UNIVERSITÀ DEGLI STUDI DI PADOVA

FACOLTÀ DI INGEGNERIA
CORSO DI LAUREA IN INGEGNERIA MECCATRONICA

TESI DI LAUREA MAGISTRALE

STRATEGIE INNOVATIVE PER LA PIANIFICAZIONE DEL MOTO DI MANIPOLATORI PARALLELI ADEPT

Relatore: Ch.mo Prof. GIOVANNI BOSCHETTI

Laureando: ALBERTO DUSO

Matricola 1019797-IMC

ANNO ACCADEMICO 2012-2013

Sommario

I manipolatori industriali attualmente presenti sul mercato offrono nella quasi totalità dei casi ambienti di programmazione proprietari nei quali vengono forniti comandi specifici per garantire la precisione dei posizionamenti e/o la loro velocità di esecuzione, trascurando o impedendo il più delle volte la possibilità da parte dell'operatore finale di deciderne in maniera puntuale i profili di movimentazione.

Sfruttando un particolare comando, messo a disposizione dell'azienda Adept e presente nell'ambiente di sviluppo fornito con i suoi manipolatori, si è cercato di ovviare a tale problema, sviluppando un software indipendente dal pianificatore proprio del controllore ed in grado di gestire con alta precisione i profili di posizione, velocità e accelerazione delle movimentazioni necessarie alle operazioni richieste.

Indice

Elenco delle tabelle

Elenco delle figure

Capitolo 1

Introduzione

La pianificazione del moto nei manipolatori industriali presenti tutt'oggi in commercio è, nella quasi totalità dei casi, un aspetto spesso lasciato inaccessibile all'utente finale, il quale può solo schedulare la sequenza di movimentazione per l'esecuzione delle lavorazioni necessarie al processo produttivo.

Il calcolo dei profili di moto utilizzati per lo spostamento dell'organo terminale tra due diverse posizioni nello spazio di lavoro è quindi affidato esclusivamente al pianificatore integrato nel controllore, il quale non può essere modificato o gestito dall'esterno.
Oltre il calcolo della legge di moto, il pianificatore presente nei manipolatori considerati va a definire tutti i limiti fisici e meccanici del sistema robot, i quali comprendono lo spazio di lavoro, le velocità massime e le accelerazioni massime supportate dal modello in uso.

Argomento primario della tesi è stato lo sviluppo di un software in ambiente di programmazione Adept V+ che, sfruttando particolari comandi presente in tale linguaggio, riesce a disabilitare completamente il pianificatore integrato nel controllore, lasciando all'operatore la facoltà di modificare la traiettoria della movimentazione e tutti i rispettivi limiti.

Successivamente tale software è stato integrato in modo da affiancare al controllore Adept un sistema real-time basato su ambiente Matlab Simulink in grado di occuparsi dell'effettivo calcolo in tempo reale della legge di moto desiderata per le operazioni in esecuzione dal robot, implementando inoltre l'elaborazione della cinematica diretta in maniera offline con lo scopo di sgravare ulteriormente il controllore Adept da tale onere.

Capitolo 2

I Manipolatori Paralleli

2.1 Definizione

In letteratura un manipolatore parallelo è definito come un meccanismo chiuso nel quale l'organo terminale è collegato alla base fissa (o telaio) mediante almeno due catene cinematiche indipendenti. Esso possiede n gradi di libertà e la sua movimentazione è data dalla presenza di n attuatori opportunamente montati.

I manipolatori paralleli sono generalmente costituiti da due basi, una fissa ed una mobile, collegate tra di loro mediante catene cinematiche. La base fissa è posta solitamente a telaio dove sono presenti gli attuatori, mentre la base mobile viene definita solidale all'organo terminale. Le catene cinematiche che collegano le due basi sono realizzate mediante aste rigide e diversi tipi di giunti, in maniera tale da consentire alla piattaforma mobile un certo numero di gradi di libertà.

2.2 Comparazione con i Manipolatori Seriali

I manipolatori seriali sono delle strutture a catena cinematica aperta costituite da aste rigide collegate in serie mediante accoppiamenti cinematici. Gli attuatori in essi presenti hanno la facoltà di agire indipendentemente gli uni dagli altri, al contrario dei manipolatori paralleli nei quali gli attuatori devono essere attivati in maniera collaborativa in modo da rispettare la chiusura della catena cinematica.

Quest'ultima tipologia implica una maggiore complessità dal punto di vista progettuale, ma permette il posizionamento di tutti gli attuatori a telaio, garantendo una struttura generale del robot molto più snella, la possibilità di usare servomotori di potenza maggiore ed una notevole rigidità complessiva.

2.2.1 Vantaggi dei Manipolatori Paralleli

In generale, i principali vantaggi dei manipolatori paralleli sono:

- grazie alla possibilità di posizionare tutti gli attuatori a telaio, la struttura complessiva del robot risulta molto più leggera, consentendo allo stesso tempo di utilizzare motori di taglia maggiore
- l'errore di posizionamento dell'organo terminale è ridotto, consentendo alte velocità, accuratezza di movimento, alte capacità di carico e rigidità notevole
- la maggior rigidità implica oscillazioni a frequenze naturali più elevate, riducendo gli errori di posizione dovuti a tali moti oscillatori non controllati
- caratteristiche di alte prestazioni dinamiche

Ulteriori vantaggi possono essere rilevati in fase di costruzione grazie all'eventuale simmetria strutturale:

- minori costi di produzione dovuti ad elementi meccanici uguali
- approccio costruttivo semplice e modulare

2.2.2 Svantaggi dei Manipolatori Paralleli

Assieme a molti aspetti positivi sopracitati, i manipolatori paralleli presentato alcuni lati negativi:

- lo spazio di lavoro disponibile che ne risulta non è una struttura geometrica spesso banale e appare piuttosto ridotto in rapporto alle dimensioni effettive occupate dalla struttura totale del robot
- nello spazio di lavoro si ha la presenza di punti di singolarità, che devono essere opportunamente evitati per non perdere il controllo del moto
- i rapporti di trasmissione relativi a forze e velocità, a causa della non linearità del posizionamento dell'organo terminale rispetto al moto degli attuatori, non risultano costanti in ogni punto dello spazio di lavoro; in esso si possono verificare significative variazioni delle massime velocità e forze ottenibili sull'organo terminale, variazioni che si manifestano in particolar modo in prossimità dei punti di singolarità
- la progettazione e la calibrazione dei manipolatori paralleli risultano più difficoltose a causa della maggior complessità strutturale della macchina

2.2.3 Configurazioni Singolari

Le configurazioni di singolarità che si possono manifestare nei manipolatori paralleli sono ottenibili all'interno dello spazio di lavoro o nelle immediate vicinanze dei suoi confini.

Si possono presentare due differenti tipologie di singolarità:

- singolarità seriali, ovvero configurazioni nelle quali sono richieste velocità infinite dei giunti per ottenere velocità dell'organo terminale finite
- singolarità parallele, ossia configurazioni nelle quali l'organo terminale non può sopportare alcun tipo di sforzo divenendo perciò incontrollabile

Le singolarità parallele sono da evitare in particolar modo in quanto all'avvicinarsi di tali configurazioni gli sforzi richiesti agli attuatori aumentano notevolmente, con possibili danni all'intera struttura e la contemporanea riduzione della rigidità complessiva, sfociando spesso in situazioni di perdita di controllo del moto.

2.3 Campi di Utilizzo

Grazie alle loro ottime prestazioni dinamiche e alla vasta flessibilità di lavoro, i manipolatori paralleli vengono impiegati in tutti quei campi dove sono richieste operazioni monotone e ripetitive, ma che necessitano di alte velocità di esecuzione e grande precisione.

I maggiori utilizzi si ottengono nelle fasi di packaging prevalenti in tutti i settori di produzione alimentare industriale.

La loro caratteristica di versatilità ne permette inoltre l'impiego in molti altri settori:

- pallettizzazione di oggetti aventi masse ridotte
- assemblaggio di componenti elettronici
- macchinari di misurazione
- realizzazione di interfacce aptiche
- operazioni di taglio basate su laser o water-jet

- composizione dei blister per le industrie farmaceutiche
- produzioni personalizzate di mosaici in ceramica

Le possibili applicazioni di tali manipolatori sono tutt'ora un punto cruciale durante la progettazione di nuovi prototipi.

Capitolo 3

Il Sistema in Analisi

3.1 Configurazione

L'intero lavoro di tesi svolto si basa sullo sviluppo del software di controllo per il manipolatore parallelo Adept Quattro ed è quindi facile intuire come la parte fondamentale di tutto il sistema in analisi sia il manipolatore stesso. Il setup sperimentale (Fig. 3.1) si completa poi con il controllore real-time dedicato Adept SmartController CX, il quale ha la funzione di controllare e gestire gli attuatori fissi a telaio mediante il software realizzato tramite Adept Desktop installato su PC sul quale si trova inoltre il software Matlab assieme al tool *The MathWorks Simulink*, ed in fine un PC industriale PXI National Instruments sul quale è installato un sistemo operativo real-time xPC Target . Il collegamento tra il manipolatore ed in controller è realizzato tramite un cavo FireWire che garantisce elevate velocità di trasmissione dei dati sfruttando il protocollo IEEE 1394, mentre la connessione tra lo SmartController ed i 2 PC presenti è gestita attraverso una rete Ethernet 10Mbps Full-Duplex.

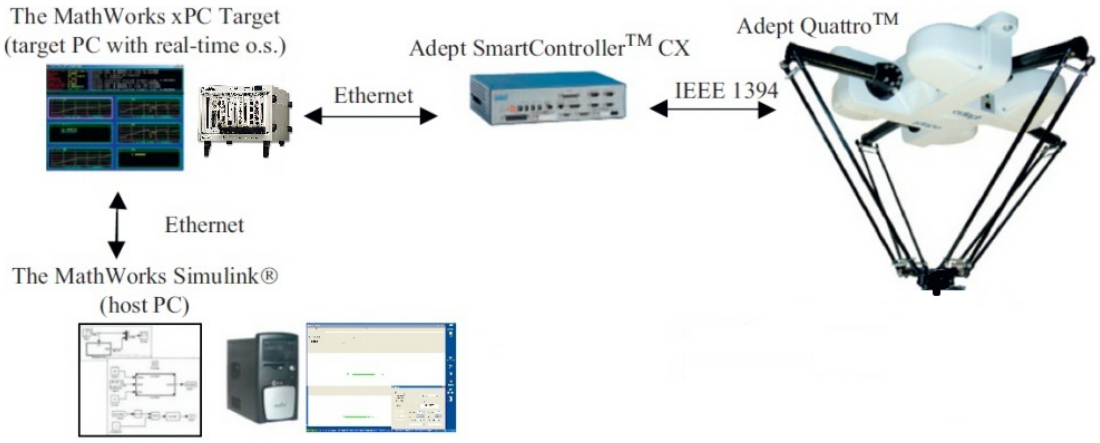

Fig. 3.1: Schema del Setup Sperimentale

3.2 Manipolatore Parallelo Adept Quattro

3.2.1 Struttura Generale

Il robot Adept Quattro s650H è un manipolatore parallelo a quattro assi che, mediante il controllo dei quattro attuatori montati a telaio, consente la traslazione nello spazio dell'organo terminale e la sua rotazione attorno all'asse verticale. Si può notare la struttura nella sua interezza nella figura 3.2.

Il manipolatore in esame è costituito principalmente da tre componenti:

- un telaio
- quattro catene cinematiche

Fig. 3.2: Manipolatore Parallelo Adept Quattro s650H

Fig. 3.3: Accoppiamento Attuatore-Biella

- una piattaforma mobile

Il telaio è una struttura fissa in alluminio nella quale sono alloggiati i quattro motori che movimentano le rispettive catene cinematiche. Ogni catena cinematica è costituita da una manovella e da una coppia di bilancieri. La manovella è collegata all'attuatore per mezzo di una coppia rotoidale che è visibile in figura 3.3.

A valle di ciascuna biella si collega una coppia di bilancieri per mezzo di due giunti sferici apprezzabili nella figura 3.4.

A valle delle quattro catene cinematiche si collega una piattaforma mobile. Ciascuna coppia di bilancieri viene vincolata alla piattaforma attraverso un giunto universale. Esistono diversi tipi di piattaforma mobile che possono essere utilizzati in questo tipo di manipolatore. Lo schema di piattaforma modellata in questa tesi è raffigurata nella figura 3.5. All'interno della piattaforma mobile assume un ruolo primario il quadrilatero identificato in figura 3.5 con le lettere F_i. La deformazione di questo quadrilatero porta alla rotazione della ruota di raggio maggiore che è collegata alla ruota di raggio minore con un

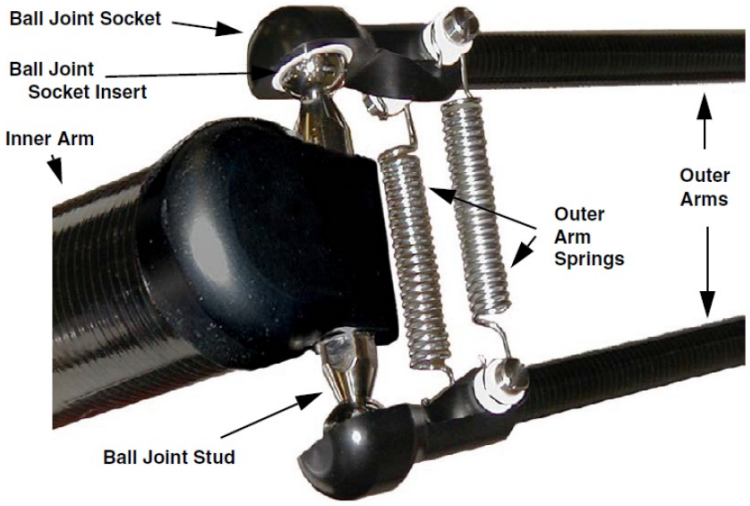

Fig. 3.4: Accoppiamento Biella-Bilancieri

rapporto di trasmissione 1 a 4. Al di sotto della ruota di raggio minore è possibile vincolare l'organo terminale più adatto a seconda del tipo di lavorazione che si vuole effettuare con il manipolatore.

3.2.2 Modello Geometrico

In questo paragrafo vengono definiti i parametri geometrici del manipolatore utilizzati in seguito per il calcolo offline della cinematica diretta di posizione. Nella figura 3.6 si propone la schematizzazione della struttura fissa a telaio vista dall'alto.

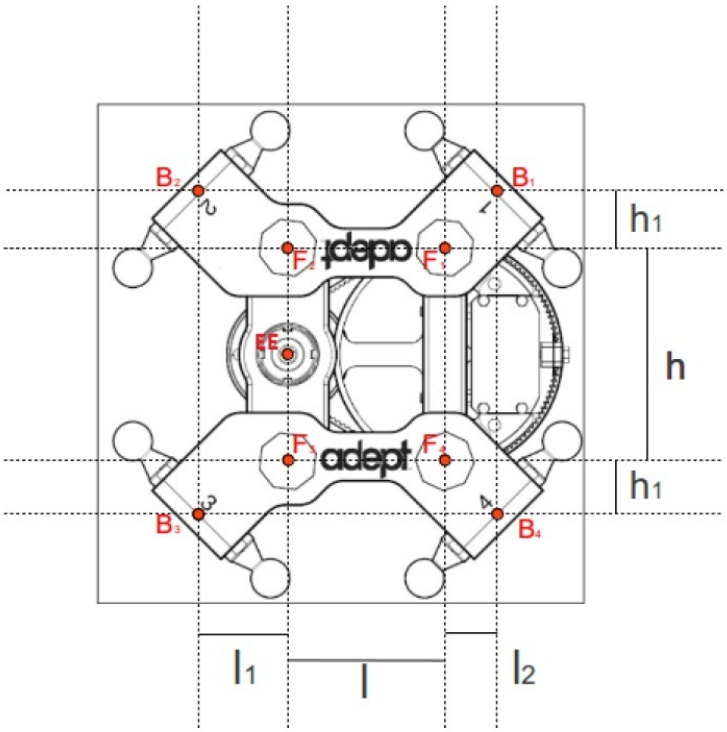

Fig. 3.5: Vista dall'Alto della Piattaforma Mobile

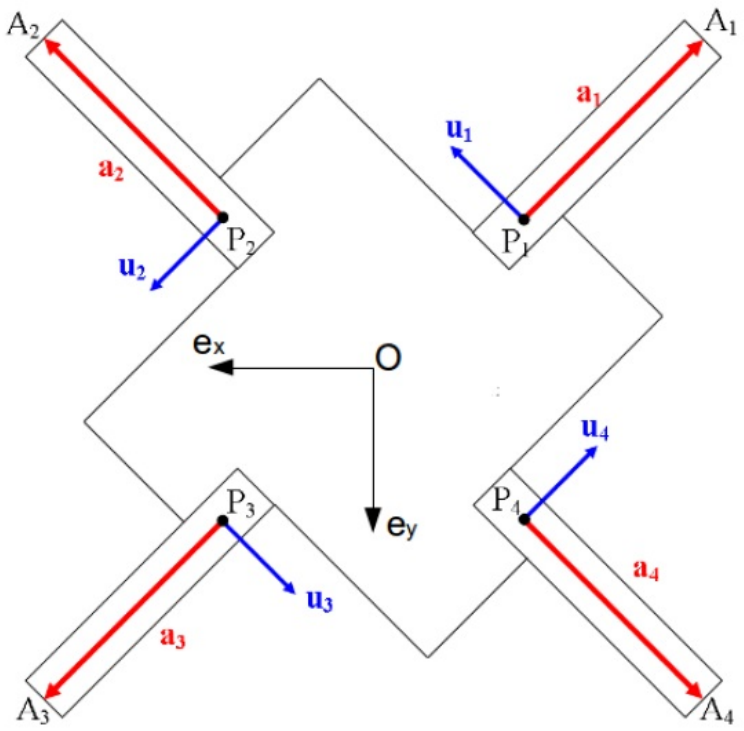

Fig. 3.6: Vista dall'Alto del Telaio Schematizzato

Il sistema di riferimento assoluto del manipolatore è posto al centro del telaio. Essendo il sistema di tipo destrorso, l'asse z, non visibile nella figura 3.6, sarà in direzione ortogonale rispetto al piano xOy e di verso uscente rispetto alla figura. Siano i punti P_i, con $i = 1, 2, 3, 4$, i punti di intersezione tra l'asse di rotazione della coppia rotoidale che vincola la biella all'attuatore e l'asse longitudinale della biella stessa. Da ciascun punto P_i avrà origine l'i-esima catena cinematica i cui elementi saranno anch'essi denominati con l'indice i. Nella figura 3.6 è possibile apprezzare la disposizione simmetrica dei punti P_i. Essi sono equidistanti dall'origine del sistema di riferimento assoluto e le loro proiezioni sul piano xOy giacciono sulle bisettrici dei quattro quadranti del medesimo piano cartesiano. La lunghezza del segmento generico $\overline{OP_i}$ e l'offset in z dei punti P_i rispetto al sistema di riferimento assoluto, che indichiamo con *zoff1*, sono parametri noti forniti dal produttore(tabella 3.1).
I vettori $\mathbf{P_i}$, che descrivono la posizione dei punti P_i, possono quindi essere definiti come di seguito:

$$\mathbf{P_1} = \begin{Bmatrix} x_{P_1} \\ y_{P_1} \\ z_{P_1} \end{Bmatrix} = \begin{Bmatrix} -\overline{OP_1} \cdot \cos\frac{\pi}{4} \\ -\overline{OP_1} \cdot \sin\frac{\pi}{4} \\ -z_{off_1} \end{Bmatrix} \tag{3.2.1}$$

$$\mathbf{P_2} = \begin{Bmatrix} x_{P_2} \\ y_{P_2} \\ z_{P_2} \end{Bmatrix} = \begin{Bmatrix} \overline{OP_2} \cdot \cos\frac{\pi}{4} \\ -\overline{OP_2} \cdot \sin\frac{\pi}{4} \\ -z_{off_1} \end{Bmatrix} \tag{3.2.2}$$

$$\mathbf{P_3} = \begin{Bmatrix} x_{P_3} \\ y_{P_3} \\ z_{P_3} \end{Bmatrix} = \begin{Bmatrix} \overline{OP_3} \cdot \cos\frac{\pi}{4} \\ \overline{OP_3} \cdot \sin\frac{\pi}{4} \\ -z_{off_1} \end{Bmatrix} \tag{3.2.3}$$

