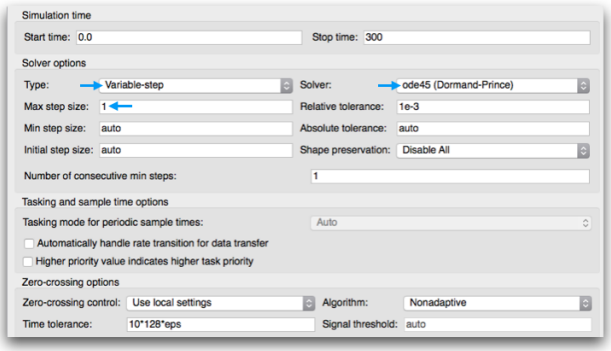

Figura 5.2: Sezione *Solver*.

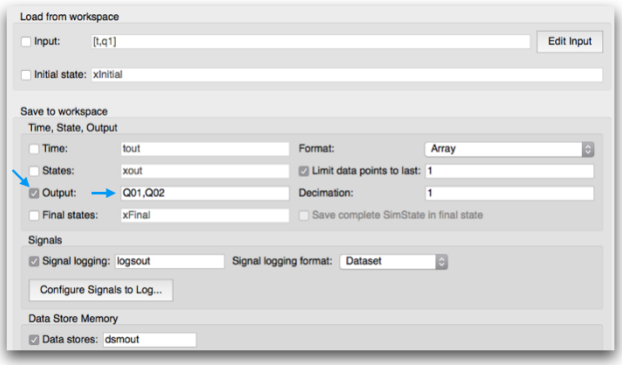

Figura 5.3: Sezione *Data Import/Export*.

Per quanto riguarda questo modello, l'analisi della sua correttezza riguarda il calcolo delle configurazioni finali in cinematica inversa: infatti in questo caso non interessa la traiettoria seguita, ma solo la posizione finale raggiunta. In tutte le simulazioni effettuate, si osserva che, nonostante la non-unicitá della soluzione, ognuna di queste portano l'organo terminale nell'intorno della posizione desiderata con una precisione dell'ordine di 10^{-13}, come riportato dall'errore di posizione nella seguente figura 5.4:

Figura 5.4: Errore di posizione.

5.2 Analisi del modello a inseguimento di traiettoria

Per il secondo modello, l'analisi viene condotta valutando l'*errore* con cui vengono tracciate le traiettorie in simulazione a partire dai profili in spazio operativo forniti dagli algoritmi di pianificazione, ampiamente discussi nei capitoli 2 e 4; per semplificare la trattazione ci si é limitati a considerare i giunti della struttura portante, lasciando costanti nel tempo quelli del polso sferico. In questo caso va inserito nella *text-box* di figura 5.1 un tempo di simulazione pari al t_f, ossia quello stesso parametro della *function* ALGINV1. Occorre anche impostare il simulatore nella sezione *solver* con il *fixed-step solver*. La prima traiettoria testata é la *circonferenza*, in figura 5.5:

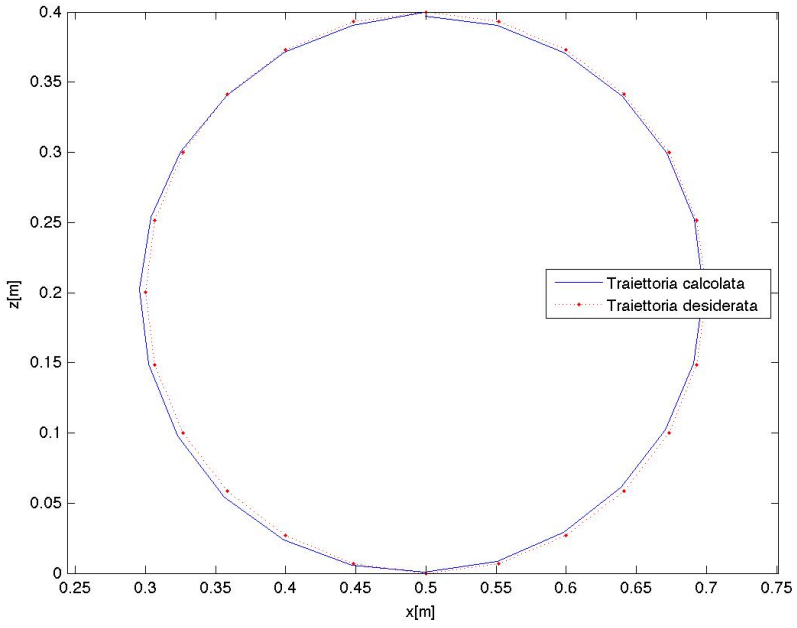

Figura 5.5: Inseguimento per una curva di circonferenza.

Come si puó osservare, il tratto blu (la traiettoria calcolata) segue in maniera abbastanza fedele il tratto rosso (la circonferenza pianificata), come si era discusso nel paragrafo 2.2.2. Tuttavia, mentre il tratto rosso é stato generato imponendo banalmente le equazioni parametriche di una circonferenza appartenente a uno dei sottospazi di lavoro (precisamente al piano xz, con $y = 0$), l'altro profilo viene ottenuto dall'organo terminale seguendo le seguenti leggi pianificate in spazio operativo e interpolate con *spline*:

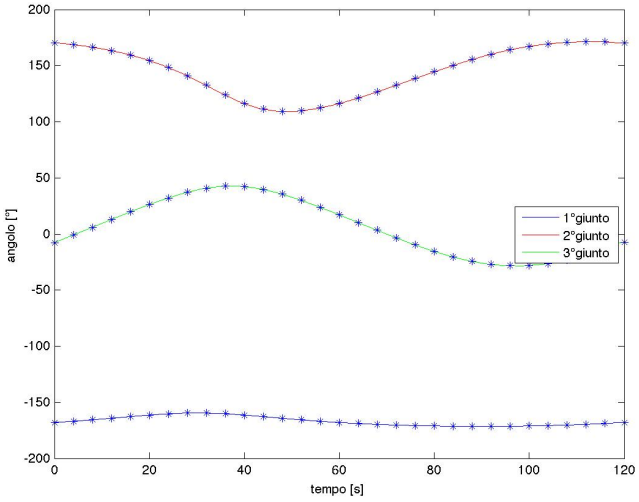

Figura 5.6: Interpolazione spline in spazio operativo.

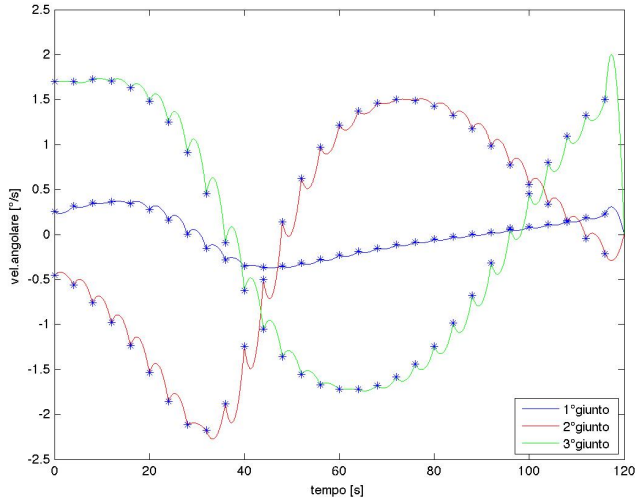

Figura 5.7: Profili di velocitá.

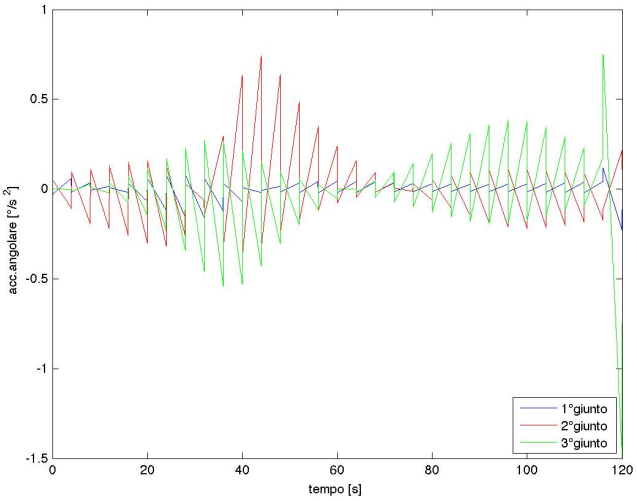

Figura 5.8: Profili di accelerazione.