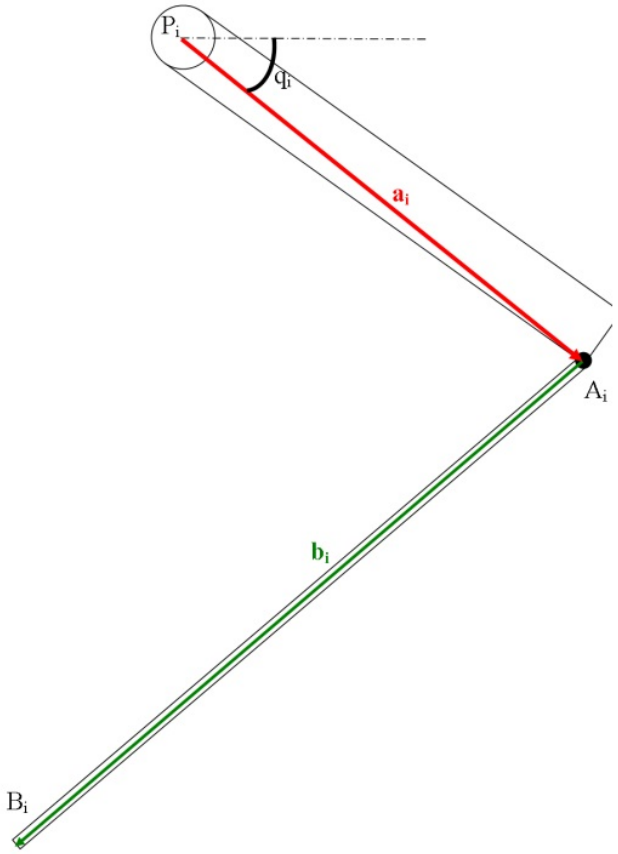

Fig. 3.7: Rappresentazione frontale di una Catena Cinematica Generica

$$\mathbf{P_4} = \begin{Bmatrix} x_{P_4} \\ y_{P_4} \\ z_{P_4} \end{Bmatrix} = \begin{Bmatrix} -\overline{OP_4} \cdot \cos\frac{\pi}{4} \\ \overline{OP_4} \cdot \sin\frac{\pi}{4} \\ -z_{off_1} \end{Bmatrix} \tag{3.2.4}$$

E quindi ora possibile definire il punto generico A_i come il punto di intersezione tra l'asse del bilanciere generico e la retta passante per il centro dei giunti sferici visibili in figura 3.4. La distanza tra i punti P_i e A_i, rappresentato dallo scalare a_i, è un parametro noto fornito dal produttore (tabella 3.1). L'inclinazione della biella rispetto al piano xOy viene indicata con gli angoli q_i. Tali angoli rappresentano le coordinate libere nell?analisi cinematica diretta di posizione che verrà trattata in seguito. In riferimento alla figura 3.7, che rappresenta una catena cinematica generica del manipolatore in esame, la variabile di giunto q_i assume valori positivi per rotazioni della biella verso il basso e valori negativi per rotazioni in verso opposto.

I vettori $\mathbf{A_i}$, che descrivono la posizione dei punti A_i, vengono definiti a partire dai punti fissi a telaio P_i definiti in precedenza.

$$\mathbf{A_i} = \begin{Bmatrix} x_{P_i} + sign(x_{P_i})\frac{\sqrt{2}}{2} a_i \cos q_i \\ y_{P_i} + sign(y_{P_i})\frac{\sqrt{2}}{2} a_i \cos q_i \\ z_{P_i} - a_i \sin q_i \end{Bmatrix} \tag{3.2.5}$$

Si possono osservare i vettori biella $\mathbf{a_1}$, $\mathbf{a_2}$, $\mathbf{a_3}$ e $\mathbf{a_4}$ nella figura 3.6. Il corrispondente vettore generico $\mathbf{a_i}$ è illustrato, in colore rosso, nella figura 3.7. Si applichi quindi una semplificazione alle quattro catene cinematiche: ciascuna coppia di bilancieri presente a

valle della manovella generica sia sostituita da un unico link, di medesima lunghezza e orientazione, posto esattamente a metà tra i due bilancieri originali.

Si definiscono ora i punti B_i come i punti di intersezione tra l'asse del bilanciere generico e la retta passante per i centri delle due sfere presenti ai quattro vertici della piattaforma mobile. Si può osservare la collocazione dei punti B_i anche in figura 3.5.

La posizione dei punti F_i è già stata descritta nel paragrafo precedente con l'ausilio della figura 3.5. I segmenti $\overline{F_2F_1}$ e $\overline{F_3F_4}$ rimangono sempre paralleli all'asse x mentre i segmenti $\overline{F_3F_2}$ e $\overline{F_4F_1}$ possono cambiare orientazione rispetto al sistema di riferimento assoluto.

E' proprio la rotazione di quest'ultima coppia di segmenti che mette in rotazione la ruota di raggio maggiore che, a sua volta, mette in rotazione, con amplificazione di fattore quattro, la ruota di raggio minore e l'organo terminale ad essa solidale.

Sempre nella figura 3.5, indicato con la notazione *EE*, è possibile osservare la posizione dell'organo terminale rispetto alle coordinate x e y.

Per concludere si riportano in tabella 3.1 tutti i parametri geometrici fin qui definiti con le misure dichiarate dal produttore ed in figura 3.8 il dettaglio dello spazio di lavoro di Adept Quattro s650H.

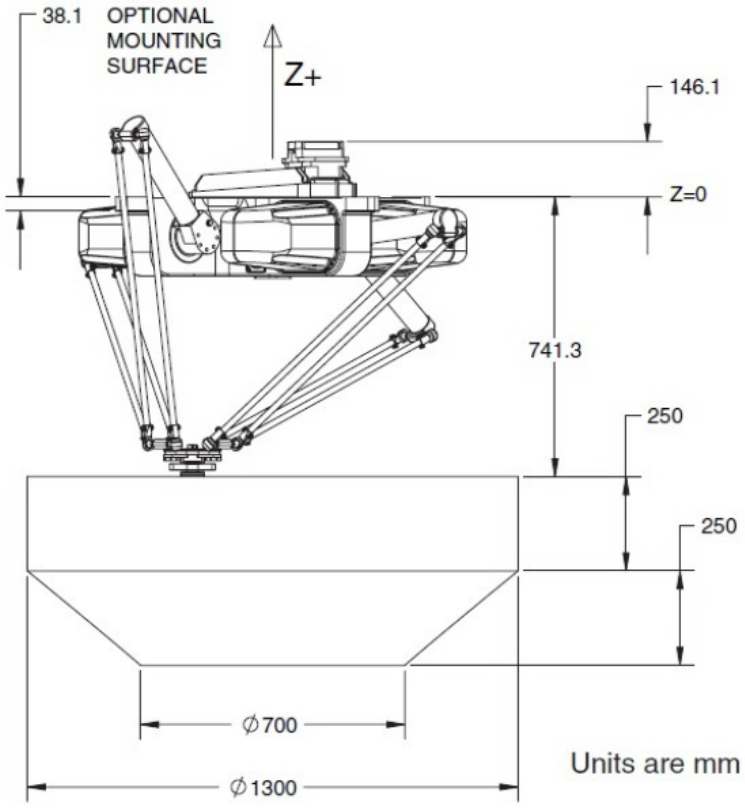

Fig. 3.8: Dettaglio dello Spazio di Lavoro di Adept Quattro s650H

3.3 Adept SmartController

Il sistema Adept SmartController (figura 3.9) è un unità di governo ad alte presta- zioni per robot e sistemi di visione, si basa sull'architettura SmartServo che consente un risparmio significativo della CPU. L'Adept SmartController dispone di un processore veloce ed è già predisposto per essere abbinato a sistemi di visione e porte encoder. Può comandare

Tab. 3.1: Parametri geometrici di Adept Quattro s650H

Parametro	Misura
$\overline{OP_i}$	0.275000 m
a_i	0.375000 m
b_i	0.825000 m
z_{off_1}	0.104775 m
z_{off_2}	0.086511 m
h	0.110000 m
h_1	0.025000 m
l	0.080750 m
l_1	0.044250 m
l_2	0.025000 m

fino a 24 assi esterni permettendo applicazioni multi assi e multi robot. L'interfaccia di comunicazione SmartServo e basata sulla tecnologia IEE 1394 (chiamata anche FireWire) quindi si abbina in modo perfetto al nostro set-up.

Fig. 3.9: Adept SmartController CX

3.4 PXI National Instruments

PXI è una piattaforma robusta basata su PC industriali per sistemi di automazione e misura. PXI combina le funzioni di bus elettrico PCI con il pacchetto modulare e robusto Eurocard di CompactPCI e aggiunge bus di sincronizzazione specializzati e funzioni software. Essa è una piattaforma a prestazioni elevate e a costi ridotti ideale per applicazioni come i test di produzione, applicazioni militari e aerospaziali, monitoraggio delle condizioni delle macchine, test industriale. e applicazioni nel settore automotive.
Sul PC industriale PXI utilizzato è stato installato un sistema operativo real-time xPC Target il quale esegue il software realizzato tramite l'ambiente di lavoro *The Mathworks Matlab Simulink*, interfacciandosi al controllore Adept SmartController per la gestione e la lettura in tempo reale del manipolatore parallelo Adept Quattro.

Fig. 3.10: PC industriale PXI National Instruments

Capitolo 4

Ambiente di sviluppo *Adept Desktop*

4.1 Introduzione

Il manipolatore parallelo Adept Quattro è fornito, come tutti gli altri manipolatori di casa Adept, di un ambiente di sviluppo proprietario necessario alla programmazione del controllore associato al robot. Tale software prende il nome di Adept Desktop e consente di avere a disposizione un'interfaccia completa e relativamente semplice per l'elaborazione dei programmi che si andranno a caricare sul controllore del manipolatore atti all'esecuzione delle più svariate operazioni in ambito industriale.

Come tutti gli ambienti di programmazione proprietari Adept Desktop mette a disposizione un'ampia libreria di comandi specifici per i propri manipolatori, utili ad eseguire ogni genere di movimentazione desiderata tramite l'integrazione con strutture di programmazione classiche come cicli, condizioni di verifica e timer.
Tali comandi vengono perciò implementati all'interno di Adept Desktop in un vero e proprio linguaggio di programmazione realizzato specificatamente per i manipolatori della casa costruttrice che prende il nome di V+, chiara analogia al linguaggio classico C++.

Tra i comandi più utilizzati durante l'implementazione di codice V+ si trovano quelli specifici per la movimentazione, quelli indirizzati alla lettura della posizione dell'organo terminale o degli encoder degli attuatori, quelli che gestiscono l'analisi dei registri interni sullo stato del manipolatore e molti altri.
Nello studio di tesi realizzato ci si è concentrati in particolar modo sull'uso di alcuni comandi speciali messi a disposizione da parte di Adept V+ che consentono all'operatore di disabilitare momentaneamente il pianificatore interno del controllore, lasciando al programmatore la completa libertà di controllo della movimentazione e dei limiti dello spazio di lavoro, seppur tale opzione non sia stata di facile utilizzo sia per la scarsa documentazione a riguardo fornita dalla casa costruttrice sia per le modalità di utilizzo dei comandi operanti in una parte così delicata del codice.

4.2 Il Pianificatore Interno

La scrittura di codice V+ per il controllo dei manipolatori Adept è basata su semplici comandi pre-implementati dalla casa costruttrice che consentono operazioni di movimentazione molto semplici.

La funzione **MOVE(S)** è quella principalmente utilizzata in tutte le fasi di spostamento, e consente di definire quale tipo di rototraslazione il manipolatore deve far eseguire all'organo terminale per arrivare in una locazione desiderata partendo dalla posizione dove attualmente esso si trova. Essa mi farà eseguire la movimentazione all'interno dello spazio di lavoro in modo da avere il minor tempo di spostamento o eventualmente tale da rendere la traiettoria di moto necessariamente lungo una linea retta.

Tutto ciò rende il comando MOVE(S) più che sufficiente per i più svariati tipi di utilizzo in ambito industriale dei manipolatori Adept, principalmente finalizzati ad operazioni pick-and-place dove l'importanza maggiore è assegnata alla rapidità di esecuzione e alla precisione nel posizionamento dell'organo di presa.

Qualora, invece, l'utilizzo del manipolatore sia finalizzato ad operazioni differenti dalle lavorazioni industriali classiche, è possibile che i comandi standard messi a disposizione non risultino sufficienti o adatti per il risultato finale desiderato.

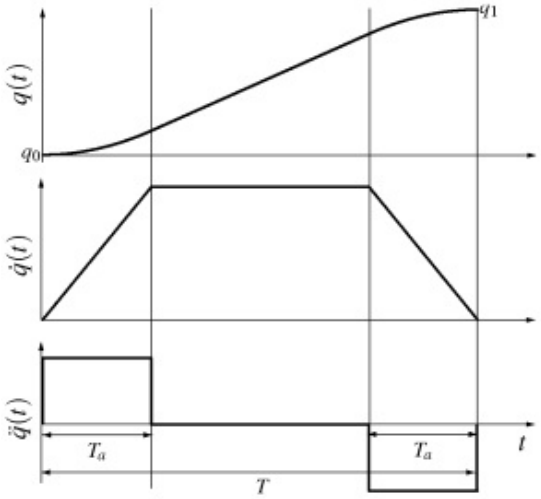

Fig. 4.1: Profilo standard del comando MOVE(S)

Ad esempio, in tutti quei casi dove la rapidità di esecuzione deve convivere con situazioni critiche dovute a oscillazioni o vibrazioni dei componenti sotto manipolazione, il profilo di movimentazione standard potrebbe risultare troppo brusco generando gli effetti indesiderati sopracitati(vedi ad es.trasporto di carichi oscillanti o fluidi).

Il *pianificatore interno* del controllore, infatti, esegue le movimentazioni specificate tramite il comando MOVE(S) generando dei profili di moto associati a leggi trapezoidali di velocità (Fig. 4.1), nei quali l'accelerazione risultante, costante a tratti, rende la dolcezza di movimento molto bassa, con tutte le ripercussioni negative del caso.

Si deve tener presente, a questo punto, che il pianificatore calcola il profilo di moto in base alla posizione iniziale, ovvero la posizione effettiva del tool all'istante di arrivo del comando MOVE(S), e alla locazione finale, specificata sempre nella sintassi di tale comando. Durante tutta la fase di movimentazione il pianificatore continua ad inviare in sequenza al manipolatore i punti intermedi della traiettoria calcolata ad una frequenza impostata all'interno del file di configurazione del controllore stesso, alla voce TRAJ_RATE. Se, ad esempio, TRAJ_RATE è impostato al valore di 250Hz, significa che il pianificatore riferisce al manipolatore i punti di passaggio intermedi ogni 4ms.
Altro valore da tenere bene a mente è il tempo ciclo proprio del controllore, ovvero la frequenza con cui esso esegue il codice V+, impostato al valore fisso di 16ms (62,5Hz).
Con tali impostazioni il risultato che ne conseguirebbe sono quattro riferimenti di posizione passati al manipolatore dal controllore per ogni ciclo macchina di quest'ultimo.

Si specificheranno in seguito i problemi causati da una non corretta impostazione di tali parametri e la strategia utilizzata per modificare il profilo del moto.

4.3 Il Ciclo Macchina del Controllore

Come sopracitato, il controllore esegue il software realizzato in codice V+ ad una frequenza fissa di 62,5Hz. Il ciclo macchina è impostato quindi a 16ms, durante i quali il controllore esegue tutti i task presenti nel software in esso caricato. Essendo progettato per un'implementazione multitasking del software, esso divide i 16ms in 16 time slice di 1ms ciascuno, durante i quali viene eseguito parte del task che ha ottenuto l'utilizzo dello slice di tempo in oggetto. In sintesi, il software realizzato può lavorare su più task, tutti concorrenti tra di loro per l'utilizzo di parte di tempo ciclo del controllore. L'ordine con cui i task vengono eseguiti dipende esclusivamente dal livello di priorità che è stato assegnato loro,

di modo che parti di software considerate di primaria importanza abbiano la sicurezza di essere eseguite nella loro interezza durante i 16ms. Qualora il numero di task sia elevato o la loro esecuzione totale necessiti di un tempo molto lungo, può capitare che alcuni di essi non riescano a venire completati all'interno del tempo ciclo sforando in cicli di esecuzione successivi. Questa è normale in sistemi multitasking e non viene considerata un problema in ambienti dove la cadenza temporale fissa non deve necessariamente venire rispettata, ovvero il completamento delle operazioni può richiedere un tempo variabile dipendentemente dal carico della cpu (vedi ad es. pc per uso domestico). Il sistema studiato in questo lavoro di tesi è al contrario rigorosamente di tipo time-based, ovvero tuttle le operazioni di movimentazione e lettura devono venire eseguite scrupolosamente entro il tempo ciclo del controllore di modo che la sincronizzazione tra riferimenti recepiti in tempo reale e la loro esecuzione nello spazio di lavoro sia esatta. Dopo numerose prove, realizzando in primo luogo il software su più task con priorità assegnate manualmente al fine di ottenere il compromesso ottimale, si è visto come tale soluzione non risulta la scelta corretta per il problema sotto analisi. Essa infatti non riesce a garantirmi la completa esecuzione di tutti i task nell'ordine corretto all'interno del tempo ciclo. Si è quindi optato per la realizzazione di un unico task all'interno del quale le operazioni da svolgere vengono implementate in maniera sequenziale nell'ordine desiderato, garantendo, con gli opportuni accorgimenti del caso, il totale completamento del task all'interno dei 16ms impostati.

4.4 Il Comando ALTER

Fine ultimo dell'esperienza svolta, riassumendo, è quello di personalizzare il profilo di moto per le movimentazioni desiderate, in modo da utilizzare leggi di maggior dolcezza utili a lavorazioni particolarmente delicate.

Tra i comandi non convenzionali messi a disposizione da Adept V+ troviamo la funzione **ALTER()** che, unita alle sintassi **ALTON()** e **ALTOFF()**, consente di disabilitare il pianificatore standard del controllore e di alterare, in tempo reale, le traiettorie delle movimentazioni schedulate mediante il comando classico MOVE(S).

Prima di tutto occorre quindi inserire nel programma in elaborazione i comandi di supporto ALTON() e ALTOFF() i quali fan si che il codice V+ contenuto tra di essi sia eseguito con il pianificatore interno disabilitato. Si ricorda che il pianificatore definisce, oltre che la legge di moto, anche i limiti dello spazio di lavoro del manipolatore, per cui si dovrà prestare molta attenzione a non eccedervi al fine di evitare possibili danneggiamenti meccanici del macchinario.
Nel software sviluppato è stato usata la sintassi "ALTON() 3", dove il parametro 3 imposta come sistema di riferimento per l'alterazione il sistema World, ossia quello fisso a telaio. Questo perché, se si andasse ad impostare come sistema di riferimento per il comando ALTER() quello solidale con l'organo terminale, eventuali rotazioni dell'organo di presa andrebbero a ruotare il sistema di riferimento per l'alterazione rispetto al sistema di riferimento World, modificando il comportamento desiderato dal comando ALTER().
In altre parole, se definissimo come sistema di riferimento per le alterazioni il tool e attraverso il comando ALTER() andassi a definire una specifica sequenza di alterazione della traiettoria lungo la direzione dell'asse X con una contemporanea rotazione dell'organo terminale, il risultato che ne derivante sarebbe una modifica non lungo la direzione X solidale a telaio ma rispetto ad un asse X solidale all'end-effector e quindi in rotazione con esso.

per quanto riguarda il comando ALTOFF(), invece, non ci sono ulteriori parametri da inserire visto che esso va solamente a terminare la parte di codice abilitata all'alterazione in tempo reale della traiettoria.

Passiamo ora a descrivere il vero e proprio comando chiave necessario alle operazioni di modifica volute, il comando ALTER().

La sintassi generale di tale funzione è:

ALTER() Dx,Dy,Dz,Rx,Ry,Rz

dove Dx,Dy,Dz coincidono rispettivamente con gli step di alterazione del percorso di moto nelle sue componenti X,Y,Z espressi in millimetri mentre, in maniera analoga, Rx,Ry,Rz definiscono gli step di alterazione delle traiettorie di rotazione attorno ai tre assi X,Y,Z espressi in gradi.

Il comando di alterazione ALTER(), quindi, deve essere associato in ogni caso ad un comando base di movimentazione definito da MOVE(S), che mi andrà a definire il percorso di movimentazione sul quale poter lavorare con le operazioni di alterazione.
In realtà, però, ciò che si è voluto realizzare in questo elaborato non è una modifica di un percorso già esistente, ma la definizione totale e nuova di un profilo di movimentazione prescelto. Tale operazione è stata possibile grazie ad un intelligente uso della coppia di comandi MOVE(S) e ALTER(), chiarita con un breve esempio.
Si supponga di aver definito in V+ una locazione HOME come posizione iniziale per l'organo terminale, e supponiamo esso già in tale posizione. Si lanci ora l'esecuzione di un programma contenente le seguenti righe di codice:

```
ALTON() 3
i=0
MOVES HOME
WHILE i < 10 DO
   ALTER() 1,0,0,0,0,0
    i=i+1
END
ALTOFF()
```

Dal momento che l'organo terminale si trova già nella locazione HOME, l'ulteriore comando MOVES non fa variare la sua posizione ma consente una sua alterazione tramite il comando ALTER().
Esso, infatti, andrà ad incrementare la posizione dell'end-effector, lungo l'asse X, di 1mm ad ogni ciclo macchina, ovvero ogni 16ms, finché la condizione è verificata (nell'esempio lo spostamento complessivo sarà di 10mm). La traiettoria risultante sarà per cui un segmento rettilineo di lunghezza 10mm lungo la direzione X composto da più spostamenti, anch'essi rettilinei, del valore di 1mm eseguiti in sequenza ad intervalli di 16ms.

Si fa quindi notare che il comando di movimentazione MOVE(S) viene utilizzato solamente come supporto alla successiva sequenza di alterazioni eseguite tramite ALTER(), che determineranno effettivamente il profilo di moto desiderato.

Come si può notare dall'esempio il comando ALTER() necessita non della posizione assoluta dei punti intermedi della traiettoria per i quali passare bensì degli step di spostamento da applicare ad ogni ciclo macchina alla locazione istantanea. Tali incrementi devono venire perciò calcolati in tempo reale in base alla posizione attuale dell'end-effector e alla posizione da raggiungere al ciclo successivo, ricavata dalla traiettoria di moto definita in funzione del tempo.
In realtà lo step viene elaborato esclusivamente come differenza dei punti teorici definiti dalla legge di moto di posizione agli istanti t e $t\text{-}1$, utilizzando la lettura della posizione effettiva, tramite gli encoder posti sui motori, solamente per una verifica grafica del percorso effettuato. In seguito verranno presi in esame anche alcuni esempi di alterazioni effettuate sfruttando il feedback di posizione, evidenziandone possibili vantaggi e difficoltà di realizzazione.

Il metodo di movimentazione del manipolatore appena descritto è dunque, una volta capita la sintassi dei comandi utilizzata, piuttosto semplice e può essere utilizzato per la

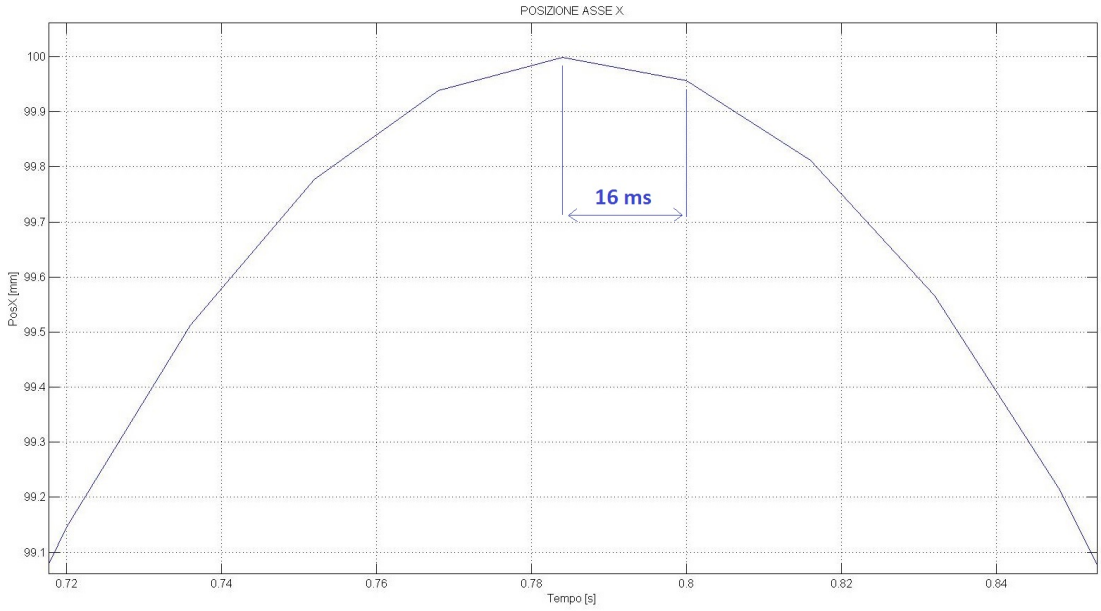

Fig. 4.2: Particolare di una traiettoria sinusoidale realizzata a tratti

realizzazione dei più svariati profili di moto.
Lo sviluppo dello spostamento tra due locazioni nello spazio di lavoro è, in sintesi, creato dalla concatenazione di brevi spostamenti rettilinei eseguiti nell'arco dei 16ms dati dal ciclo macchina del controllore. Ovvero una traiettoria curva viene generata da una traiettoria spezzata dove i segmenti che la compongono sono equispaziati nel tempo di 16ms, mentre la lunghezza dei segmenti effettuati dipende dalla tipologia del profilo di moto: dove la velocità è bassa gli step calcolati saranno più piccoli rispetto alle zone della traiettoria dove la velocità di movimentazione è maggiore (Fig. 4.2).
Tale comportamento scandito in maniera rigorosa dal ciclo macchina è necessario per tutti i sistemi di movimentazione basati su riferimenti che vengono generati in tempo reale, dove la perfetta sincronizzazione tra spostamento e tempo di esecuzione è essenziale al fine del problema da risolvere.
Si deve essere certi, in altre parole, che l'organo terminale del manipolatore si trovi in precise locazioni dello spazio di lavoro a ben precisi istanti di tempo predefiniti.
Un esempio, affrontato in alcuni lavori di ricerca, è lo smorzamento delle vibrazioni create dalla movimentazione di carichi oscillanti. La soppressione di tali vibrazioni si basa sulla variazione del ritardo con cui il riferimento di posizione calcolato viene trasmesso al manipolatore. In tale controllo risulta perciò essenziale una sincronizzazione il più possibile priva di ritardi di movimentazione tra il generatore del riferimento e l'esecuzione da parte del manipolatore, che può essere realizzata esclusivamente con sistemi time-based come il lavoro sotto analisi.

4.5 Il Problema del TRAJ_RATE

Al par. 4.2 si è accennato ad alcune impostazioni delicate presenti all'interno del controllore Adept. Particolare attenzione deve venir posta sulla voce TRAJ_RATE, la frequenza con cui il controllore riferisce al manipolatore i punti intermedi della traiettoria di movimentazione.
Se la traiettoria è calcolata autonomamente dal pianificatore interno durante l'uso del classico comando MOVE(S), maggiore è la frequenza di calcolo dei punti intermedi di

passaggio è maggiore sarà la continuità e fluidità dei movimenti dell'end-effector. Anche il pianificatore interno realizza alla fine moti curvilinei tramite concatenazione di più tratti rettilinei, percui maggiore è la densità di tali tratti e maggiore sarà la fedeltà della traiettoria realizzata rispetto a quella teorica.
Come già evidenziato la limitazione dei comandi di moto predefiniti è l'impossibilità di variare il tipo di profilo di moto da parte dell'operatore finale.

Ritornando ora all'utilizzo del comando ALTER() si è specificato come esso necessiti del calcolo continuo degli step di movimentazione da effettuare.Tali step sono calcolati all'interno del software V+ caricato sul controllore e perciò sono limitati in frequenza dalla cadenza imposta dal ciclo macchina, fissata a 16ms ovvero 62,5Hz.
In realtà all'interno dei 16ms di clock del controllore si riescono ad eseguire molti calcoli di step in quanto semplici differenze da eseguire. Il problema sta nell'esecuzione di tali step da parte del manipolatore, che non sempre garantisce più alterazioni all'interno del ciclo macchina. Si è preferito quindi limitarci alla frequenza di 62,5Hz propria del controllore onde evitare possibili perdite di sincronizzazione tra riferimento acquisito e la sua esecuzione effettiva.
A valle di tale scelta è perciò necessario impostare correttamente il valore di TRAJ_RATE anch'esso a 62,5Hz, per mantenere unitario il numero di movimentazioni eseguite ad ogni ciclo. Qualora si impostassero valori di TRAJ_RATE maggiori, ad esempio 125Hz, il manipolatore eseguirebbe due alterazioni per ogni ciclo macchina relative allo stesso step calcolato. Ne conseguirebbe una traiettoria reale identica a quella desiderata ma di ampiezze radoppiate.

Ricapitolando, ad ogni ciclo macchina di 16ms il controllore acquisisce in tempo reale il riferimento di posizione, calcola lo step necessario in base alla locazione attuale, esegue l'alterazione dell'organo terminale e legge la posizione finale raggiunta tramite gli encoder.

4.6 I Comandi DECOMPOSE, HERE e JHERE

La verifica della corretta movimentazione dell'organo terminale lungo la traiettoria pianificata è un aspetto essenziale al fine di valutare la bontà del metodo di controllo.
Come già reso noto al par. 4.4 gli step di alterazione sono calcolati dal software utilizzando il profilo di posizione teorico di riferimento, per cui tale controllo si può definire "a catena aperta" ovvero non ci sono feedback di posizione all'intero dell'anello di calcolo. Essi infatti mi introdurrebbero ulteriori ritardi , aspetto che si vuole cercare ridurre il più possibile. La lettura della posizione dell'end-effector viene quindi utilizzata come verifica offline per raffrontare la traiettoria teorica da quella effettivamente eseguita dal manipolatore.
A tale scopo si sono seguite due strade, delle quali si evidenzieranno pro e contro in base alle necessità di progetto.

La scelta più semplice e veloce che si può fare per la lettura della posizione del tool nello spazio di lavoro è l'utilizzo del comando **DECOMPOSE** associato alla funzione **HERE**. Quest'ultima restituisce la locazione della posizione dell'organo terminale al momento della sua esecuzione, racchiudendo la posizione assoluta e l'orientazione dello stesso. Per poter estrarre da tale locazione le componenti in X,Y,Z nello spazio di lavoro e contemporaneamente l'orientazione dell'end-effector si deve utilizzare la funzione DECOMPOSE, che andrà a scompormi la locazione in un vettore di sei celle, tre per la posizione e tre per le rotazioni attorno ai tre assi di riferimento. Esempio di sintassi:

DECOMPOSE vett_param[] = HERE

Come si può notare l'utilizzo di tale sistema ci consente di avere la lettura di posizione e orientazione in maniera estremamente semplice. Ovviamente la semplicità ha sempre un prezzo da pagare ed in questo caso esso si traduce in tempo di esecuzione. La funzione

HERE, infatti, si affida al controllore per il calcolo della cinematica diretta di posizione a partire dalla lettura degli encoder posizionati sugli attuatori. La procedura per tale calcolo è di tipo iterattivo e non in forma chiusa, per cui l'errore del risultato non è mai nullo e il tempo impiegato per completare la convergenza del metodo può essere eccessivo. Attraverso molteplici test si è realizzato che la lettura della locazione con questo sistema necessita di alcuni millisecondi che, considerato un tempo ciclo di 16ms, possono risultare troppi. Avendo infatti la necessità di compiere il calcolo dello step, la movimentazione e la lettura della posizione rigorosamente all'interno del ciclo macchina, un'eccessivo utilizzo degli slice time messi a disposizione durante il ciclo può compromettere la sincronizzazione delle alterazioni successive. Questa strada risultà quindi molto semplice dal punto di vista realizzativo, ma racchiude in sè diverse problematiche di tempo che vanno a scontrarsi con la tipologia di controllo che si sta studiando, rigorosamente time-based.

Per ovviare a tali problemi si è pensato di eseguire il calcolo della cinematica diretta attraverso una procedura offline, affidata all'embedded pc a cui il controllore è collegato ed attraverso il quale esso riceve i riferimenti di posizione. Esso è equipaggiato con un sistema operativo real-time capace di eseguire programmi realizzati attraverso Matlab-Simulink ed opportunamente compilati, dei quali si illustrerà lo sviluppo nei capitoli successivi. Il calcolo offline della cinematica diretta di posizione necessita in ogni caso della lettura dagli encoder posizionati sui motori, funzione che può essere affidata ancora una volta esclusivamente al controllore Adept. Il comando **JHERE**, analogo al precedente HERE, mi restituisce il valore dei quattro angoli di rotazione dei motori fissi a telaio rispetto al sistema di riferimento assoluto fisso anch'esso a telaio. Tali angoli, espressi in gradi, verranno successivamente passati al calcolatore real-time che li sfrutterà per la risoluzione delle equazioni di chiusura della cinematica di posizione. Tutto ciò risulta notevolmente più complesso della scelta precedentemente analizzata ma apporta un grosso vantaggio sull'utilizzo di parte del tempo ciclo, ridotto di alcune decine di volte. La funzione JHERE, infatti, impiega qualche decimo di millisecondo per la sua esecuzione, lasciando il resto del tempo ciclo libero di poter essere sfruttato per l'operazione fondamentale del manipolatore ovvero l'alterazione.