Nei grafici precedenti si possono osservare le derivate successive dei profili di pianificazione in spazio operativo. Nel primo é rappresentata la legge oraria che ciascun giunto deve rispettare per l'inseguimento del moto circolare da parte dell'organo terminale e sono indicati con asterischi l'insieme di valori ottenuti dall'algoritmo di inversione cinematica, mentre il tratto continuo é il raccordo effettuato dalle *spline*, che, come si puó vedere, ne minimizzano la curvatura. Nel secondo e terzo si mostra invece l'andamento discontinuo tipico delle derivate successive: il segnale di velocitá (come giá detto é l'ingresso del modello) presenta molte oscillazioni raccordate in punti angolosi che conferiscono ai profili di accelerazione i caratteristici andamenti impulsivi (figura 5.8).
Queste analisi possono essere condotte in maniera del tutto analoga per il *set* di cammini in appendice, composto da *Circonferenza* (v. A.7.1), *Rodonea* (v. A.7.2), *Spirale di Archimede* (v. A.7.3) e *Quadrato* (v. A.7.4). Ció che si é osservato é un buon inseguimento per tutte le traiettorie pianificate, come nel caso del grafico in figura 5.5. Tuttavia alcuni fattori vanno tenuti in considerazione, in particolare gli ingressi dell'algoritmo di pianificazione ALGINV1: non bisogna infatti assegnare un passo di campionamento dt troppo ampio rispetto all'ampiezza finale dell'intervallo di parametrizzazione della curva t_f; infatti la matrice $\mathbf{K}$ della formula (2.26) é *definita positiva* e *diagonale*, con autovalori (assegnabili ad arbitrio) posti in dipendenza del reciproco del passo dt (vd. codice A.9.2): questi autovalori non devono abbassarsi sotto un certo margine di stabilitá della simulazione stessa.

5.3 Analisi del modello Master-Slave

Analogamente a quanto visto nel modello precedentemente descritto, anche nel caso del modello *Master-Slave* la bontá della simulazione é rappresentabile dall'*errore* ottenuto confrontando gli andamenti delle traiettorie pianificate e quelle calcolate con l'algoritmo ALGINV2. In questo caso il parametro di stima dell'errore é il *vettore posizione* tra i due organi terminali, che nel loro moto devono mantenere una distanza fissa. Come giá detto, é sufficiente pianificare una traiettoria per il primo manipolatore (*Master*) per avere automaticamente identificati i corrispettivi profili in spazio operativo del secondo (*Slave*). Con le impostazioni settate in maniera identica alle precedenti nella sezione *Configuration Parameters* e definendo come ingressi della *function* ALGINV2 (v. §4.3) un tempo di simulazione $t_s = 30s$ e un passo $dt = 3s$ (10 campioni) si ottiene il seguente arco di circonferenza:

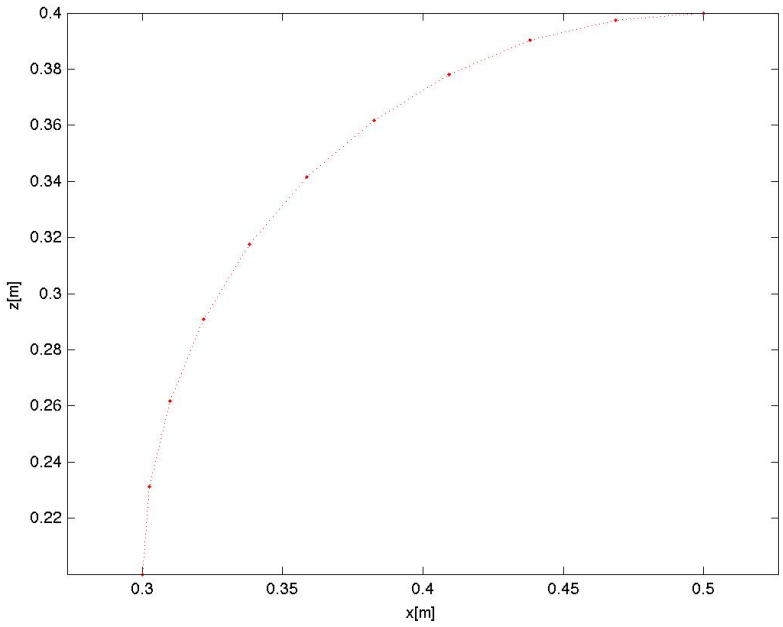

Figura 5.9: Traiettoria pianificata per il *Master*.

In base a quanto giá descritto, i due manipolatori, partendo da posizioni diametralmente opposte , devono effettuare un movimento coordinato, in modo che, partendo dai "poli" opposti della circonferenza pianificata, all'istante finale si trovino allineati sulla linea dell'equatore, sempre in posizione opposta.

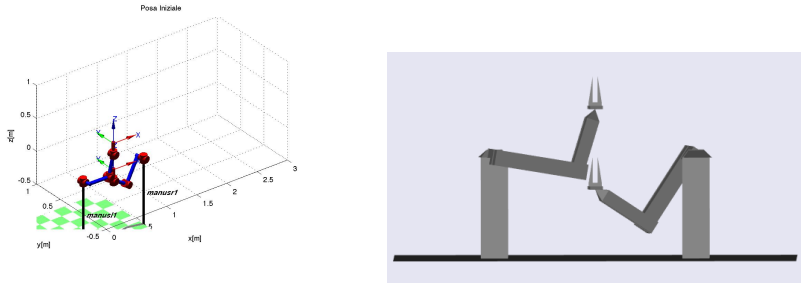

Figura 5.10: Posizione iniziale.

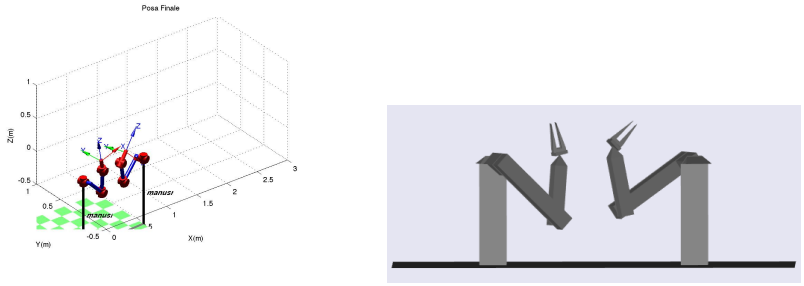

Figura 5.11: Posizione finale.

Come si puó vedere dalle figure precedenti, le simulazioni (figure a destra) confermano quanto predetto in fase di pianificazione (figure a sinistra). In particolare per questa pianificazione si é usato il software di *Robotics Toolbox*, gratuito e scaricabile dal sito omonimo [16] di P.Corke. Questo *tool* é uno strumento per la visualizzazione di strutture esemplificative di manipolatori posti in particolari configurazioni (nel nostro caso iniziali e finali) con annessi sistemi di riferimento solidali con gli organi terminali; inoltre, scaricandola si possono ottenere giá precompilate come *function* gran parte degli strumenti fondamentali in robotica (e.g. Jacobiano, Parametri DH, trasformazioni omogenee, etc.). Una volta eseguito il *download* (dal sito [16]) come *script* su MATLAB, é possibile richiamare tutte le *function* di questo *tool*, specialmente per la codifica degli algoritmi di inversione (vd. A.9.1, A.9.2).
A questo punto si puó effettuare la simulazione e verificare in tempo reale la pianificazione, ricordando che le configurazioni di polso sferico rimangano per entrambi costanti nel tempo: in effetti, bisogna osservare il movimento del centro di polso per rendersi conto dell'effettiva correttezza della simulazione. Un'ulteriore verifica del fatto che l'inseguimento di traiettoria venga raggiunto in simulazione con errore di posizione relativamente basso ed accettabile, é mostrata nel seguente grafico:

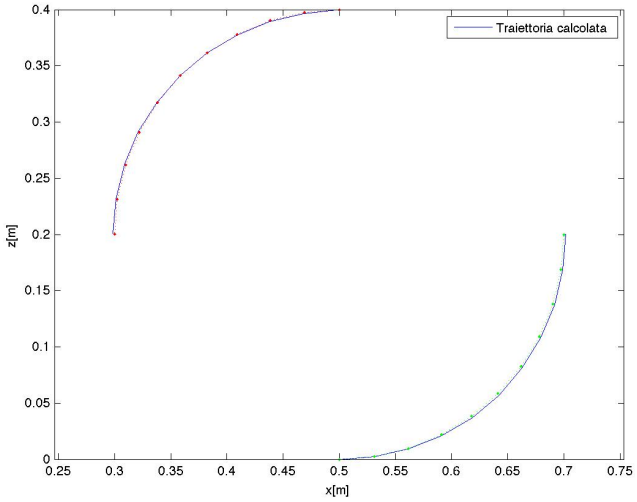

Figura 5.12: Inseguimento dei due manipolatori sul piano xy.

É importante osservare come entrambe seguano la *stessa* circonferenza pianificata, mostrando peró in spazio operativo profili *differenti* (figure 5.13, 5.14):

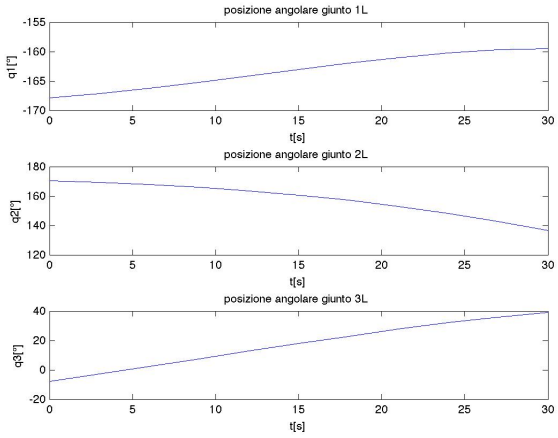

Figura 5.13: Profili della struttura portante del *Master*.