4.7 Descrizione Generale del Programma V+

Viene ora fatta una rapida descrizione del codice realizzato in V+ nella sua generalità, analizzando le varie fasi in ordine di esecuzione da parte del controllore. Per un riscontro visivo più immediato si fa riferimento alla figura 4.3 che rappresenta, tramite uno schema a blocchi, le varie parti del software realizzato, mentre per il codice in dettaglio si faccia riferimento all'appendice.

La prima operazione, eseguita dal controllore nella fase iniziale dell'esecuzione del software, è il posizionamento dell'organo terminale nella locazione di *Home* prestabilita. E' stata scelta come posizione di riposo il punto $(x_h, y_h, z_h) = (0, 0, -900)$. In tale configurazione, infatti, in manipolatore dispone della massima escursione di movimento sia sul piano $X - Y$, sia lungo l'asse Z.

Una volta posizionato il manipolatore in *Home*, il controllore apre una connessione Ethernet tramite protocollo UDP con il real-time xPC, mettendosi in attesa del pacchetto contenente le informazioni sul riferimento di posizione successivo per l'end-effector. Tale operazione avviene ad una frequenza di 62.5Hz come già descritto nei paragrafi precedenti.

Ricevuto il pacchetto del riferimento, esso viene scompattato e vengono calcolati gli step di movimentazione per le tre componenti X, Y, Z e, se desiderata, per la rotazione. Tali incrementi sono calcolati come differenza dal riferimento di posizione ricevuto al ciclo precedente ed il riferimento appena letto dall'xPC. Qualora gli step da far eseguire al manipolatore fossero troppo ampi rispetto alle sue capacità meccaniche, essi vengono

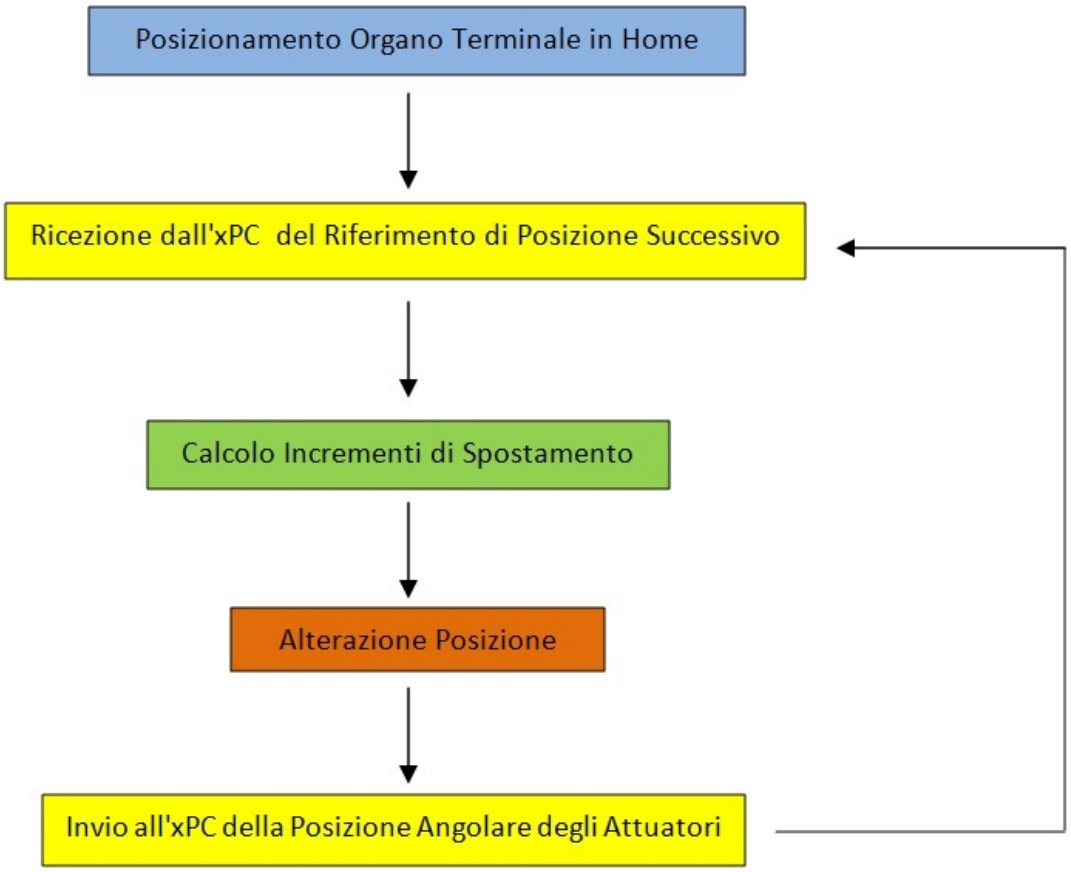

Fig. 4.3: Schema di Flusso del software V+ eseguito dal Controllore

limitati al massimo incremento consentito riserbandosi di far eseguire l'eccedenza nelle movimentazioni successive.

A valle di tutti i calcoli preventivi c'è poi la vera e propria alterazione, ovvero la movimentazione dell'organo terminale degli incrementi ottenuti. Essa viene eseguita attraverso il comando ALTER() analizzato in precedenza.

Una volta terminata la movimentazione vengono letti dagli encoder, posti sugli attuatori del manipolatore, le posizioni angolari di quest'ultimi tramite la funzione JHERE. Viene in seguito costruita ed incapsulata in un pacchetto UDP la stringa di dati da inviare all'xPC, il quale calcolerà in modo offline la cinematica diretta di posizione consentendo una verifica finale della traiettoria reale eseguita dal tool e quella teorica inviata come riferimento.

Qui il ciclo macchina del controllore termina, ripartendo dalla fase di ricezione dei riferimenti una volta scaduti i 16ms canonici.

Capitolo 5

Ambiente di sviluppo *Matlab Simulink*

5.1 Introduzione

Nel capitolo precedente è stato descritto l'ambiente di sviluppo proprietario per la gestione del controllore Adept SmartController. Esso esegue le funzioni basilari per la movimentazione del manipolatore e la lettura degli encoder dello stesso tramite il software realizzato in codice V+. Tale programma, in ogni caso, non può lavorare autonomamente per il tipo di controllo desiderato, ma deve necessariamente appoggiarsi ad un'unita esterna che analizza, crea ed invia in tempo reale i dati ed i riferimenti indispensabili per la movimentazione del robot. Come schematizzato in figura 3.1 tale compito è devoluto al PC industriale PXI National Instruments equipaggiato con s.o. real-time xPC Target, sul quale viene fatto eseguire il software per la generazione dei riferimenti di posizione ed il calcolo della cinematica diretta realizzato tramite l'ambiente di sviluppo Matlab Simulink.

Si andrà a descrivere di seguito tale programma in tutte le sue parti analizzate singolarmente.

5.2 Analisi Generale

Il software realizzato in ambiente Simulink è composto da più blocchi funzionali interconnessi tra di loro al fine di generare i riferimenti di posizione per i tre assi di lavoro più la rotazione dell'organo terminale del manipolatore Adept.

Si può osservare lo schema generale di interconnessione tra i vari blocchi in figura 5.1.

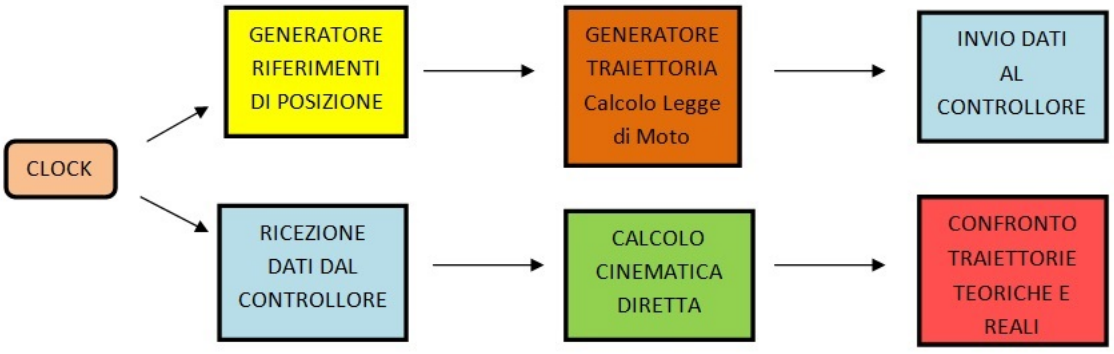

Fig. 5.1: Schematizzazione Software Simulink

Descrivendo rapidamente la sequenza di esecuzione troviamo:

- il *Clock*, il cui compito è quello di garantire la sincronizzazione tra l'unità real-time xPC Target e il controllore Adept SmartController
- il *Generatore dei Riferimenti* che definisce la successione di punti da raggiungere nello spazio di lavoro

- il *Generatore della Traiettoria*, ovvero il blocco principale che esegue il calcolo delle leggi di moto desiderate per i vari assi in base ai limiti di velocità ed accelerazione impostati

- il blocco di *Invio dei Dati* al controllore che si occupa dell'impacchettamento e la trasmissione via rete dei riferimenti calcolati

- il blocco di *Ricezione dei Dati* dal controllore che si occupa dello spacchettamento dei valori ricevuti dagli encoder del manipolatore

- il *Calcolo della Cinematica Diretta*, che consente di determinare offline la posizione dell'organo terminale in base alla posizione degli attuatori del manipolatore

- la parte di *Analisi e Confronto Grafico* delle traiettorie reali rispetto a quelle teoriche calcolate dai blocchi precedenti

Tutti i blocchi menzionati sono implementati in un unico modello Simulink in primo luogo, successivamente codificato ed implementato per l'esecuzione su xPC Target in tempo reale.

5.3 Generatore Riferimenti di Posizione

Con tale blocco si vuole cercare di simulare l'arrivo, in tempo reale, di successive locazioni nello spazio di lavoro che il manipolatore deve raggiungere nel minor tempo possibile in base ai limiti impostati nel blocco successivo.

La particolarità del software sviluppato, infatti, è quella di riuscire a calcolare online i comandi di movimentazione per il manipolatore in base alle richieste di posizionamento che arrivano in tempo reale, ovvero non si conoscono a priori le traiettorie di movimentazione che il robot dovrà eseguire.

Sono state definite perciò diverse locazioni, da raggiungere in sequenza dal manipolatore, tali da sfruttare l'intero spazio di lavoro del robot (figura 3.8) in modo da mettere in evidenza il comportamento reale del controllo anche in prossimità dei limiti fisici e meccanici propri di Adept Quattro s650H (figure 5.2, 5.3, 5.4).

Partendo dalla posizione iniziale $P_0 = (x_0, y_0, z_0) = (0, 0, 0)$ i punti scelti sono stati in ordine:

- $P_1 = (300, 300, 110)$

- $P_2 = (200, -100, 120)$

- $P_3 = (-100, 200, -50)$

- $P_4 = (-375, -490, 155)$

- $P_5 = (440, 500, -70)$

- $P_6 = (0, 0, 0)$

Si precisa che tutti i valori sono espressi in millimetri e che la posizione di riposo P_0=P_6 corrisponde in realtà alla locazione (0,0,-900), in modo da lasciare al manipolatore il massimo range di mobilità lungo l'asse Z.

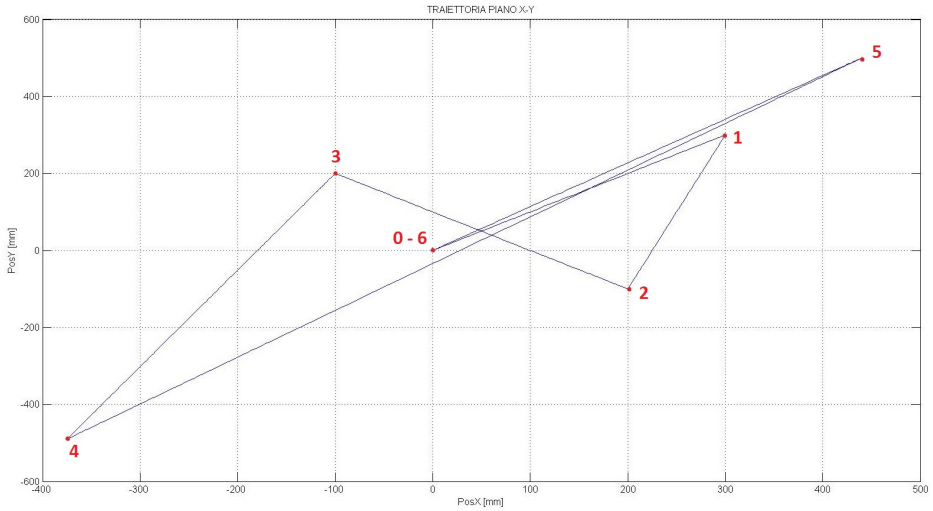

Fig. 5.2: Vista sul piano $X - Y$ della Sequenza di Movimentazione

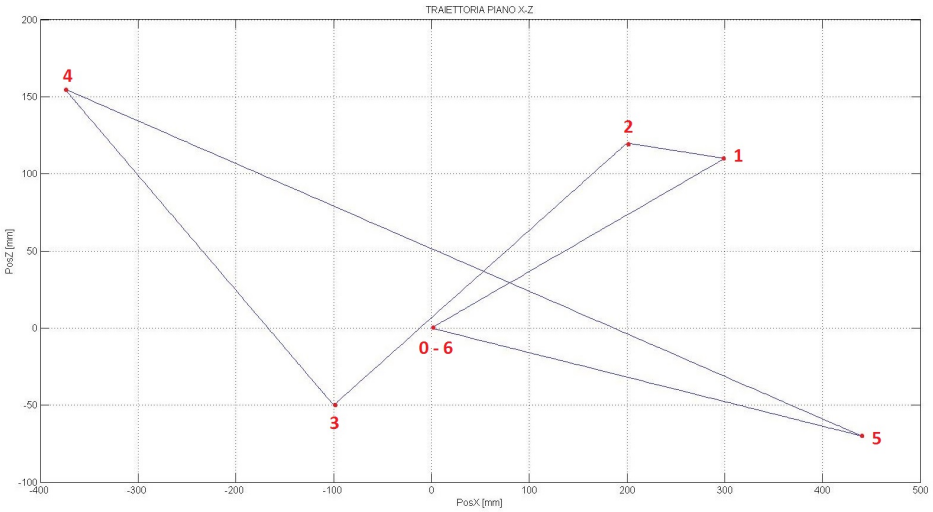

Fig. 5.3: Vista sul piano $X - Z$ della Sequenza di Movimentazione

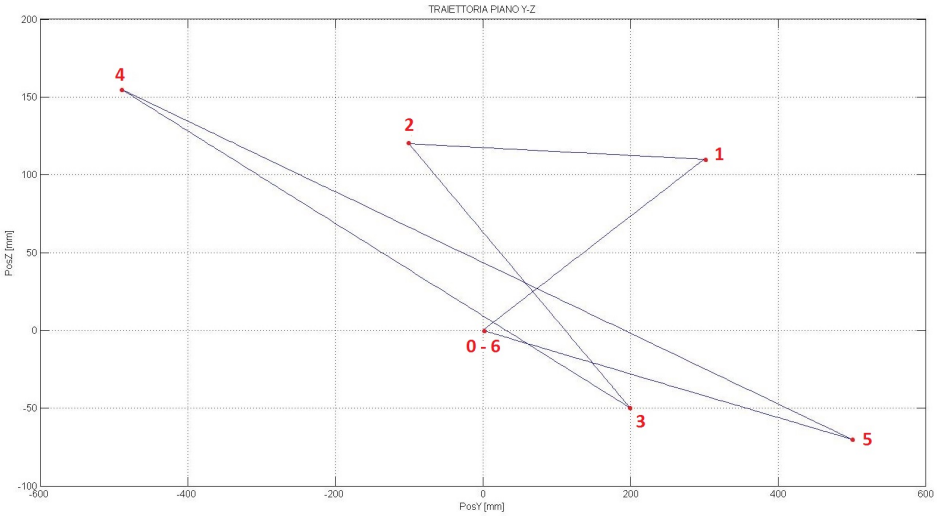

Fig. 5.4: Vista sul piano $Y - Z$ della Sequenza di Movimentazione

5.4 Blocchi Ricezione ed Invio Dati

Compito di questi due blocchi è quelli di creare il pacchetto dati da inviare o ricevere dal controllore. Il software realizzato in Simulink, infatti, viene eseguito dal real-time xPC Target, il quale è connesso al controllore Adept SmartController attraverso una rete Ethernet sulla quale si è scelto di sfruttare il protocollo UDP per la trasmissione dei dati. I valori di controllo necessari al manipolatore devono perciò essere convertiti in formato standard e successivamente impacchettati sotto forma di stringhe in un pacchetto UDP. Stesso discorso vale per i dati letti dagli encoder del manipolatore che devono venire spediti al real-time xPC dal controllore.

Prendendo come esempio i riferimenti per i tre assi X,Y,Z del manipolatore calcolati dal software Simulink, essi verranno convertiti ciascuno in una stringa di 4 byte. Successivamente tali stringhe saranno unite in una stringa unica di 12 byte, ulteriormente estesa con un'altra stringa di 4 byte contenente il checksum per il controllo degli errori. Si avrà perciò una stringa di 16 byte che, incapsulata in un pacchetto UDP, verrà spedita al controllore. In esso il codice V+ in esecuzione andrà a ricevere il pacchetto, lo scompatterà e dopo aver verificato che non ci siano stati errori di trasmissione utilizzerà i valori ricevuti per il controllo del manipolatore.

Discorso analogo ma speculare vale per il percorso inverso, ovvero l'invio dei dati degli encoder da parte del controllore, nel qual caso il software creato in Simulink andrà a scompattare il pacchetto ricevuto e successivamente utilizzerà i valori per il calcolo della cinematica diretta e la validazione dei risultati.

5.5 Calcolo della Cinematica Diretta

Come spiegato nel capitolo precedente, il controllore ha già la capacità di calcolare la cinematica diretta, e quindi di restituire la posizione dell'organo terminale, con un semplice comando implementato in V+.

Lo scopo di calcolare in maniera offline la posizione dell'end-effector tramite l'xPC è quello di sgravare il controllore da tale onere in modo da lasciare più spazio, durante il suo tempo ciclo, alle operazioni di movimentazione a priorità maggiore. La conoscenza della posizione dell'organo terminale è, infatti, non necessaria per il controllo del manipolatore, ma molto utile qualora si voglia fare un riscontro grafico sulla bontà della movimentazione.

Il calcolo della cinematica diretta parte direttamente dalla conoscenza degli angoli di rotazione degli attuatori, letti dagli encoder montati sugli stessi, dei dati costruttivi e della geometria del manipolatore, specificati in tabella 3.1 e nelle figure 3.5, 3.6, 3.7.

figura 5.5.

Si consideri ora la figura 3.7. I punti P_i individuano le intersezioni tra l'asse di rotazione di ciascuna coppia rotoidale che vincola la biella all'attuatore e l'asse longitudinale della biella stessa. Essi sono dei punti fissi e sono calcolati mediante le equazioni 3.2.1, 3.2.2, 3.2.3, 3.2.4 già esplicitate. Diversamente, i punti A_i individuano le intersezioni tra l'asse di ciascuna biella e l'asse del bilanciere ad essa collegato mediante le coppie rotoidali di figura 3.4, i quali variano la loro posizione in base al valore assunto dall'angolo di rotazione dell'attuatore q_i.

Per determinare in maniera esatta la posizione dell'organo terminale, pensato come punto di intersezione degli assi dei quattro bilancieri, si può usare un semplice sistema di quattro equazioni, ciascuna delle quali descrive una sfera di centro mobile A_i e di raggio fisso b_i. In questo modo si andrà ad identificare il punto $\mathbf{P_t}$ comune alle quattro sfere, che rappresenterà proprio la posizione del tool nello spazio di lavoro (figura 5.5).

Tale semplificazione si può sfruttare a patto di non far realizzare all'organo terminale movimentazioni contenenti rotazioni, caso nel quale la soluzione in forma chiusa del si-

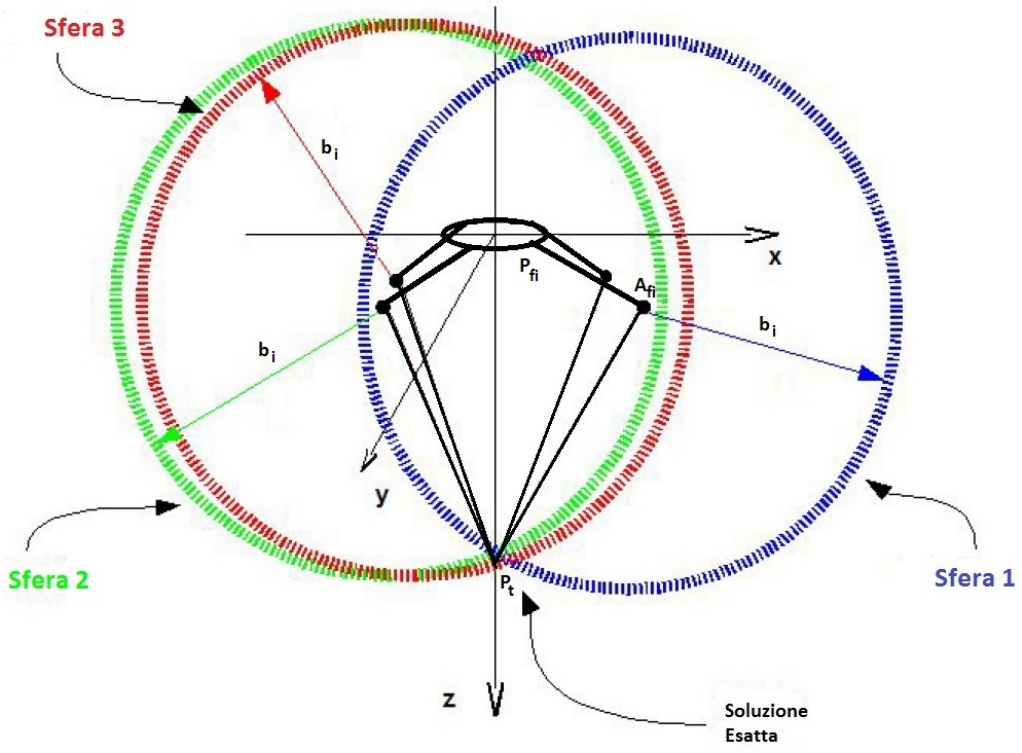

Fig. 5.5: Rappresentazione delle sfere di centro A_i e raggio b_i

stema non è possibile, rendendo necessario un calcolo iterativo mediante approssimazioni successive, molto più oneroso in termini computazionali e peraltro già implementato dalle funzioni interne al controllore Adept.

In realtà bisogna però stare attenti alla posizione effettiva del tool, che non si trova veramente sull'intersezione dei quattro bilancieri. Essi infatti non sono collegati tra di loro in un unico giunto, che farebbe degenerare i punti B_1, B_2, B_3, B_4 in un teorico punto unico, ma attraverso la piattaforma mobile visibile in figura 3.5, nella quale si può vedere l'effettiva posizione dell'organo terminale nel punto EE.

Quest'aspetto, che rende inutilizzabile il metodo sopracitato, può essere agilmente aggirato agendo sulla posizione dei punti P_i. L'idea di base è quella di traslare tali punti nello spazio, in modo tale da far coincidere i punti B_i con il punto di interesse EE, con la conseguente possibilità di far ritornare applicabile il metodo del sistema di equazioni di sfere.

Prendendo, quindi, come riferimenti la figura 3.5 e la tabella 3.1, si possono definire i quattro nuovi punti fittizi $\mathbf{P_{f_1}}$, $\mathbf{P_{f_2}}$, $\mathbf{P_{f_3}}$, $\mathbf{P_{f_4}}$ come:

$$\mathbf{P_{f_1}} = \left\{ \begin{array}{c} x_{P_{f_1}} \\ y_{P_{f_1}} \\ z_{P_{f_1}} \end{array} \right\} = \left\{ \begin{array}{r} x_{P_1} + l + l_2 \\ y_{P_1} + \frac{h}{2} + h_1 \\ -z_{off_1} - z_{off_2} \end{array} \right\} \tag{5.5.1}$$

$$\mathbf{P_{f_2}} = \left\{ \begin{array}{c} x_{P_{f_2}} \\ y_{P_{f_2}} \\ z_{P_{f_2}} \end{array} \right\} = \left\{ \begin{array}{r} x_{P_2} - l_1 \\ y_{P_2} + \frac{h}{2} + h_1 \\ -z_{off_1} - z_{off_2} \end{array} \right\} \tag{5.5.2}$$

$$\mathbf{P_{f_3}} = \left\{ \begin{array}{c} x_{P_{f_3}} \\ y_{P_{f_3}} \\ z_{P_{f_3}} \end{array} \right\} = \left\{ \begin{array}{r} x_{P_3} - l_1 \\ y_{P_3} - \frac{h}{2} - h_1 \\ -z_{off_1} - z_{off_2} \end{array} \right\} \tag{5.5.3}$$

$$\mathbf{P_{f_4}} = \left\{ \begin{array}{c} x_{P_{f_4}} \\ y_{P_{f_4}} \\ z_{P_{f_4}} \end{array} \right\} = \left\{ \begin{array}{r} x_{P_4} + l + l_2 \\ y_{P_4} - \frac{h}{2} - h_1 \\ -z_{off_1} - z_{off_2} \end{array} \right\} \tag{5.5.4}$$

Si fa notare come le quote lungo l'asse Z sono state a loro volta già modificate della quantità z_{off_2} che corrisponde alla traslazione lungo l'asse verticale dell'organo terminale rispetto al piano della piattaforma mobile.

A loro volta i punti A_i (figura 5.5),corrispondenti ai centri delle sfere considerate, vengono sostituiti con i centri sfera fittizi $\mathbf{A_{f_i}}$ mediante le equazioni:

$$\mathbf{A_{f_i}} = \left\{ \begin{array}{c} x_i \\ y_i \\ z_i \end{array} \right\} = \left\{ \begin{array}{r} x_{P_{f_i}} + sign(x_{P_i})\frac{\sqrt{2}}{2} a_i \cos q_i \\ y_{P_{f_i}} + sign(y_{P_i})\frac{\sqrt{2}}{2} a_i \cos q_i \\ z_{P_{f_i}} - a_i \sin q_i \end{array} \right\} \tag{5.5.5}$$

A questo punto si può scrivere il sistema di equazioni che, attraverso la sua risoluzione, andrà ad individuarmi in maniera esatta la posizione dell'end effector:

$$\left\{ \begin{array}{l} (x_{P_t} - x_1)^2 + (y_{P_t} - y_1)^2 + (z_{P_t} - z_1)^2 = b_i^2 \\ (x_{P_t} - x_2)^2 + (y_{P_t} - y_2)^2 + (z_{P_t} - z_2)^2 = b_i^2 \\ (x_{P_t} - x_3)^2 + (y_{P_t} - y_3)^2 + (z_{P_t} - z_3)^2 = b_i^2 \\ (x_{P_t} - x_4)^2 + (y_{P_t} - y_4)^2 + (z_{P_t} - z_4)^2 = b_i^2 \end{array} \right. \tag{5.5.6}$$

Il sistema 5.5.6 risulta chiaramente sovradeterminato in quanto le incognite in esso presenti sono tre, ovvero x_{P_t}, y_{P_t}, z_{P_t}, mentre le equazioni sono quattro, una per ogni catena cinematica, per cui ne basteranno solamente tre per la risoluzione analitica del sistema usando ad esempio il metodo della sostituzione.

Per semplificare le equazioni risolutive si definiscono i parametri:

$$A = \frac{y_2 - y_1}{x_1 - x_2} \tag{5.5.7}$$

$$B = \frac{z_2 - z_1}{x_1 - x_2} \tag{5.5.8}$$

$$C = \frac{x_1^2 - x_2^2 + y_1^2 - y_2^2 + z_1^2 - z_2^2}{2(x_1 - x_2)} \tag{5.5.9}$$

$$D = \frac{B(x_2 - x_3) + z_2 - z_3}{A(x_3 - x_2) + (y_3 - y_2)} \tag{5.5.10}$$

$$E = \frac{2C(x_2 - x_3) + x_3^2 - x_2^2 + y_3^2 - y_2^2 + z_3^2 - z_2^2}{2A(x_3 - x_2) + 2(y_3 - y_2)} \tag{5.5.11}$$

$$F = A^2D^2 + B^2 + 2ABD + D^2 + 1 \tag{5.5.12}$$

$$G = 2(A^2ED + ABE + ACD + BC + DE - ADx_3 - Bx_3 - Dy_3 - z_3) \tag{5.5.13}$$

$$H = A^2E^2 + C^2 + E^2 + 2ACE - 2AEx_3 - 2Cx_3 - 2Ey_3 + x_3^2 + y_3^2 + z_3^2 - b_i^2 \tag{5.5.14}$$

Dopo semplici passaggi matematici la soluzione del sistema che si ottiene risulta:

$$\mathbf{P_t} = \left\{ \begin{array}{c} x_{P_t} \\ y_{P_t} \\ z_{P_t} \end{array} \right\} = \left\{ \begin{array}{r} Ay_{P_t} + Bz_{P_t} + C \\ Dz_{P_t} + E \\ \frac{-G-\sqrt{G^2-4FH}}{2F} \end{array} \right\} \tag{5.5.15}$$

che mi identifica in maniera esatta la posizione spaziale dell'organo terminale. Si deve specificare in aggiunta che tale metodo di calcolo offline della cinematica diretta mediante soluzione in forma chiusa non è in grado di determinare anche l'angolo di rotazione dell'organo terminale, per il quale l'unica strada possibile risulta essere il metodo iterativo, peraltro già implementato nella funzione *HERE* presente in V+.

5.6 Calcolo della Legge di Moto

Uno dei blocchi più importanti realizzati nel software Simulink che consente la completa gestione della pianificazione del moto è quello designato alla generazione della traiettoria effettiva da inviare come riferimento. Il suo compito è quello di calcolare la legge di moto da far eseguire al manipolatore, prendendo come parametri di ingresso le locazioni da raggiungere fornite dal blocco Generatore dei Riferimenti, i limiti di velocità ed accelerazione scelti dall'operatore e il tipo di legge oraria decisa.

I limiti meccanici ai quali si fa riferimento per l'impostazione della velocità massima e dell'accelerazione massima sono quelli dichiarati sul datasheet del costruttore, nel quale viene attestata una velocità massima ottenibile dall'organo terminale di $10m/s$, con un'accelerazione massima di $150m/s^2$. Durante i test eseguiti non si è mai arrivati a tali impostazioni ma si è sempre cercato di mantenere un certo margine di sicurezza al fine di evitare possibili danneggiamenti meccanici della struttura del manipolatore.

La scelta della legge di moto è, in seconda istanza, un aspetto fondamentale per quanto riguarda la dolcezza finale della movimentazione. Tale scelta deve venire fatta con cura in base al tipo di lavorazione che si andrà ad eseguire. Per movimentazioni di carichi oscillanti o recipienti contenenti liquidi la legge di moto più consigliabile sarà una a maggiore dolcezza, come possono essere leggi polinomiali di quinto o settimo grado, oppure leggi di natura trigonometrica, eventualmente modificate in modo opportuno.

Le leggi di moto realizzate nel software sviluppato sono:

- Triangolare in Velocità
- Trapezoidale in Velocità
- Polinomiale di 3° grado
- Polinomiale di 5° grado
- Polinomiale di 7° grado
- Freudestein 1-3
- Freudestein 1-3-5
- Gutman 1-3
- Armonica
- Cicloidale
- Trapezoidale in Accelerazione
- Trapezoidale Modificata

- Sinusoidale Modificata

Il calcolo della legge oraria viene realizzato considerando la traiettoria di movimentazione, tra la locazione attuale e quella da raggiungere, come una retta. Sapendo il punto di partenza e quello di arrivo è possibile calcolare il valore assoluto dello spostamento come

$$d = \sqrt{(x_f - x_i)^2 + (y_f - y_i)^2 + (z_f - z_i)^2} \tag{5.6.1}$$

dove x_i, y_i, z_i sono le coordinate della locazione di partenza del tool, mentre x_f, y_f, z_f sono le coordinate della locazione di arrivo, entrambe espresse nel sistema di riferimento assoluto. E' quindi su tale spostamento d che viene calcolata la legge di moto e successivamente ogni punto di essa viene scomposto nelle coordinate x, y, z tramite una semplice proporzione matematica. I riferimenti da inviare al controllore devono infatti essere definiti singolarmente per le tre componenti cartesiane più l'eventuale rotazione, in modo da rendere immediato, alla ricezione del pacchetto, il loro utilizzo per il calcolo degli step e il conseguente utilizzo nel comando ALTER().

Di seguito vengono illustrate le leggi orarie implementate con le opportune scelte effettuate nella loro realizzazione ed i risultati ottenuti. Da tener presente che la legge di moto è calcolata, come precedentemente scritto, sullo spostamento assoluto nello spazio, per cui i limiti di velocità ed accelerazioni massimi impostati non si raggiungeranno mai sui singoli assi, salvo casi dove lo spostamento interessa una sola direzione dello spazio di lavoro, nel qual caso lo spostamento assoluto corrisponderà allo spostamento in tale direzione.

Si utilizzeranno i simboli a, v, s, t rispettivamente per l'accelerazione, la velocità,lo spostamento ed il tempo di esecuzione totale. Il tempo totale di movimentazione T verrà calcolato preventivamente in base ai coefficienti di accelerazione e velocità C_a e C_v della legge di moto scelta ed in base alla velocità massima $vel_{max} = 500mm/s$ e all'accelerazione massima $acc_{max} = 1000mm/s^2$ impostate:

$$T = \sqrt{\frac{d}{acc_{max}} C_a} \tag{5.6.2}$$

oppure

$$T = \frac{d}{vel_{max}} C_v \tag{5.6.3}$$

verrà scelto il T risultante maggiore tra i due metodi, in modo da considerare sempre il vincolo più stringente tra quello di velocità e quello di accelerazione. Per le leggi in cui le fasi di accelerazione e decelerazione devono venire calcolate automaticamente in modo da minimizzare il tempo di movimentazione, il periodo T verrà calcolato successivamente ad essi come semplice somma delle varie fasi.

Verrà inoltre utilizzato il tempo normalizzato:

$$q = \frac{t - t_0}{T} \tag{5.6.4}$$

dove t_0 è l'istante di inizio della legge di moto, mentre t parte dall'inizio dell'esecuzione del programma.