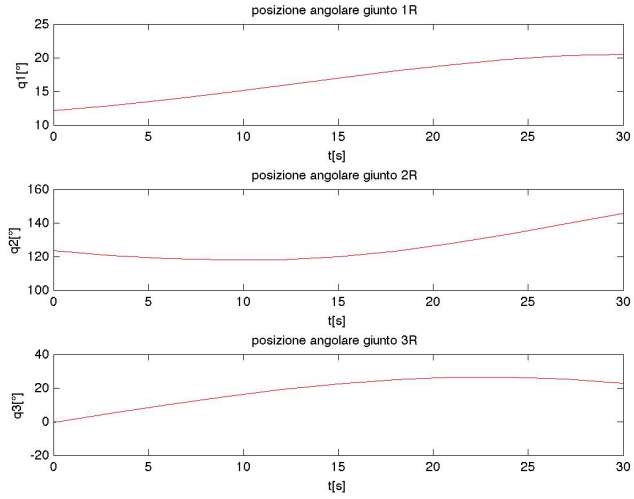

Figura 5.14: Profili della struttura portante dello *Slave*.

Capitolo 6

Conclusioni

Riportando quanto delineato nell'introduzione, l'obiettivo di questa tesi é lo studio e la pianificazione di traiettorie di un manipolatore antropomorfo in un ambiente di simulazione. Da quanto si é discusso nell'elaborato e soprattutto da quanto emerge dal capitolo dell'analisi dei risultati, é possibile affermare che:

- Le grandezze desiderate in ciascuno dei modelli sono state osservate in simulazione con errori piccoli e accettabili, dando dunque un resoconto positivo riguardo la bontá e la validitá dei modelli in simulazione.
- In particolare per gli ultimi due modelli, tutte le traiettorie pianificate sono state seguite in simulazione con performance confrontabili e discreta fedeltá di inseguimento.

Nello sviluppo di questa tesi, il problema della modellazione é stato risolto suddividendo il sistema come se fosse composto da sottosistemi piú semplici, autonomi, che interconnessi gli uni agli altri hanno portato alla modellazione su livelli di complessitá maggiori. In questo modo ad esempio é stato via via possibile trattare la pianificazione di traiettorie e il movimento coordinato di due manipolatori, partendo da modelli piú semplici, come quelli della struttura meccanica del manipolatore o dell'algoritmo del problema cinematico inverso. Dunque per quanto si evince da queste poche righe, questo lavoro di tesi é stato svolto secondo un principio di *modularitá* nell'approccio ai casi di studio, concentrandosi alla risoluzione di problemi non banali, ma di portata limitata, per poi comporli in argomentazioni piú complesse e strutturate. Ovviamente, anche le *approssimazioni* hanno giocato un forte ruolo. La principale é l'aver affrontato tutto il problema della modellazione con approccio puramente cinematico, trascurando le *cause* del movimento dei manipolatori. La mancanza di una dipendenza *causa-effetto* é dunque il maggior *limite* della pianificazioni e

delle simulazioni, che essendo tratte da algoritmi cinematici realizzano gli *effetti* (traiettoria desiderata), ma non tengono conto, nella definizione delle leggi orarie in spazio operativo, di tutte le cause esterne (gravitá, forze esterne) e interne (moti interni, non-linearitá derivanti dall'uso di cinghie, attriti, forze apparenti sui bracci). Inoltre, non conoscendo tutti i dettagli costruttivi, alcuni elementi non sono stati modellati, mentre altri (vd. accoppiamento di giunto) sono considerati solo per un modello in particolare (il primo), venendo trascurati nei successivi per ridurre la difficoltá computazionale. Infine, vi sono alcuni problemi aperti riguardanti gli aspetti delle simulazioni stesse, che possono manifestare diversi comportamenti a seconda delle condizioni assegnate.
I *limiti* di questa tesi possono essere rivisitati in chiave di *sviluppi futuri*: infatti uno studio piú dettagliato delle effettive componenti reali del MANUS (masse, momenti d'inerzia, parametri di rigidezza etc.) potrebbe consentire di studiare, con le opportune linearizzazioni, un modello dinamico in simulazione. Con questo, similmente a quanto fatto in questa tesi, sará possibile pianificare in spazio operativo le leggi orarie che effettivamente portano il manipolatore reale a muoversi lungo le traiettorie desiderate, codificabili su microcontrollori capaci di comunicarle all'elettronica di controllo del manipolatore. Quest'ultimo sarebbe lo sviluppo principe di questo lavoro, in quanto permetterebbe effettivamente di testare la consistenza di ció che si osserva in simulazione attraverso la conferma sperimentale. Detto questo peró, rimane l'utilitá fondamentale dei modelli in simulazione, cioé quella di ottenere una rappresentazione semplificata di un sistema complesso e allo stesso tempo di permettere di testare in maniera ripetuta come il sistema si comporterebbe in determinate situazioni operative.

Appendice A

A.1 Menu' di avvio per il modello punto-punto

```
coder.extrinsic('sprintf');
sprintf('Benvenuto nel menÃź di avvio del modello in simulazione del
    manipolatore MANUS.\n\nPer pilotare in CINEMATICA DIRETTA, inserire
     "D", altrimenti inserire "I" per pilotare in CINEMATICA INVERSA.\
    nPer caricare i dati premere "s", per modificarli "m".\nPer
    concludere la modifica, premere 0.\nInserire il comando "f" (
    fold_out) per ritornare alla posizione di HOMING.\n\n\nNOTE: ogni
    comando di tipo "carattere" deve essere inserito tra apici singoli.
    ')
go=input('Premi "s" per inizializzare ,"m" per modificare: ');
if(go=='s')
mod=input('Premi "D" per Cin.Dir., "I" per Cin.Inv: ');
    if(mod=='I')
    f=1;
    P=input('\nInserisci POSIZIONE DESIDERATA XYZ [mm]: ');
    RPY=input('Inserisci ORIENTAMENTO DESIDERATO RPY [deg]: ');
    Q01=input('Inserisci CONDIZIONI INIZIALI GIUNTI (1,2,3) [deg]: ');
    Q02=input('Inserisci CONDIZIONI INIZIALI GIUNTI (4,5,6) [deg]: ');
    Q1=zeros([1,3]);
    Q2=zeros([1,3]);
    sprintf('\nPuoi avviare la simulazione.')
    else
        f=0;
        P=zeros([1,3]);
        RPY=zeros([1,3]);
        Q1=input('\nInserisci CONFIGURAZIONE FINALE AI GIUNTI (1,2,3) [
            deg]: ');
        Q2=input('Inserisci CONFIGURAZIONE FINALE AI GIUNTI (4,5,6) [
            deg]: ');
        Q01=input('Inserisci CONDIZIONI INIZIALI GIUNTI (1,2,3) [deg]:
            ');
        Q02=input('Inserisci CONDIZIONI INIZIALI GIUNTI (4,5,6) [deg]:
            ');
        sprintf('\nPuoi avviare la simulazione.')
    end
elseif(go=='f')
    f=0;
    Q1=zeros([1,3]);
    Q2=zeros([1,3]);
    P=zeros([1,3]);
    RPY=zeros([1,3]);
    sprintf('\nPuoi avviare la simulazione.')
end
    while(go=='m')
    go=input('Digita un numero [1:4] per modificare una sezione: ');
    switch go
```

```
    case 1
        if(mod=='I')
        P=input('Modifica  POSIZIONE DESIDERATA XYZ [mm]: ');
        else
            Q1=input('Modifica  CONFIGURAZIONE ANGOLARE AI GIUNTI
                (1,2,3) [deg]: ');
        end
    go='m';
    case 2
        if(mod=='I')
        Q=input('Modifica ORIENTAMENTO INIZIALE RPY [deg]: ');
        else
            Q2=input('Modifica  CONFIGURAZIONE ANGOLARE AI GIUNTI
                (3,4,5) [deg]: ');
        end
    go='m';
    case 3
    Q01=input('Modifica CONDIZIONI INIZIALI GIUNTI (1,2,3) [deg]: '
        );
    go='m';
    case 4
    Q02=input('Modifica CONDIZIONI INIZIALI GIUNTI (4,5,6) [deg]: '
        );
    go='m';
    otherwise
    sprintf('\nPuoi avviare la simulazione.')
    go='s';
end
end
```

A.2 Spazio di lavoro

```
n=20;

q1=linspace(-pi,pi,n);
q2=linspace(-pi,pi,n);
q3=linspace(-pi,pi,n);
x=zeros(n,n);
y=zeros(n,n);
z=zeros(n,n);

for i=1:1:n
    Q1=q1(i);
    for j=1:1:n
        Q2=q2(j);
        for k=1:1:n
        Q3=q3(k);
        Q=[Q1,Q2,Q3];
        P=CIND(Q);
        x(i,j,k)=P(1);
        y(i,j,k)=P(2);
        z(i,j,k)=P(3);
        end
    end
end
figure()
xlabel('x[mm]');
ylabel('y[mm]');
zlabel('z[mm]');
for i=1:1:n
    for j=1:1:n
        for k=1:1:k;
        a=x(i,j,k);
        b=y(i,j,k);
        c=z(i,j,k);
        hold on;
        plot3(a,b,c,'.');
        end
    end
end
```

A.3 Coordinate di centro polso

```
function [xw,yw,zw]=WRIST(p,q)

x=p(1);
y=p(2);
z=p(3);

a=q(1); %imbardata (yaw)
b=q(2); %beccheggio (pitch)
c=q(3); %rollio (roll)

l4=160;

xw=x-l4*(cos(a)*sin(b)*cos(c)+sin(a)*sin(c));
yw=y-l4*(cos(a)*sin(b)*sin(c)-sin(a)*cos(c));
zw=z-l4*cos(a)*cos(b);

end
```

A.4 Cinematica inversa per la struttura portante

```
function [qi,i1,b1] = CINV1(a,q0,ti)

coder.extrinsic('sprintf');

x=a(1);
y=a(2);
z=a(3);

l1=105;
l2=400;
l3=320;

q01=q0(1);
q02=q0(2);
q03=q0(3);

q0=[q01,q02,q03];

%calcolo delle quattro soluzioni della cinematica inversa; le quattro
    terne sono espresse nelle variabili Q1,Q2,Q3,Q4.

if((x^2+y^2)>=l1^2&&((x^2+y^2+z^2)>=l1^2+(l2-l3)^2)&&((x^2+y^2+z^2)<=l1
    ^2+(l2+l3)^2))

b1=1;
qi31=asin((x^2+y^2+z^2-l1^2-l2^2-l3^2)/(2*l2*l3));
qi32=pi-qi31;

c31=cos(qi31);
s31=sin(qi31);
c32=cos(qi32);
s32=sin(qi32);

a1=l3*c31;
b1=l3*s31+l2;
a2=l3*c32;
b2=l3*s32+l2;

qi21=2*atan((-b1+sqrt(x^2+y^2-l1^2))/(a1-z));
qi22=2*atan((-b1-sqrt(x^2+y^2-l1^2))/(a1-z));
qi23=2*atan((-b2+sqrt(x^2+y^2-l1^2))/(a2-z));
qi24=2*atan((-b2-sqrt(x^2+y^2-l1^2))/(a2-z));

p1=l3*sin(qi21+qi31)+l2*cos(qi21);
p2=l3*sin(qi22+qi31)+l2*cos(qi22);
```

```
qi11=atan2(y*p1-x*l1,x*p1+y*l1);
qi12=atan2(y*p2-x*l1,x*p2+y*l1);

Q1=[qi11,-qi21,-qi31];
Q2=[qi12,-qi22,-qi31];
Q3=[qi11,-qi23,-qi32];
Q4=[qi12,-qi24,-qi32];

%scelta della soluzione finale;

if(abs(qi11-q01)<abs(qi12-q01))
q1=qi11;
    if(norm(Q1-q0)<norm(Q3-q0))
        q2=Q1(2);
        q3=Q1(3);
        i1=1;
    else
        q2=Q3(2);
        q3=Q3(3);
        i1=2;
    end
else
    q1=qi12;
        if(norm(Q2-q0)<norm(Q4-q0))
            q2=Q2(2);
            q3=Q2(3);
            i1=3;
        else
            q2=Q4(2);
            q3=Q4(3);
            i1=4;
        end
end

else
    b1=0;
    i1=0;
    if(ti==0)
    X=sprintf('\nNON RAGGIUNGIBILE DAL CENTRO DEL POLSO');
    disp(X);
    end
    q1=q01;
    q2=q02;
    q3=q03;
end

qi=[q1,q2,q3];

end
```

A.5 Cinematica inversa per il polso sferico

```
function [qii,b1,i1] = CINV2(q,qi,q02,ti)

coder.extrinsic('sprintf');

%angoli di giunto relativi alla struttura portante
q1=qi(1);
q2=qi(2);
q3=qi(3);

%condizioni iniziali
q04=q02(1);
q05=q02(2);
q06=q02(3);

%angoli aeronautici RPY
a=q(1);
b=q(2);
c=q(3);

r11=sin(q2+q3)*sin(b)+cos(q2+q3)*cos(b)*cos(c)*cos(q1)+cos(q2+q3)*cos(b
    )*sin(c)*sin(q1);
r12=cos(q2+q3)*sin(q1)*(cos(a)*cos(c)+sin(a)*sin(b)*sin(c))-cos(q2+q3)*
    cos(q1)*(cos(a)*sin(c)-cos(c)*sin(a)*sin(b))-sin(q2+q3)*cos(b)*sin(
    a);
r13=cos(q2+q3)*cos(q1)*(sin(a)*sin(c)+cos(a)*cos(c)*sin(b))-cos(q2+q3)*
    sin(q1)*(cos(c)*sin(a)-cos(a)*sin(b)*sin(c))-sin(q2+q3)*cos(a)*cos(
    b);
r21=-sin(c-q1)*cos(b);
r22=-cos(q1)*(cos(a)*cos(c)+sin(a)*sin(b)*sin(c))-sin(q1)*(cos(a)*sin(c
    )-cos(c)*sin(a)*sin(b));
r23=cos(q1)*(cos(c)*sin(a)-cos(a)*sin(b)*sin(c))+sin(q1)*(sin(a)*sin(c)
    +cos(a)*cos(c)*sin(b));
r31=cos(q2+q3)*sin(b)-sin(q2+q3)*cos(b)*cos(c)*cos(q1)-sin(q2+q3)*cos(b
    )*sin(c)*sin(q1);
r32=sin(q2+q3)*cos(q1)*(cos(a)*sin(c)-cos(c)*sin(a)*sin(b))-sin(q2+q3)*
    sin(q1)*(cos(a)*cos(c)+sin(a)*sin(b)*sin(c))-cos(q2+q3)*cos(b)*sin(
    a);
r33=sin(q2+q3)*sin(q1)*(cos(c)*sin(a)-cos(a)*sin(b)*sin(c))-sin(q2+q3)*
    cos(q1)*(sin(a)*sin(c)+cos(a)*cos(c)*sin(b))-cos(q2+q3)*cos(a)*cos(
    b);

q41=atan2(-r23,-r13);
q51=atan2(sqrt(r13^2+r23^2),r33);
q61=atan2(-r32,r31);

q42=atan2(r23,r13);
```

```
q52=atan2(-sqrt(r13^2+r23^2),r33);
q62=atan2(r32,-r31);
Q1=[q41-q04,q51-q05,q61-q06];
Q2=[q42-q04,q52-q05,q62-q06];
b1=1;

%scelta della soluzione

if(norm(Q1)<norm(Q2))
    q4=q41;
    q5=q51;
    q6=q61;
    i1=1;
else
    q4=q42;
    q5=q52;
    q6=q62;
    i1=2;
end

if(q5<-126*(pi/180)||q5>126*(pi/180))
    if(ti==0)
     X=sprintf('\nNON RAGGIUNGIBILE DA POLSO SFERICO');
     disp(X);
    end
    b1=0;
end

qii=[q4,q5,q6];

end
```

A.6 Traiettoria punto-punto

```
function y = PUNTO-PUNTO(ti,ts,q0,qf)

tf=ts/2;
q1=abs(qf-q0);

if((q1>=0)&&(q1<=49))
qc1=((q1)^(0.005)-1);
elseif((q1>49)&&(q1<=90))
    qc1=((q1)^(0.005)-0.985);
    else
    qc1=((q1)^(0.02)-0.98);
end

if(qf<q0)
qc1=-qc1;
end

tc1=0.5*(tf-sqrt(abs((tf^2)-4*((qf-q0)/qc1))));

if((ti>=0)&&(ti<tc1))
y1 = (q0+0.5*qc1*(ti)^2);
else
    y1=0;
end

if((ti>=tc1)&&(ti<tf-tc1))
y2 = (q0+qc1*tc1*(ti-0.5*tc1));
else
    y2=0;
end

if((ti>=tf-tc1)&&(ti<=tf))
y3 = (qf-0.5*qc1*(tf-ti)^2);
else
    y3=0;
end

if((ti>tf)&&(ti<=ts))
y4 = qf;
else
    y4=0;
end

y=y1+y2+y3+y4;

end
```

A.7 Traiettorie per i percorsi

A.7.1 Circonferenza

```
function A = CIRC1(r,rx,rz,t)

a=pi/2;

d=pi/2;

xl=r*cos(a*t/30+d)+rx;
zl=r*sin(a*t/30+d)+rz;

A=[xl,zl];

end
```