5.6.1 Legge Triangolare in Velocità

La prima legge implementata è stata la legge triangolare in velocità. Essa è caratterizzata da due fasi di movimentazione: la prima nella quale si ha un'accelerazione costante e quindi una velocità che cresce linearmente; una seconda in cui si ha una decelerazione costante con conseguente velocità che decresce in modo lineare. Si è scelto di mantenere la

Tab. 5.1: Parametri delle Leggi di Moto

Legge	C_v	C_a	Dolcezza
Triangolare Velocità	2	4	*
Trapezoidale Velocità	var	var	*
Polinomiale 3° grado	1.5	6	**
Armonica	1.57	4.9	**
Trapezoidale Accelerazione	var	var	***
Polinomiale 5° grado	1.87	5.7	****
Cicloidale	2	6.28	****
Trapezoidale Modificata	2	4.9	****
Sinusoidale Modificata	1.75	5.5	****
Freudestein 1-3	2	5.4	****
Gutman 1-3	2	5.1	****
Freudestein 1-3-5	2	5.1	****
Polinomiale 7° grado	2.2	7.5	*****

legge simmetrica per cui il tempo di accelerazione risulta uguale al tempo di decelerazione, in modo da raggiungere la velocità massima a metà dello spostamento totale.

$$a = \begin{cases} acc_{max} & 0 < q \leq \frac{1}{2} \\ -acc_{max} & \frac{1}{2} < q \leq 1 \end{cases} \tag{5.6.5}$$

5.6.2 Legge Trapezoidale in Velocità

Questa legge oraria, molto simile alla legge triangolare, si differenzia da essa per contenere una fase intermedia tra le due di accelerazione e decelerazione, nella quale la velocità rimane costante alla velocità massima. Anche in questo caso il tempo della fase di accelerazione è uguale al tempo della fase di decelerazione, ed essi vengono calcolati automaticamente dal programma tali da far raggiungere nel minor tempo possibile la velocità massima impostata,non sforando mai in ogni caso l'accelerazione massima, in modo da minimizzare il tempo totale di movimentazione. La frazione del periodo T in cui si hanno le fasi di accelerazione e decelerazione è definito come λ.

$$a = \begin{cases} acc_{max} & 0 < q \leq \lambda \\ 0 & \lambda < q \leq (1-\lambda) \\ -acc_{max} & (1-\lambda) < q \leq 1 \end{cases} \tag{5.6.6}$$

5.6.3 Legge Polinomiale di 3° grado

La legge cubica è la prima legge polinomiale realizzata. Essa è caratterizzata da una dolcezza di movimentazione maggiore delle leggi in cui l'accelerazione è costante a tratti, sebbene anche in questo caso l'accelerazione presenta delle discontinuità all'inizio e alla fine del moto, generando possibili inneschi di vibrazioni. L'equazione per il calcolo della traiettoria è:

$$s = d[3q^2 - 2q^3] \quad 0 \leq q \leq 1 \tag{5.6.7}$$

5.6.4 Legge Polinomiale di 5° grado

La legge polinomiale di 5°grado è una legge oraria decisamente più dolce della cubica. Essa non presenta infatti discontinuità nell'accelerazione, mentre contiene ancora due discontinuità sul jerk. L'equazione per il calcolo della traiettoria è:

$$s = d[10q^3 - 15q^4 + 6q^5] \quad 0 \leq q \leq 1 \tag{5.6.8}$$

5.6.5 Legge Polinomiale di 7° grado

La legge polinomiale di 7° grado è una legge estremamente morbida,la più dolce tra tutte quelle realizzate, da utilizzare qualora si avessero problemi seri di vibrazioni. La sua dolcezza, caratterizzata da continuità sia sull'accelerazione sia sul jerk, porta come conseguenza velocità ed accelerazioni notevolmente superiori alle altre leggi orarie. L'equazione per il calcolo della traiettoria è:

$$s = d[35q^4 - 84q^5 + 70q^6 - 20q^7] \quad 0 \leq q \leq 1 \tag{5.6.9}$$

5.6.6 Legge Freudestein 1-3

Questa è la prima delle leggi di moto basate su scomposizione in serie di Fourier e successivo filtraggio delle componenti spettrali. Esse sono caratterizzate da una notevole dolcezza, paragonabile a quella di una legge polinomiale di 5° grado se non superiore. Sono state largamente utilizzate nel settore meccanico per lo sviluppo dei profili delle camme grazie alla loro morbidezza, pagata purtroppo da coefficienti di velocità maggiori. Nella Freudestein 1-3 vengono considerate solo la prima e la terza armonica della sua scomposizione in serie di Fourier, scalate successivamente in modo da far risultare l'ampiezza di spostamento corretta. L'equazione per il calcolo della traiettoria è:

$$s = d[q - \tfrac{1}{2\pi}\tfrac{27}{28}\sin(2\pi q) + \tfrac{1}{84}\sin(6\pi q)] \quad 0 \leq q \leq 1 \tag{5.6.10}$$

5.6.7 Legge Freudestein 1-3-5

Tale legge è del tutto analoga alla precedente come metodo di sviluppo, con la sola differenza dell'ulteriore presenza della quinta armonica dello sviluppo in serie di Fourier. L'equazione per il calcolo della traiettoria è:

$$s = d[q - \tfrac{1}{2\pi}\tfrac{1125}{1192}\sin(2\pi q) + \tfrac{1}{54}\sin(6\pi q) + \tfrac{1}{1250}\sin(10\pi q)] \quad 0 \leq q \leq 1 \tag{5.6.11}$$

5.6.8 Legge Gutman 1-3

Altra legge molto famosa basata sulla scomposizione in serie di Fourier, anch'essa caratterizzata da una particolare dolcezza utile contro possibili fenomeni di vibrazioni. L'equazione per il calcolo della traiettoria è:

$$s = d[q - \tfrac{1}{\pi}(\tfrac{15}{32}\sin(2\pi q) + \tfrac{1}{96}\sin(6\pi q))] \quad 0 \leq q \leq 1 \tag{5.6.12}$$

5.6.9 Legge Armonica

Legge molto simile alla polinomiale di 3° grado, presenta due discontinuità in accelerazione quindi non risulta particolarmente dolce. Indicata per movimentazioni rest-to-rest ha buoni coefficienti di velocità e accelerazione massimi. L'equazione per il calcolo della traiettoria è:

$$s = \tfrac{d}{2}(1 - \cos(\pi q)) \quad 0 \leq q \leq 1 \tag{5.6.13}$$

5.6.10 Legge Cicloidale

La legge di moto cicloidale ha la particolarità di avere diagrammi di spostamento, velocità e accelerazione continui nel tempo e quindi risulta abbastanza dolce. Tuttavia i valori massimi di velocità e accelerazione risultano più elevati di quelli ottenibili con altre leggi. Il diagramma delle accelerazioni è una sinusoide, da cui il diagramma della velocità è un coseno traslato verso l'alto e lo spostamento risulta essere un seno più una rampa. Tale legge è nata per soddisfare esclusivamente profili rest-to-rest, è molto simile alla legge polinomiale di 5° grado ma con un contenuto armonico che la rende un più dolce. Usata particolarmente negli intermittori e nelle camme elettroniche è andata un po' in disuso negli anni. L'equazione per il calcolo della traiettoria è:

$$s = d[q - \tfrac{1}{2\pi}\sin(2\pi q)] \quad 0 \leq q \leq 1 \tag{5.6.14}$$

5.6.11 Legge Trapezoidale in Accelerazione

La legge trapezoidale in accelerazione è una variante della legge ad accelerazione costante. Anziché avere un profilo di accelerazione costante a tratti, essa presenta un jerk costante a tratti e di conseguenza un'accelerazione di tipo trapezoidale. Le fasi di accelerazione e decelerazione λ sono calcolate automaticamente per minimizzare i tempi di esecuzione alo stesso modo della legge di base, per cui le velocità massime ottenibili sono identiche. Variano invece le accelerazioni massime in base alle fasi in cui il jerk risulta positivo o negativo, definite dal parametro λ_{t_R} che viene fissato direttamente come parametro d'ingresso al blocco di calcolo delle traiettorie. Nelle prove effettuate è stato scelto $\lambda_{t_R} = \frac{1}{8}$ che mi garantisce un buon compromesso al fine di avere accelerazioni non troppo elevate. Per l'analisi della sua implementazione in Matlab si faccia riferimento al codice presente in appendice.

5.6.12 Legge Trapezoidale Modificata

Molto simile alla legge precedente, essa è caratterizzata da un profilo di accelerazione nel quale le fasi definite dal parametro λ_{t_R}, anziché essere delle rampe, sono sostituite da raccordi di tipo sinusoidale, garantendomi una dolcezza di movimento maggiore, contenendo al tempo stesso i coefficienti di velocità e accelerazione C_v e C_a. Può essere considerata una delle leggi migliori per applicazioni rest-to-rest nelle quali si hanno problemi di oscillazioni.

5.6.13 Legge Sinusoidale Modificata

Quest'ultima legge parte dal profilo della legge precedente diminuendo ulteriormente il numero di discontinuità sul jerk, passando da quattro a due. Questo è possibile raccordando le fasi di accelerazione e decelerazione con un unico profilo sinusoidale, mantenendo le fasi λ_{t_R} quelle della legge trapezoidale modificata. E' una legge ancora più dolce della precedente, a patto di avere velocità e accelerazioni assolute più elevate. A differenza della legge sinusoidale, nella quale il profilo di accelerazione è dato da un unico seno e perciò con possibilità di modifica ridotte, qui si hanno tre sinusoidi raccordate, con la conseguente possibilità di modificarne le rispettive frequenze, variando perciò i tempi di accelerazione e decelerazione a proprio piacimento, riuscendo a contenere C_v e C_a ed al tempo stesso mantenendone la dolcezza caratterizzante la legge cicloidale.

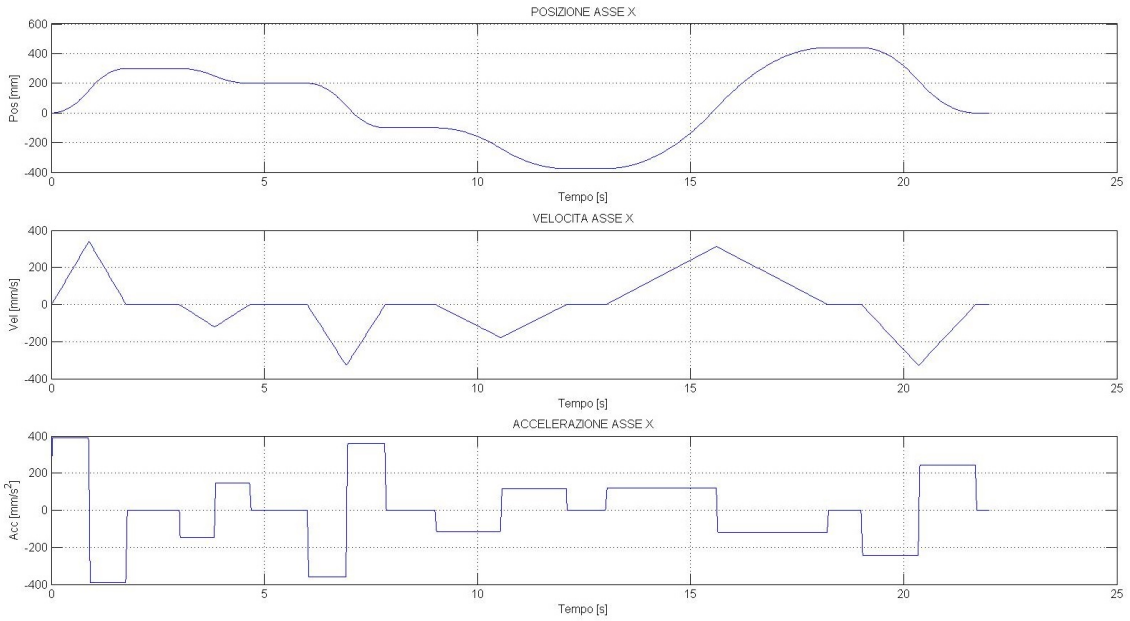

Fig. 5.6: Legge Triangolare: posizione, velocità ed accelerazione dell'asse X

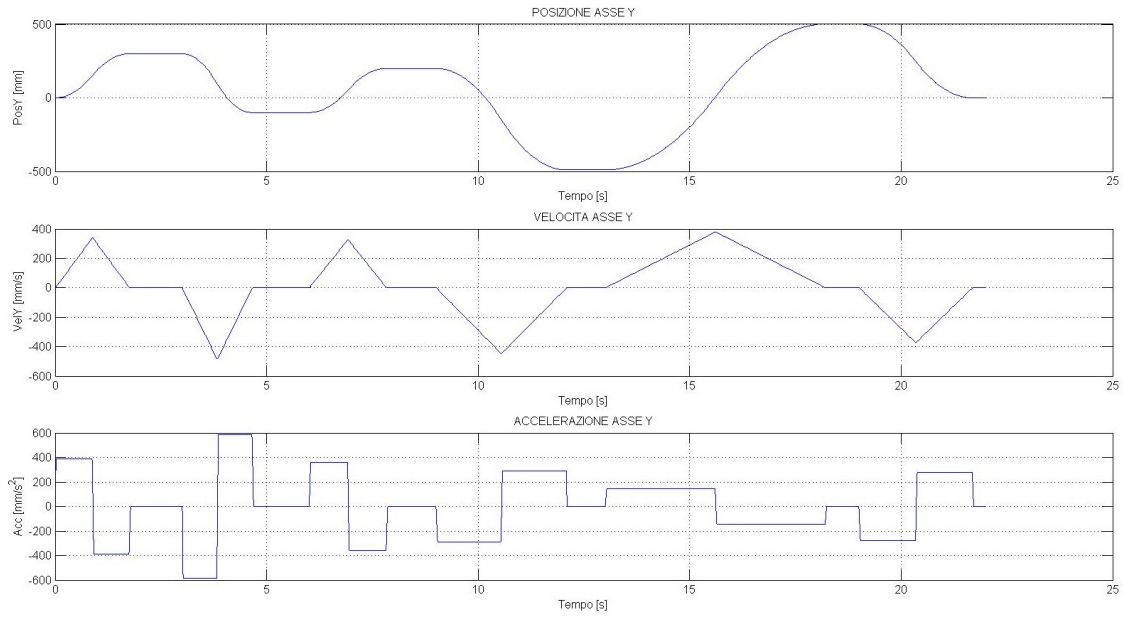

Fig. 5.7: Legge Triangolare: posizione, velocità ed accelerazione dell'asse Y

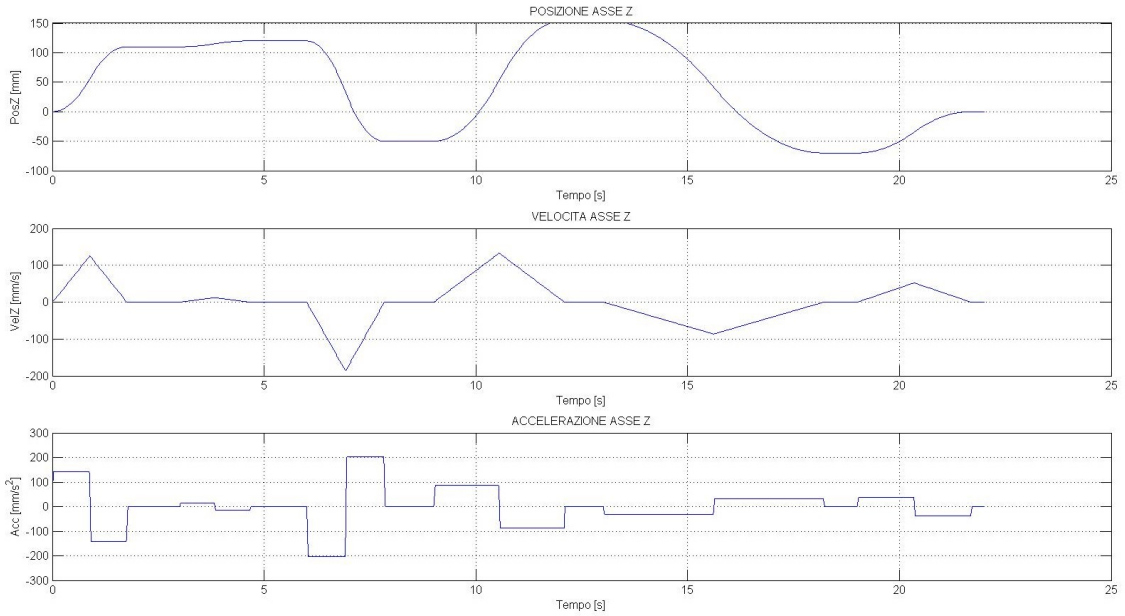

Fig. 5.8: Legge Triangolare: posizione, velocità ed accelerazione dell'asse Z

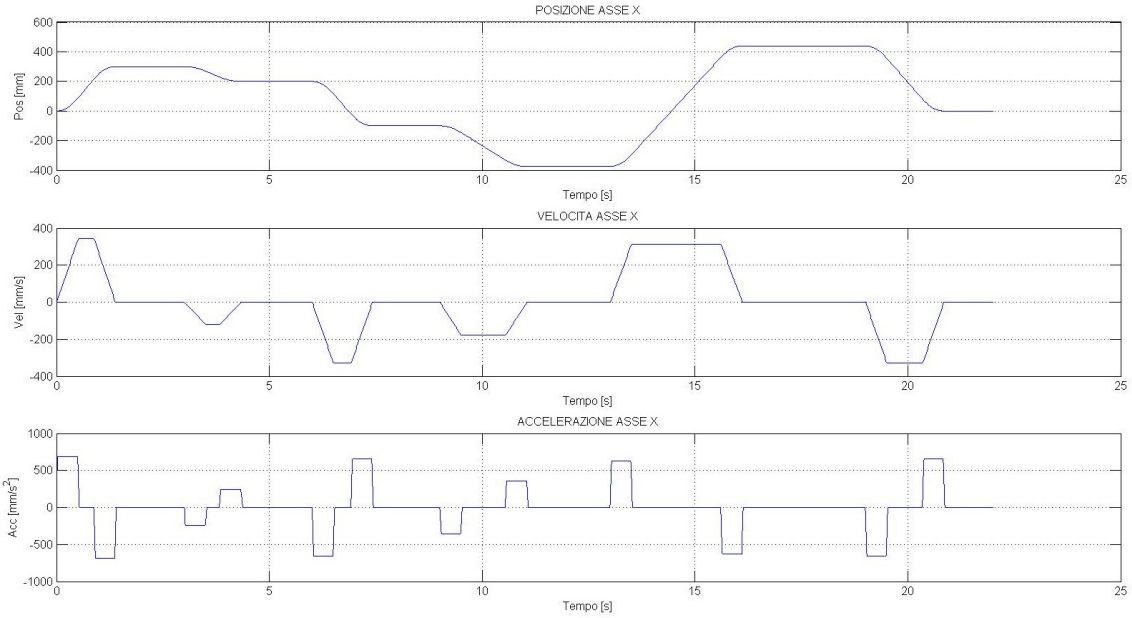

Fig. 5.9: Legge Trapezoidale in Velocità: posizione, velocità ed accelerazione dell'asse X