A.7.2 Rodonea

```
function A = ROD(r,rx,rz,k,t)

a=pi;

xl=r*sin(a*k*t)*cos(a*t)+rx;
zl=r*sin(a*k*t)*sin(a*t)+rz;

A=[xl,zl];

end
```

A.7.3 Spirale di Archimede

```
function A = SPI_ARC(r,rx,rz,t)

a=pi/2;

d=pi/2;

xl=r*(t/pi)*cos(a*t+d)+rx;
zl=r*(t/pi)*sin(a*t+d)+rz;

A=[xl,zl];

end
```

A.7.4 Quadrato

```
function A = SQUARE(x1,x2,z1,z2,dt,tf)

b=abs(x2-x1);
h=abs(z2-z1);

n=tf/dt;
n1=ceil((b/(h+b))*(n/2));
n2=ceil((b/(h+b))*(n/2));
db=b/n1;
dh=h/n2;

xl1=x1:db:x2;
zl1=linspace(z1,z1,n1);

xl2=linspace(x2,x2,n2);
zl2=z1:dh:z2;

xl3=x2:-db:x1;
zl3=linspace(z2,z2,n1);

xl4=linspace(x1,x1,n2);
zl4=z2:-dh:z1;

xl5=xl1(1);
zl5=zl1(1);

xl=horzcat(xl1,xl2,xl3,xl4,xl5);
zl=horzcat(zl1,zl2,zl3,zl4,zl5);
A=[xl;zl];

end
```

A.8 Splines cubiche

```
function [a0,a1,a2,a3]=SPLINE(GA,FA,vf)

t=GA.time;
GA=GA.data;
FA=FA.data;
n=length(t);
s=size(GA);
n1=s(2);

v=zeros(n,n1);
T=zeros(1,n-1);
a0=zeros(n-1,n1);
a1=zeros(n-1,n1);
a2=zeros(n-1,n1);
a3=zeros(n-1,n1);
v(1:n-1,:)=FA;
v(n,:)=vf;

for i=1:1:n-1
    T(i)=t(i+1)-t(i);
end

 for j=1:1:n1
    for i=1:1:n-1
    a0(i,j)=GA(i,j);
    a1(i,j)=FA(i,j);
    a2(i,j)=(1/T(i))*(3*(GA(i+1,j)-GA(i,j))/(T(i))-2*v(i,j)-v(i+1,j));
    a3(i,j)=(1/T(i)^2)*(2*(GA(i,j)-GA(i+1,j))/(T(i))+v(i,j)+v(i+1,j));
    end
 end

end
```

A.9 Algoritmi di inversione

A.9.1 Algoritmo per un manipolatore

```
function [q0,Q1,Q2,Q3,Q4,Q5,Q6]=ALGINV1(dt,dT,tf,c)

close all;
K=(1/dt)*eye(3,3);
e=zeros(3,1);

r=0.2;
rx=0.5;
rz=0.2;
n=floor(tf/dt+1);
xl=zeros(1,n);
zl=zeros(1,n);
G=zeros(n,6);
F=zeros(n-1,6);
pos=zeros(3,n);

switch c
    case 1
        for t=0:dt:tf
            i=floor(t/dt+1);
            A=CIRC2(r,rx,rz,t);
            xl(i)=A(1);
            zl(i)=A(2);
        end
    case 2
        k=4;
        for t=0:dt:tf
            i=floor(t/dt+1);
            A=ROD(r,rx,rz,k,t);
            xl(i)=A(1);
            zl(i)=A(2);
        end
    case 3
        for t=0:dt:tf
            i=floor(t/dt+1);
            A=SPI_ARC(r,rx,rz,t);
            xl(i)=A(1);
            zl(i)=A(2);
        end

     case 4
        for t=0:dt:tf
            i=floor(t/dt+1);
            A=SPI_LOG(r,rx,rz,t);
```

```
            xl(i)=A(1);
            zl(i)=A(2);
        end
    case 5
        A=SQUARE(0.3,0.7,0,0.4,dt,tf);
        xl=(A(1,:)');
        zl=(A(2,:)');
end

figure()
plot(xl,zl,'r:.');
axis equal
xlabel('x');
ylabel('z');

%*********************************

disp('Robotics, Vision & Control: (c) Peter Corke 1992-2011 http://
    www.petercorke.com')

if verLessThan('matlab', '7.0')
    warning('You are running a very old (and unsupported) version of
        MATLAB.  You will very likely encounter significant problems
        using the toolboxes but you are on your own with this');
end
tb = false;
rvcpath = fileparts( mfilename('fullpath') );

robotpath = fullfile(rvcpath, 'robot');
if exist(robotpath,'dir')
    addpath(robotpath);
    tb = true;
    startup_rtb
end

visionpath = fullfile(rvcpath, 'vision');
if exist(visionpath,'dir')
    addpath(visionpath);
    tb = true;
    startup_mvtb
end

if tb
    addpath(fullfile(rvcpath, 'common'));
    addpath(fullfile(rvcpath, 'simulink'));
end

clear tb rvcpath robotpath visionpath

%*********************************
```

```
links(1) = Link([    0       0       0      -pi/2   0]);
links(2) = Link([    0       0.105   0.400     0    0]);
links(3) = Link([    0       0       0      -pi/2   0]);
links(4) = Link([    0       0.320   0       pi/2   0]);
links(5) = Link([    0       0       0      -pi/2   0]);
links(6) = Link([    0       0.160   0       0      0]);

manus1 = SerialLink(links, 'name', 'manus1');
manus2 = SerialLink(links, 'name', 'manus2');
pos_i=[xl(1),0,zl(1)];
T=transl(pos_i);
qi=manus1.ikine(T);
G(1,:)=qi;

figure()
manus1.plot(qi);
axis([0 3 -0.5 1 -0.5 1]);
title('Posa Iniziale');
xlabel('X(m)');
ylabel('Y(m)');
zlabel('Z(m)');
q=qi;
pos(:,1)=T(1:3,4);


for i=1:1:n-1
    vl=[xl(i+1)-xl(i);0;zl(i+1)-zl(i)]/dt;
    v=vl+K*e;
    j1=manus1.jacob0(q);

    j1=j1(1:3,1:3);
    q1=j1\v;
    F(i,1:3)=q1;

    q(1)=q(1)+q1(1)*dt;
    q(2)=q(2)+q1(2)*dt;
    q(3)=q(3)+q1(3)*dt;
    G(i+1,:)=q;

    T=manus1.fkine(q);

    pos(:,i+1)=T(1:3,4);
    e=[xl(i+1);0;zl(i+1)]-pos(:,i+1);

end

qf=G(n,:);
c=180/pi;
G=c*G;
```

```
F=c*F;
figure()
manus2.plot(qf);
axis([0 3 -0.5 1 -0.5 1]);
title('Posa Finale');
xlabel('X(m)');
ylabel('Y(m)');
zlabel('Z(m)');

figure()
plot(pos(1,:),pos(3,:),'b');
axis equal;
xlabel('x');
ylabel('z');
legend('Traiettoria calcolata','Traiettoria desiderata','location','
    east');
title('PIANIFICAZIONE TRAIETTORIA');
hold on
plot(xl,zl,'r:.');
axis equal
legend('Traiettoria calcolata','Traiettoria desiderata','location','
    east');

t=linspace(0,tf,n);
t1=linspace(0,tf-dt,n-1);
G1=timeseries(G,t);
F1=timeseries(F,t1);
vf=zeros(6,1);
[a0,a1,a2,a3]=SPLINE2(G1,F1,vf);
q0=G(1,:);

T=0:dT:tf;
N=length(T);
q=zeros(6,N);
dq=zeros(6,N);
ddq=zeros(6,N);
l=1;
flag=0;

for j=1:1:n-1
while(T(l)<t(j+1)&&T(l)>=t(j)&&flag==0)
    for k=1:1:3
        q(k,l)=a0(j,k)+a1(j,k)*(T(l)-t(j))+a2(j,k)*((T(l)-t(j))^2)+a3(j
            ,k)*((T(l)-t(j))^3);
        dq(k,l)=a1(j,k)+2*a2(j,k)*(T(l)-t(j))+3*a3(j,k)*((T(l)-t(j))^2)
            ;
        ddq(k,l)=2*a2(j,k)+6*a3(j,k)*(T(l)-t(j));
    end

    for k=4:1:6
```

```
        q(k,l)=G(1,k);
    end
l=l+1;
if(l==N)
    flag=1;
end
end
end

for i=l:1:N
q(:,i)=G(n,:)';
end

figure();
plot(T,q(1,:),'b');
hold on
plot(T,q(2,:),'r');
hold on
plot(T,q(3,:),'g');
hold on
plot(t,G(:,1),'*');
hold on
plot(t,G(:,2),'*');
hold on
plot(t,G(:,3),'*');
title('SPLINE CUBICA');
legend('giunto 1','giunto 2','giunto 3','Location','east');
figure();
plot(T,dq(1,:),'b');
hold on
plot(T,dq(2,:),'r');
hold on
plot(T,dq(3,:),'g');
hold on
plot(t1,F(:,1),'*');
hold on
plot(t1,F(:,2),'*');
hold on
plot(t1,F(:,3),'*');
title('Profilo di velocita'');
legend('giunto 1','giunto 2','giunto 3','Location','southeast');
figure();
plot(T,ddq(1,:),'b');
hold on
plot(T,ddq(2,:),'r');
hold on
plot(T,ddq(3,:),'g');
title('Profilo delle accelerazioni');
legend('giunto 1','giunto 2','giunto 3','Location','southeast');
```

```
Q1=timeseries(dq(1,:),T);
Q2=timeseries(dq(2,:),T);
Q3=timeseries(dq(3,:),T);
Q4=timeseries(dq(4,:),T);
Q5=timeseries(dq(5,:),T);
Q6=timeseries(dq(6,:),T);

end
```

A.9.2 Algoritmo per due manipolatori a distanza fissa

```
function [qA0,qB0,vqA,vqB]=ALGINV2(dt,tf)

close all
n=floor(tf/dt+1);
K=(1/dt)*eye(3,3);
e=zeros(3,1);
eVP=zeros(3,1);

r=0.2;
rx=0.5;
rz=0.2;
xl=zeros(1,n);
zl=zeros(1,n);
xr=zeros(1,n);
zr=zeros(1,n);
GA=zeros(n,6);
GB=zeros(n,6);
FA=zeros(n-1,6);
FB=zeros(n-1,6);
PL=zeros(3,n);
PR=zeros(3,n);

for t=0:dt:tf
    i=floor(t/dt+1);
    A=CIRC1(r,rx,rz,t);
    A1=CIRC1(-r,rx,rz,t);
    xl(i)=A(1);
    zl(i)=A(2);
    xr(i)=A1(1);
    zr(i)=A1(2);
end

figure()
plot(xl,zl,'r:.');
axis equal
xlabel('x');
ylabel('z');

%*********************************

disp('Robotics, Vision & Control: (c) Peter Corke 1992-2011 http://
    www.petercorke.com')

if verLessThan('matlab', '7.0')
    warning('You are running a very old (and unsupported) version of
        MATLAB.  You will very likely encounter significant problems
        using the toolboxes but you are on your own with this');
```

```
end
tb = false;
rvcpath = fileparts( mfilename('fullpath') );

robotpath = fullfile(rvcpath, 'robot');
if exist(robotpath,'dir')
    addpath(robotpath);
    tb = true;
    startup_rtb
end

visionpath = fullfile(rvcpath, 'vision');
if exist(visionpath,'dir')
    addpath(visionpath);
    tb = true;
    startup_mvtb
end

if tb
    addpath(fullfile(rvcpath, 'common'));
    addpath(fullfile(rvcpath, 'simulink'));
end

clear tb rvcpath robotpath visionpath

%*********************************

links(1) = Link([    0       0       0      -pi/2    0]);
links(2) = Link([    0       0.105   0.400     0     0]);
links(3) = Link([    0       0       0      -pi/2    0]);
links(4) = Link([    0       0.320   0       pi/2    0]);
links(5) = Link([    0       0       0      -pi/2    0]);
links(6) = Link([    0       0.160   0       0       0]);

manusl1=SerialLink(links, 'name','manusl1');
manusr1=SerialLink(links, 'name','manusr1','base', transl(1,0,0));
manusl=SerialLink(links, 'name','manusl');
manusr=SerialLink(links, 'name','manusr','base', transl(1,0,0));
PL(:,1)=[xl(1),0,zl(1)];
PR(:,1)=[xr(1),0,zr(1)];
TL=transl(PL(:,1));
TR=transl(PR(:,1));
qA0=manusl.ikine(TL);
qB0=manusr.ikine(TR);
GA(1,:)=qA0;
GB(1,:)=qB0;

figure()
manusl1.plot(qA0);
hold on
```

```
manusr1.plot(qB0);
axis([0 3 -0.5 1 -0.5 1]);
title('Posa Iniziale');
xlabel('X(m)');
ylabel('Y(m)');
zlabel('Z(m)');

VP(:,1)=[xr(1)-xl(1);0;zr(1)-zl(1)];
qA=qA0;
qB=qB0;

for i=1:1:n-1
    vl=[xl(i+1)-xl(i);0;zl(i+1)-zl(i)]/dt;
    v1=vl+K*e;
    j1=manusl.jacob0(qA);
    j2=manusr.jacob0(qB);

    j1=j1(1:3,1:3);
    j2=j2(1:3,1:3);
    q1=j1\v1;
    FA(i,1:3)=q1;
    VP(:,i+1)=[xr(i+1)-xl(i+1);0;zr(i+1)-zl(i+1)];
    DVP=(VP(:,i+1)-VP(:,i))/dt;
    v2=K*eVP+DVP+j1*q1;
    q2=j2\v2;
    FB(i,1:3)=q2;

    qA(1)=qA(1)+q1(1)*dt;
    qA(2)=qA(2)+q1(2)*dt;
    qA(3)=qA(3)+q1(3)*dt;
    GA(i+1,:)=qA;

    qB(1)=qB(1)+q2(1)*dt;
    qB(2)=qB(2)+q2(2)*dt;
    qB(3)=qB(3)+q2(3)*dt;
    GB(i+1,:)=qB;

    T1=manusl.fkine(qA);
    T2=manusr.fkine(qB);
    PL(:,i+1)=T1(1:3,4);
    PR(:,i+1)=T2(1:3,4);
    e=[xl(i+1);0;zl(i+1)]-PL(:,i+1);
    eVP=VP(:,i+1)-(PR(:,i+1)-PL(:,i+1));

end

qaf=GA(n,:);
qbf=GB(n,:);
c=180/pi;
GA=c*GA;
```

```
GB=c*GB;
FA=c*FA;
FB=c*FB;
qA0=GA(1,:);
qB0=GB(1,:);

figure()
manusl.plot(qaf);
hold on
manusr.plot(qbf);
axis([0 3 -0.5 1 -0.5 1]);
title('Posa Finale');
xlabel('X(m)');
ylabel('Y(m)');
zlabel('Z(m)');

figure()
plot(PL(1,:),PL(3,:),'b');
legend('Traiettoria calcolata','location','northeast');
hold on
plot(PR(1,:),PR(3,:),'b')
axis equal;
xlabel('x');
ylabel('z');
title('PIANIFICAZIONE TRAIETTORIA');
hold on
plot(xl,zl,'r:.');
hold on
plot(xr,zr,'g:.');
axis equal

qa1=GA(:,1);
qa2=GA(:,2);
qa3=GA(:,3);

qb1=GB(:,1);
qb2=GB(:,2);
qb3=GB(:,3);

t=0:dt:tf;
t1=0:dt:tf-dt;

figure()
subplot(3,1,1)
plot(t,qa1);
title('posizione angolare giunto 1L')
xlabel('t[s]');
ylabel('q1[gradi]');
subplot(3,1,2)
plot(t,qa2);
```

```
title('posizione angolare giunto 2L')
xlabel('t[s]');
ylabel('q2[gradi]');
subplot(3,1,3)
plot(t,qa3);
title('posizione angolare giunto 3L')
xlabel('t[s]');
ylabel('q3[gradi]');
figure()
subplot(3,1,1)
plot(t,qb1,'r');
title('posizione angolare giunto 1R')
xlabel('t[s]');
ylabel('q1[gradi]');
subplot(3,1,2)
plot(t,qb2,'r');
title('posizione angolare giunto 2R')
xlabel('t[s]');
ylabel('q2[gradi]');
subplot(3,1,3)
plot(t,qb3,'r');
title('posizione angolare giunto 3R')
xlabel('t[s]');
ylabel('q3[gradi]');

G1=timeseries(GA,t);
F1=timeseries(FA,t1);
G2=timeseries(GB,t);
F2=timeseries(FB,t1);
vf=zeros(6,1);
[a0,a1,a2,a3]=SPLINE2(G1,F1,vf);
[b0,b1,b2,b3]=SPLINE2(G2,F2,vf);
dT=0.06; %specifica MANUS
T=0:dT:tf;
N=length(T);
qA=zeros(6,N);
vqA=zeros(6,N);
ddqA=zeros(6,N);
qB=zeros(6,N);
vqB=zeros(6,N);
ddqB=zeros(6,N);
l=1;

for j=1:1:n-1
while(T(l)<t(j+1)&&T(l)>=t(j))
    for k=1:1:3
        qA(k,l)=a0(j,k)+a1(j,k)*(T(l)-t(j))+a2(j,k)*((T(l)-t(j))^2)+a3(
            j,k)*((T(l)-t(j))^3);
        vqA(k,l)=a1(j,k)+2*a2(j,k)*(T(l)-t(j))+3*a3(j,k)*((T(l)-t(j))
            ^2);
```

```
        ddqA(k,l)=2*a2(j,k)+6*a3(j,k)*(T(l)-t(j));

        qB(k,l)=b0(j,k)+b1(j,k)*(T(l)-t(j))+b2(j,k)*((T(l)-t(j))^2)+b3(
            j,k)*((T(l)-t(j))^3);
        vqB(k,l)=b1(j,k)+2*b2(j,k)*(T(l)-t(j))+3*b3(j,k)*((T(l)-t(j))
            ^2);
        ddqB(k,l)=2*b2(j,k)+6*b3(j,k)*(T(l)-t(j));
    end

    for k=4:1:6
        qA(k,l)=GA(1,k);
        qB(k,l)=GB(1,k);
    end
l=l+1;
end
end

for i=1:1:N
qA(:,i)=GA(n,:)';
qB(:,i)=GB(n,:)';
end

figure();
plot(T,qA(1,:),'b');
hold on
plot(T,qA(2,:),'r');
hold on
plot(T,qA(3,:),'g');
hold on
plot(t,GA(:,1),'*');
hold on
plot(t,GA(:,2),'*');
hold on
plot(t,GA(:,3),'*');
title('SPLINE CUBICA');
legend('giunto 1','giunto 2','giunto 3','Location','east');
figure();
plot(T,vqA(1,:),'b');
hold on
plot(T,vqA(2,:),'r');
hold on
plot(T,vqA(3,:),'g');
hold on
plot(t1,FA(:,1),'*');
hold on
plot(t1,FA(:,2),'*');
hold on
plot(t1,FA(:,3),'*');
title('Profilo di velocita');
legend('giunto 1','giunto 2','giunto 3','Location','southeast');
```

```
figure();
plot(T,ddqA(1,:),'b');
hold on
plot(T,ddqA(2,:),'r');
hold on
plot(T,ddqA(3,:),'g');
title('Profilo delle accelerazioni');
legend('giunto 1','giunto 2','giunto 3','Location','southeast');

figure();
plot(T,qB(1,:),'b');
hold on
plot(T,qB(2,:),'r');
hold on
plot(T,qB(3,:),'g');
hold on
plot(t,GB(:,1),'*');
hold on
plot(t,GB(:,2),'*');
hold on
plot(t,GB(:,3),'*');
title('SPLINE CUBICA');
legend('giunto 1','giunto 2','giunto 3','Location','east');
figure();
plot(T,vqB(1,:),'b');
hold on
plot(T,vqB(2,:),'r');
hold on
plot(T,vqB(3,:),'g');
hold on
plot(t1,FB(:,1),'*');
hold on
plot(t1,FB(:,2),'*');
hold on
plot(t1,FB(:,3),'*');
title('Profilo di velocita'');
legend('giunto 1','giunto 2','giunto 3','Location','southeast');
figure();
plot(T,ddqB(1,:),'b');
hold on
plot(T,ddqB(2,:),'r');
hold on
plot(T,ddqB(3,:),'g');
title('Profilo delle accelerazioni');
legend('giunto 1','giunto 2','giunto 3','Location','southeast');

vqA=timeseries(vqA',T);
vqB=timeseries(vqB',T);

end
```