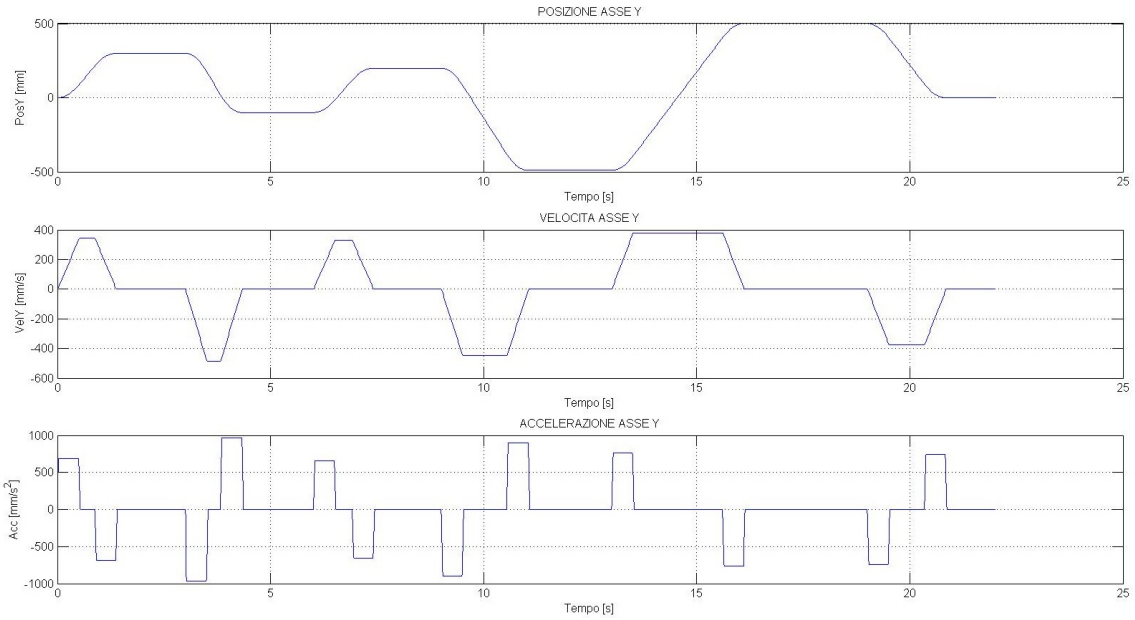

Fig. 5.10: Legge Trapezoidale in Velocità: posizione, velocità ed accelerazione dell'asse Y

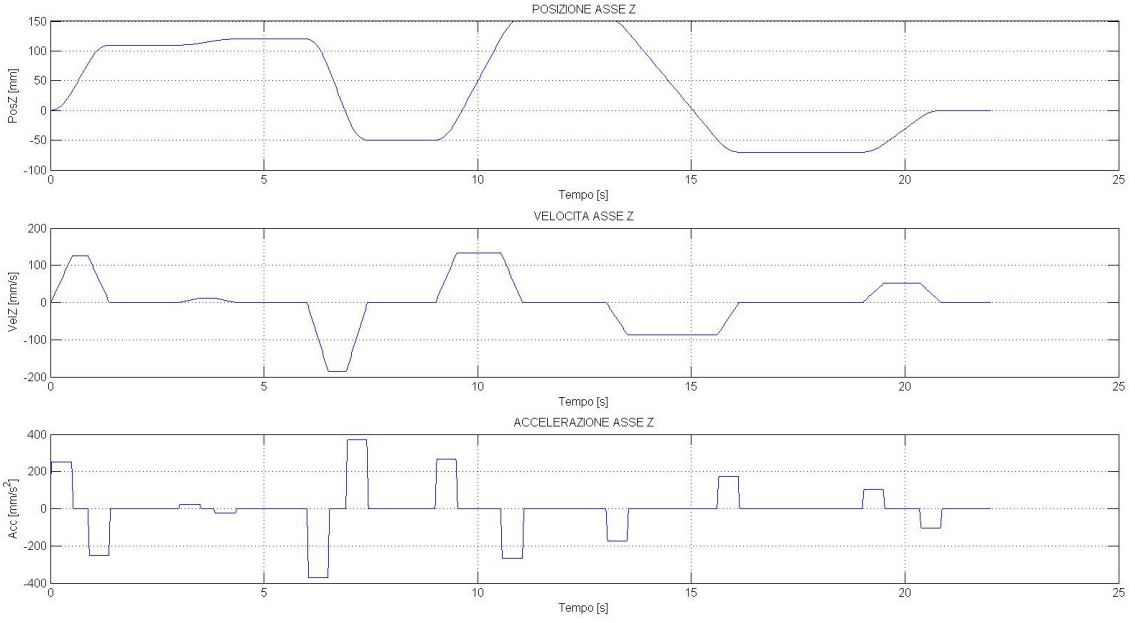

Fig. 5.11: Legge Trapezoidale in Velocità: posizione, velocità ed accelerazione dell'asse Z

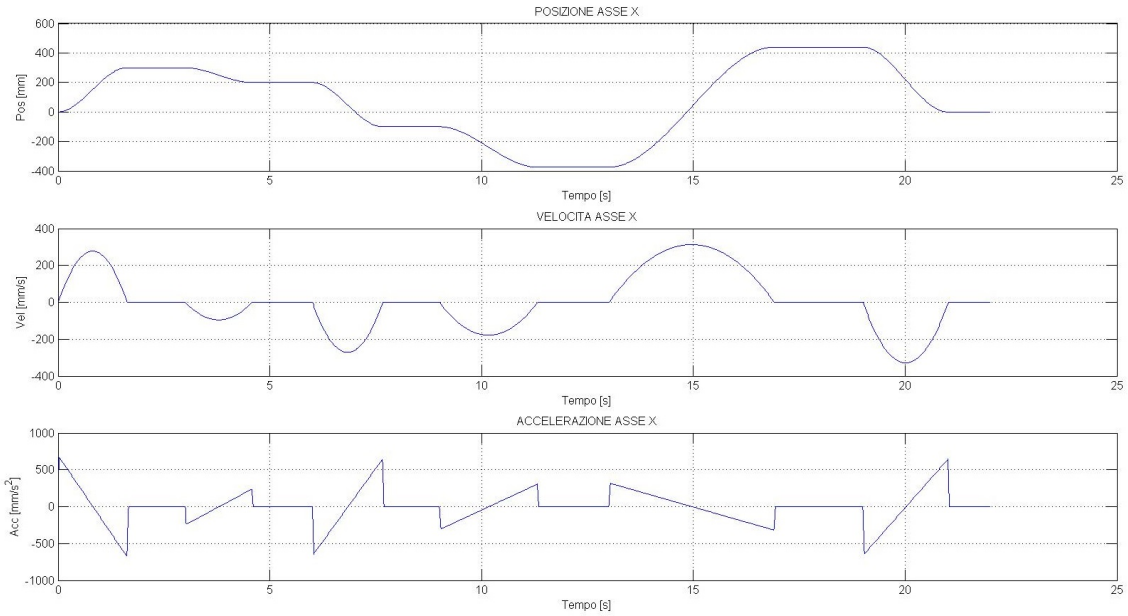

Fig. 5.12: Legge Polinomiale 3° grado: posizione, velocità ed accelerazione dell'asse X

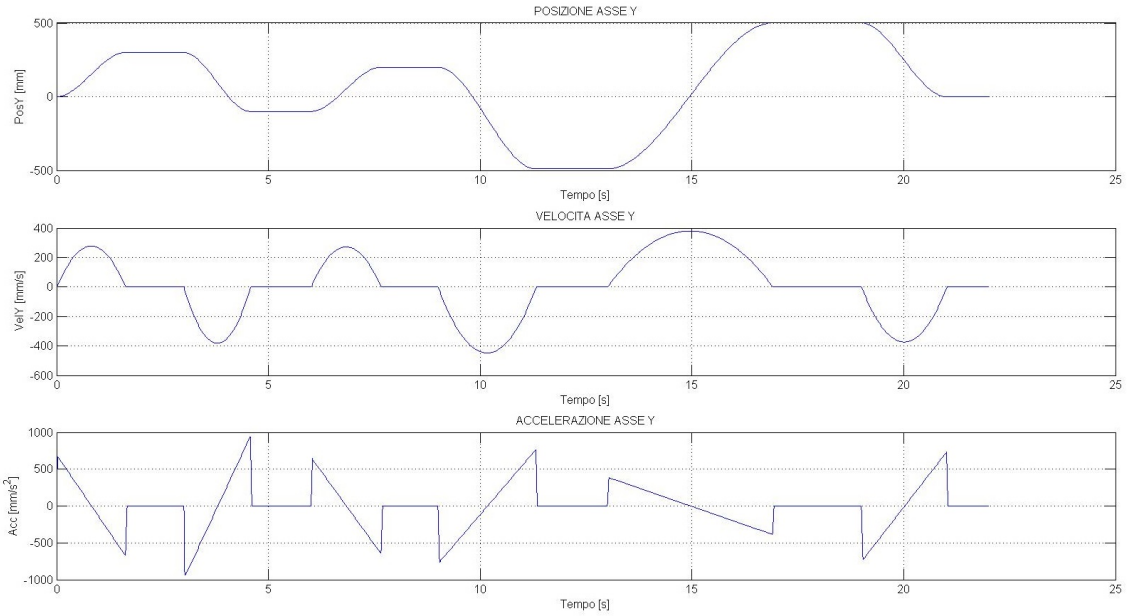

Fig. 5.13: Legge Polinomiale 3° grado: posizione, velocità ed accelerazione dell'asse Y

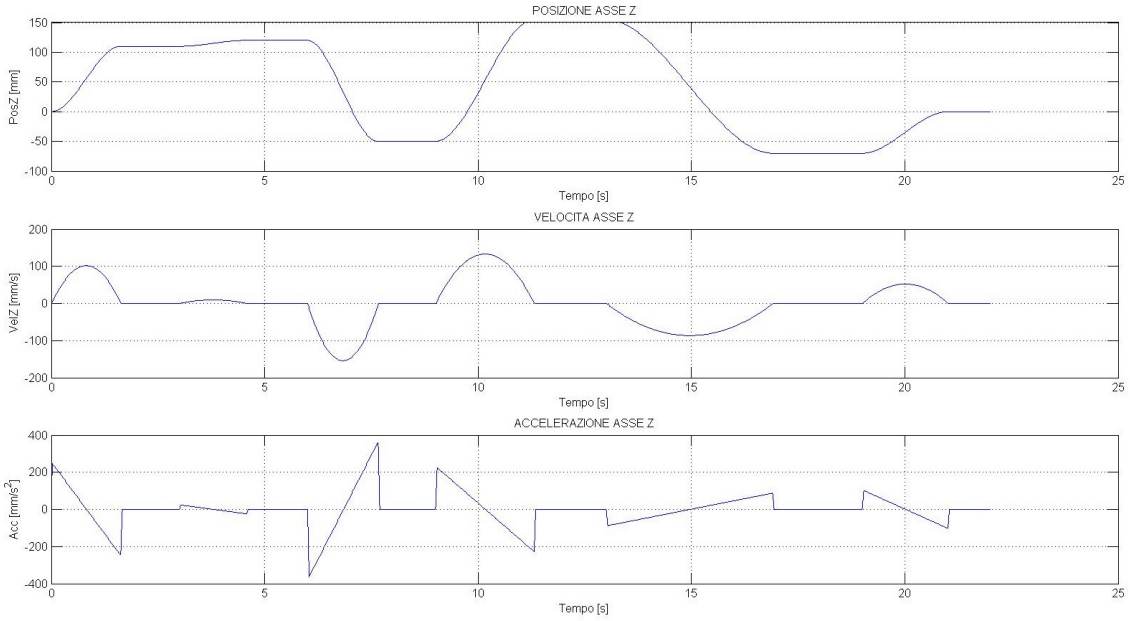

Fig. 5.14: Legge Polinomiale 3° grado: posizione, velocità ed accelerazione dell'asse Z

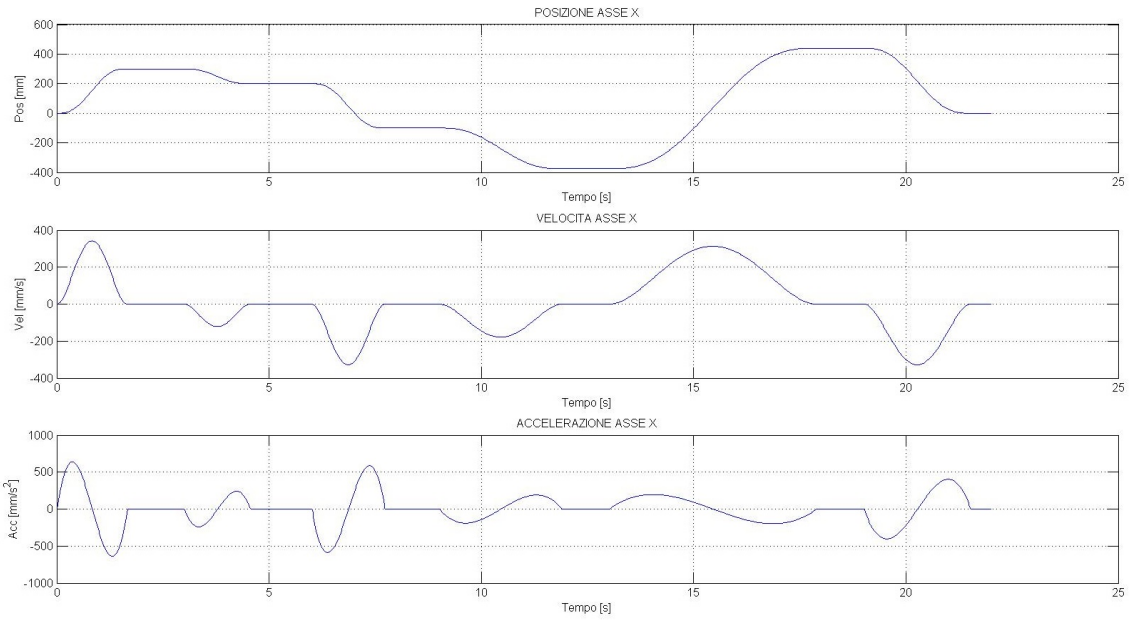

Fig. 5.15: Legge Polinomiale 5° grado: posizione, velocità ed accelerazione dell'asse X

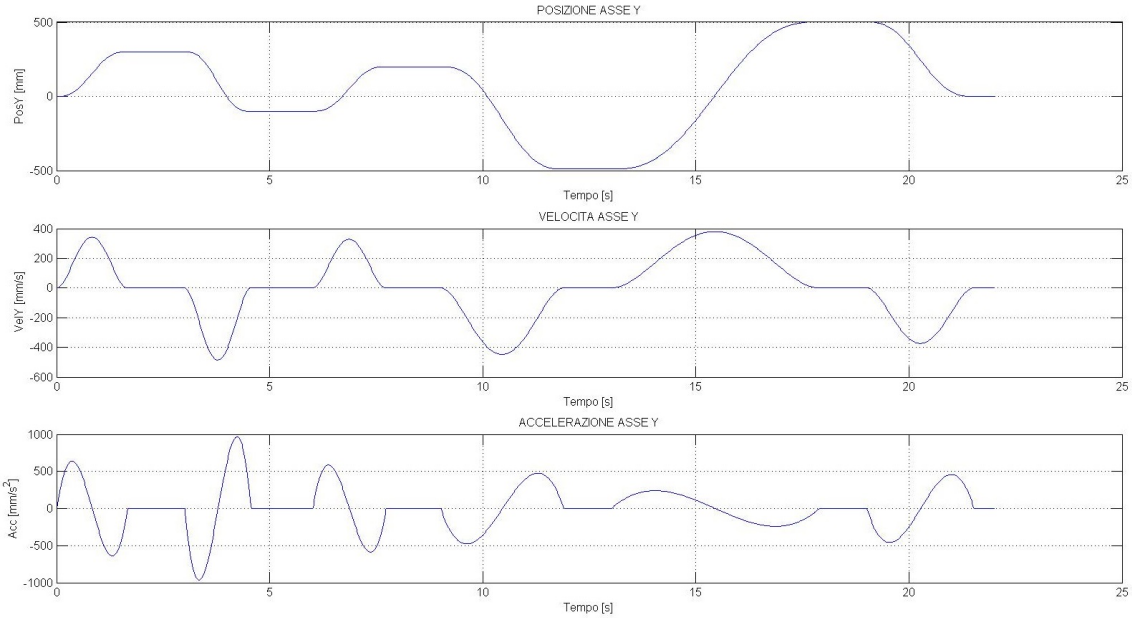

Fig. 5.16: Legge Polinomiale 5° grado: posizione, velocità ed accelerazione dell'asse Y