Appendice B

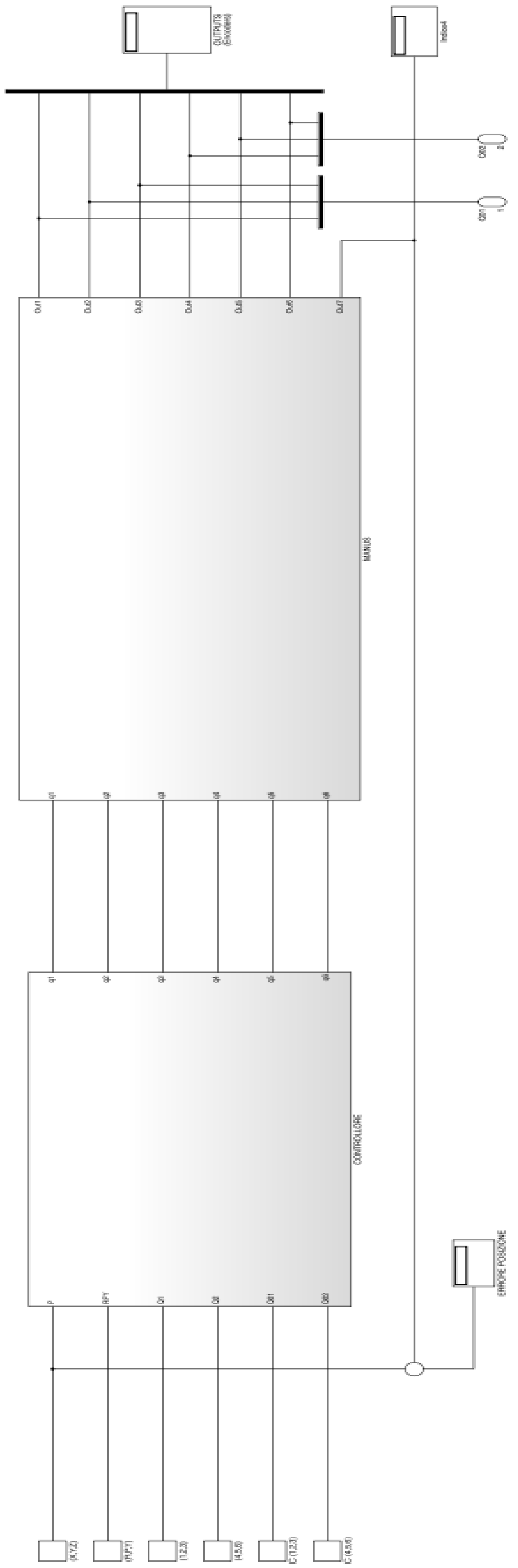

Figura B.1

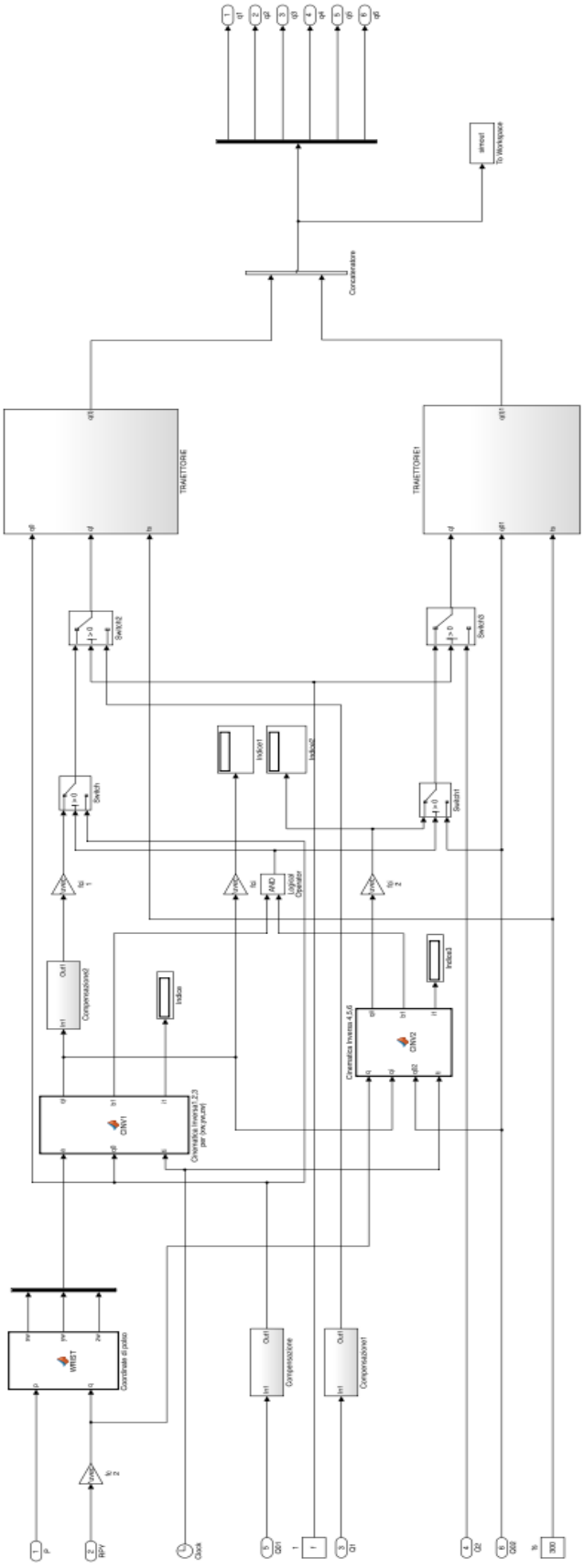
q1
q2
q3
q4
q5
q6
simout
To Workspace
Concatenate
TRAIETTORIE
TRAIETTORIE1
Switch2
Switch
Switch1
Switch3
Index1
Index2
Index3
AND
Logical Operator
Out1
Compensazione2
Index
Cinematica Inversa 4,5,6
CINV2
Cinematica Inversa(1,2,3) per xls (xls.cpp)
CINV1
WRIST
Coordinate di polso
Compensazione
Compensazione1
Data
Q01
Q1
Q2
Q02
300

Figura B.2

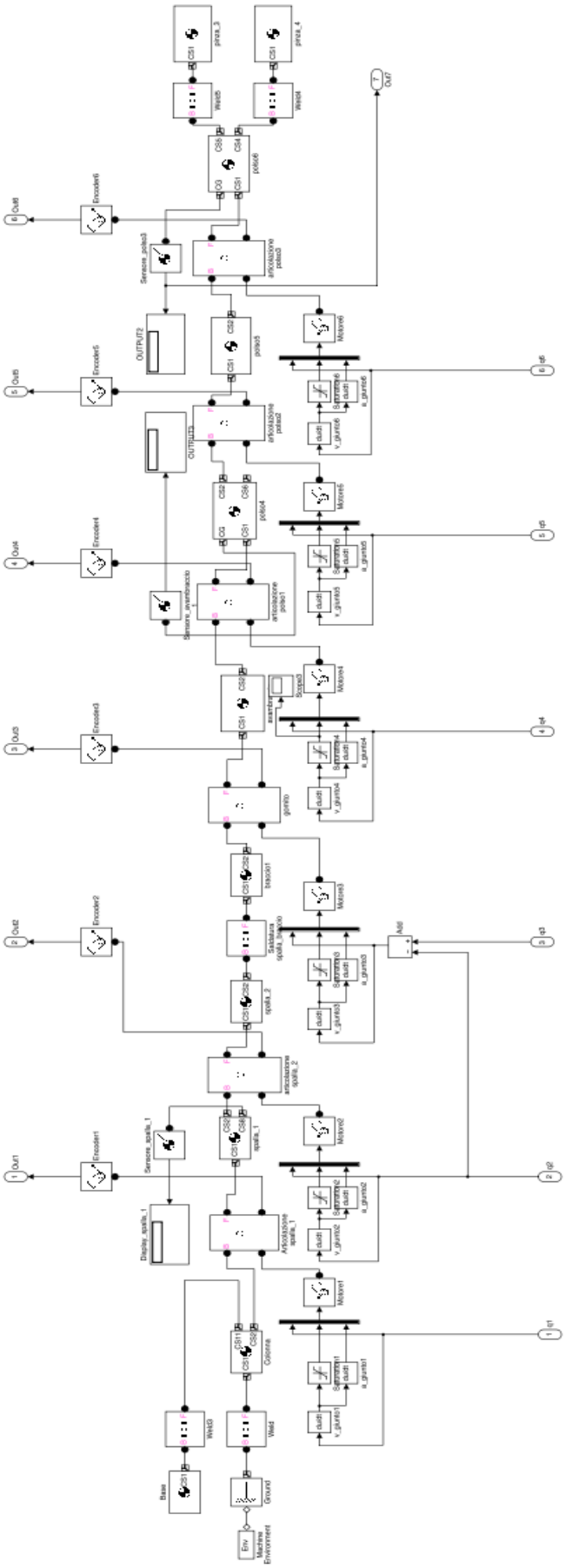

Figura B.3

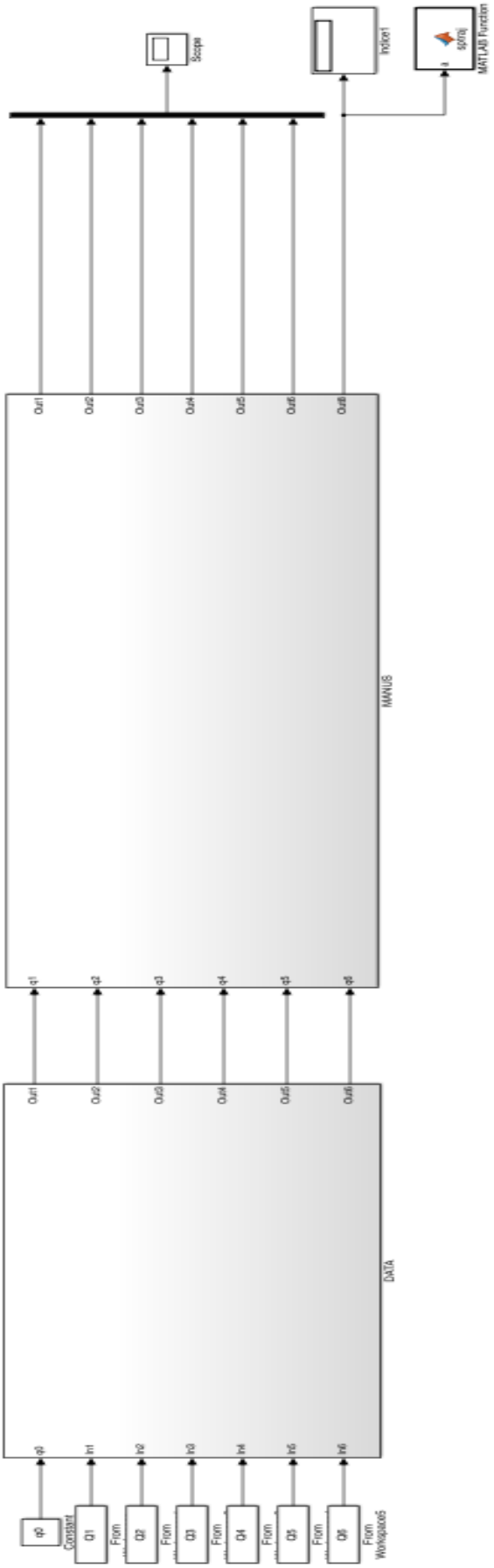
Scope
Index1
sprintj
MATLAB Function
Out1
Out2
Out3
Out4
Out5
Out6
MANUS
In1
In2
In3
In4
In5
In6
DATA
q0
Constant
Q1
Q2
Q3
Q4
Q5
Q6
From
From
From
From
From
From
Workspace

Figura B.4

Bibliografia

[1] Bonci T., Lonzi B., Mencarelli M., *Rappresentazione in realtá virtuale del manipolatore robotico MANUS*, Universitá Politecnica delle Marche, a.a. 2008-2009.

[2] Callegari M., Traú M., *Ricerca bibliografica sui metodi di calibrazione dei robot*, Corso di meccanica dei robot, Universitá Politecnica delle Marche, a.a. 2004-2005.

[3] Fabrizio M., *Elementi di meccanica classica*, Zanichelli, Bologna, 2003.

[4] Fanfoni F., *Presa di oggetti con un braccio robotico mediante controllo visivo*, Tesi di laurea specialistica in Ingegneria Informatica, Universitá degli studi di Parma, a.a. 2006-2007.

[5] Ferrari G., *Realizzazione di un'architettura software di governo per un robot manipolatore per compiti di assistenza*, Tesi di laurea in Ingegneria Informatica, Universitá degli studi di Parma, a.a. 2004-2005.

[6] Groothuis S. S., Stramigioli S., Carloni R., *Lending an helping hand*, in "IEEE Robotics and Automation Magazine", 2013.

[7] Jamisola R.S., Roberts R.G., *A more compact expression of relative Jacobian based on individual manipulator Jacobians*, in "Robotics and Autonomous Systems", n. 63, 2015.

[8] Longhi S., Iarlori S., Pomante A., Sabusco F., *Controllo ai giunti e controllo cartesiano cinematica diretta e inversa del robot manipolatore Manus*, Tesina del corso di robotica assistiva, Universitá Politecnica delle Marche, a.a. 2010-2011.

[9] McCaffrey E. J., *Kinematic analysis and evaluation of wheelchair mounted robotic arms*, Degree of Master of Science in Mechanical Engineering, University of South Florida, 2003.

[10] Nizzoli A., *Progettazione e realizzazione di un sistema di asservimento visivo per un robot manipolatore*, Tesi di laurea in Ingegneria Informatica, Universitá degli studi di Parma, a.a. 2005-2006.

[11] Occhi P., *Progettazione e realizzazione del sistema di controllo di un robot manipolatore per compiti di assistenza*, Tesi di laurea in Ingegneria Elettronica, Universitá degli studi di Parma, a.a 2003-2004.

[12] Sabbioni F., *Studio simulativo di leggi anti-windup per manipolatori robotici*, Tesi di laurea in Ingegneria Informatica, Universitá degli studi di Roma "Tor Vergata", a.a. 2003-2004.

[13] Siciliano B., Sciavicco L., Villani L., Oriolo G., *Robotica. Modellistica, pianificazione e controllo*, McGraw-Hill, III edizione, Napoli, 2008.

[14] Exact Dynamics: `http://www.exactdynamics.nl/site/`

[15] Riassunto cinematica MANUS: `http://www.exactdynamics.nl/site/`

[16] `petercorke.com/Robotics_Toolbox.html`

[17] `it.mathworks.com/`

[18] `https://it.wikipedia.org/`