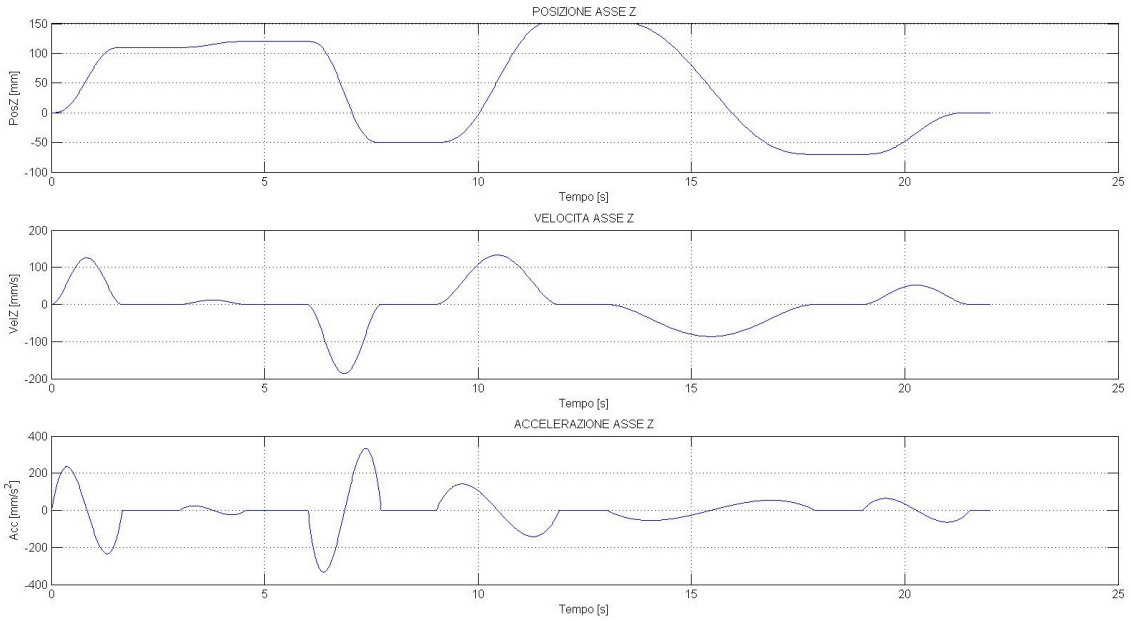

Fig. 5.17: Legge Polinomiale 5° grado: posizione, velocità ed accelerazione dell'asse Z

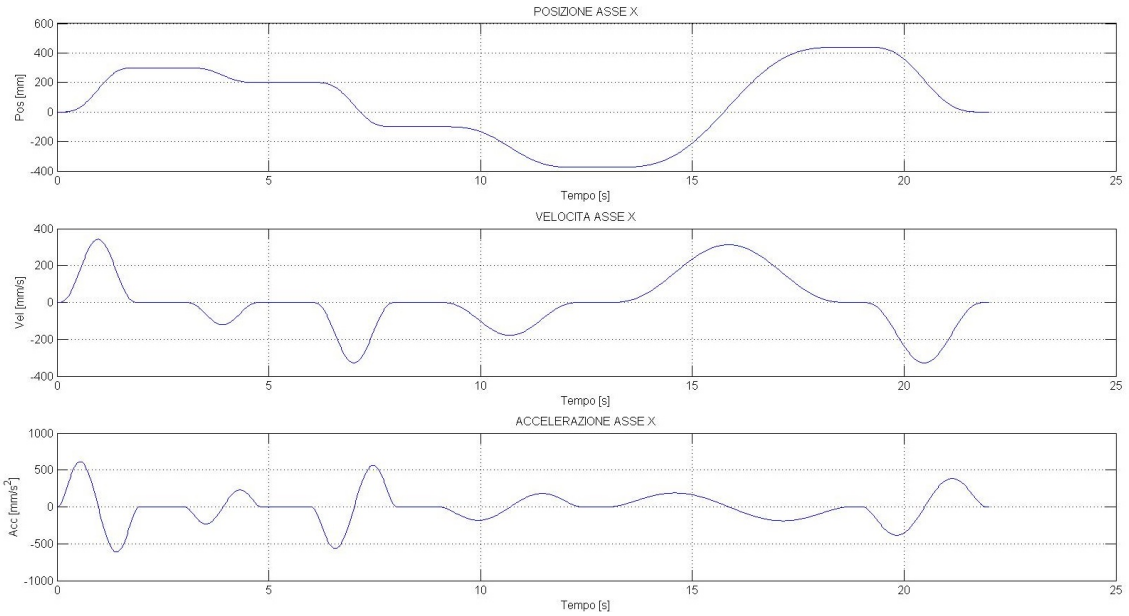

Fig. 5.18: Legge Polinomiale 7° grado: posizione, velocità ed accelerazione dell'asse X

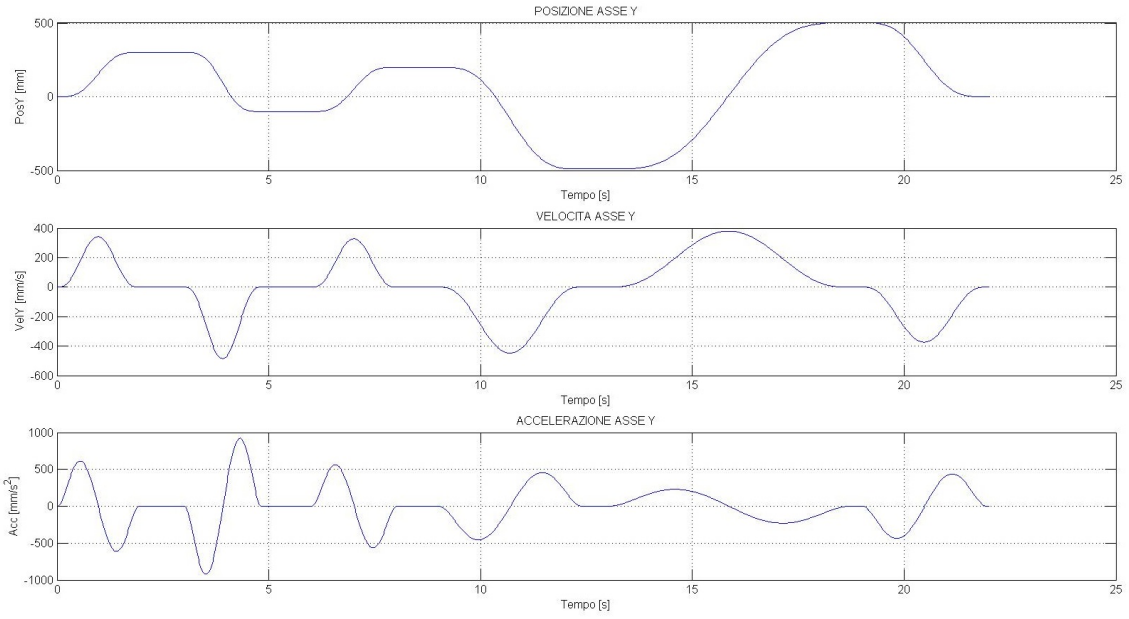

Fig. 5.19: Legge Polinomiale 7° grado: posizione, velocità ed accelerazione dell'asse Y

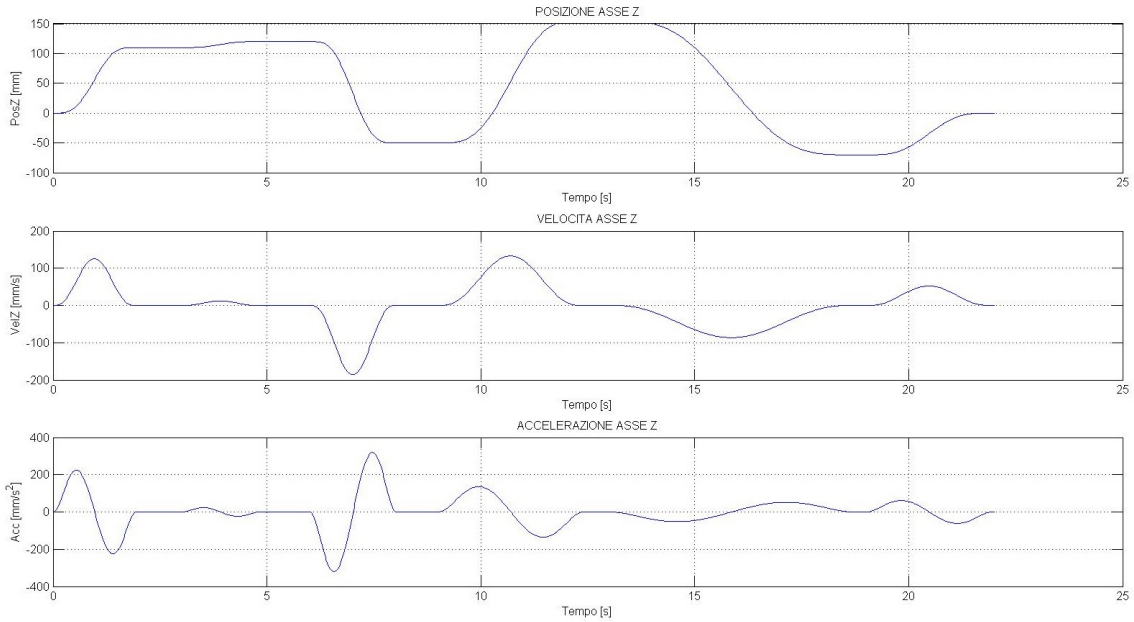

Fig. 5.20: Legge Polinomiale 7° grado: posizione, velocità ed accelerazione dell'asse Z

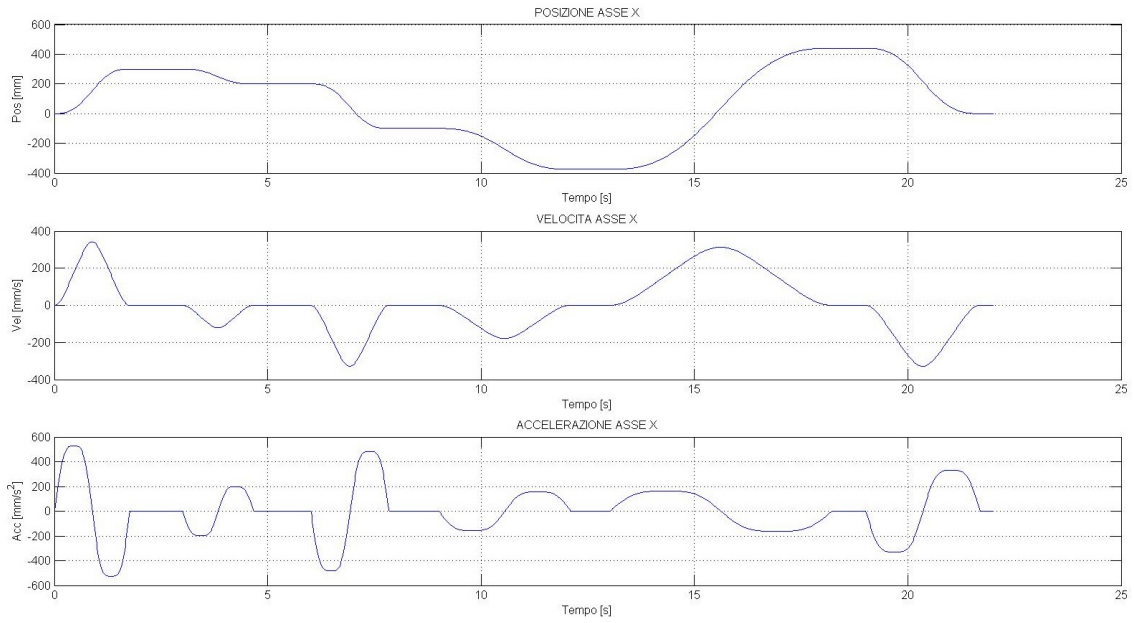

Fig. 5.21: Legge Freudestein 1-3: posizione, velocità ed accelerazione dell'asse X

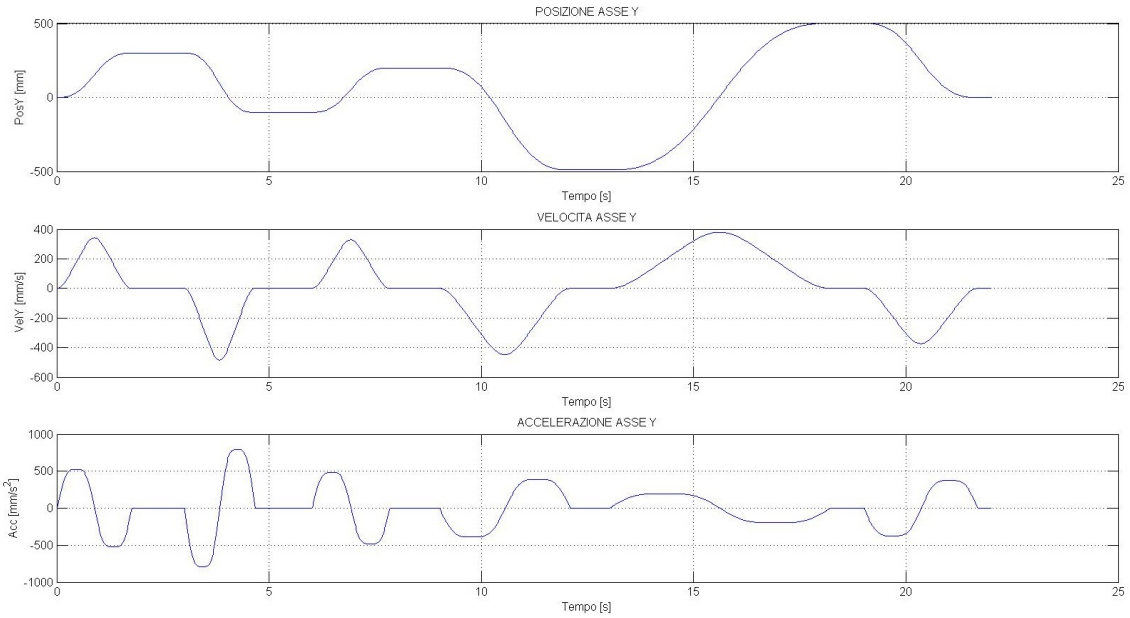

Fig. 5.22: Legge Freudestein 1-3: posizione, velocità ed accelerazione dell'asse Y

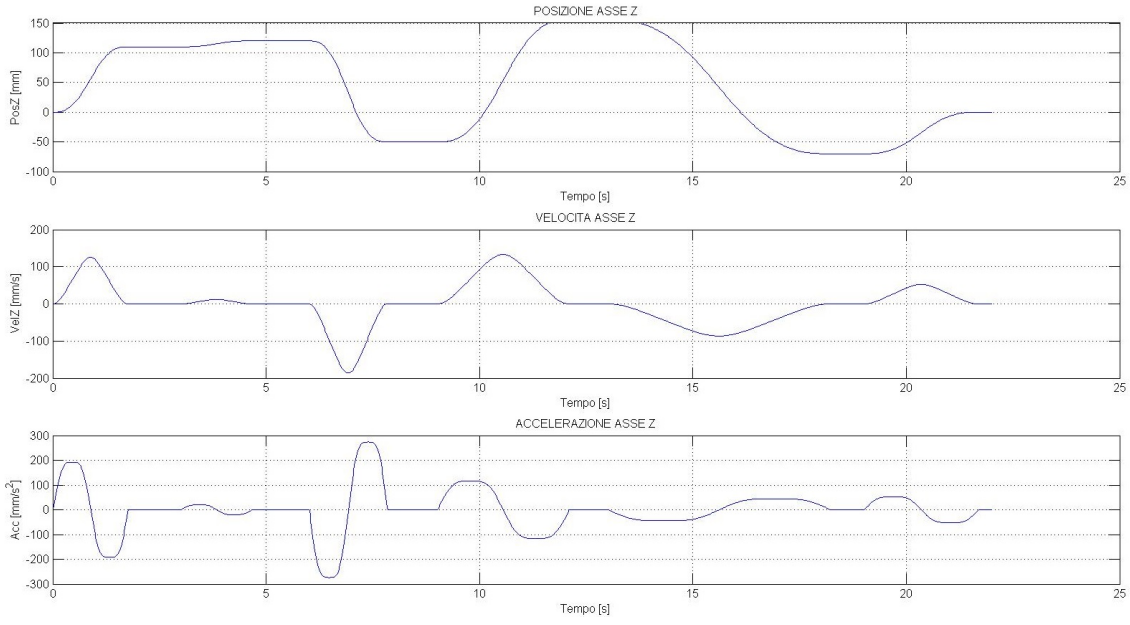

Fig. 5.23: Legge Freudestein 1-3: posizione, velocità ed accelerazione dell'asse Z

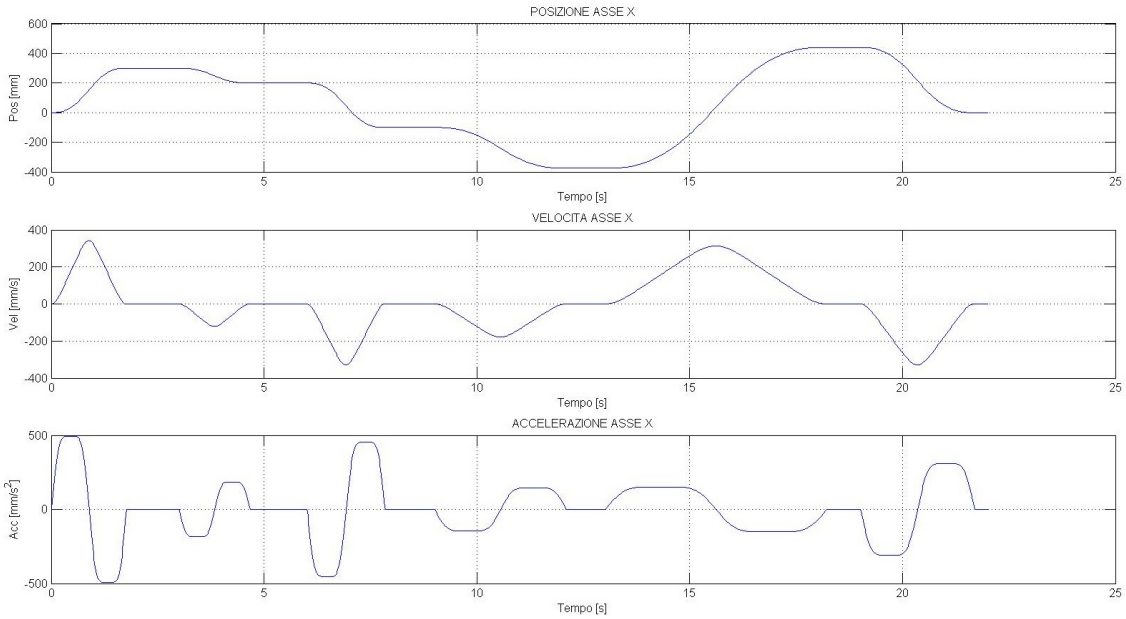

Fig. 5.24: Legge Freudestein 1-3-5: posizione, velocità ed accelerazione dell'asse X

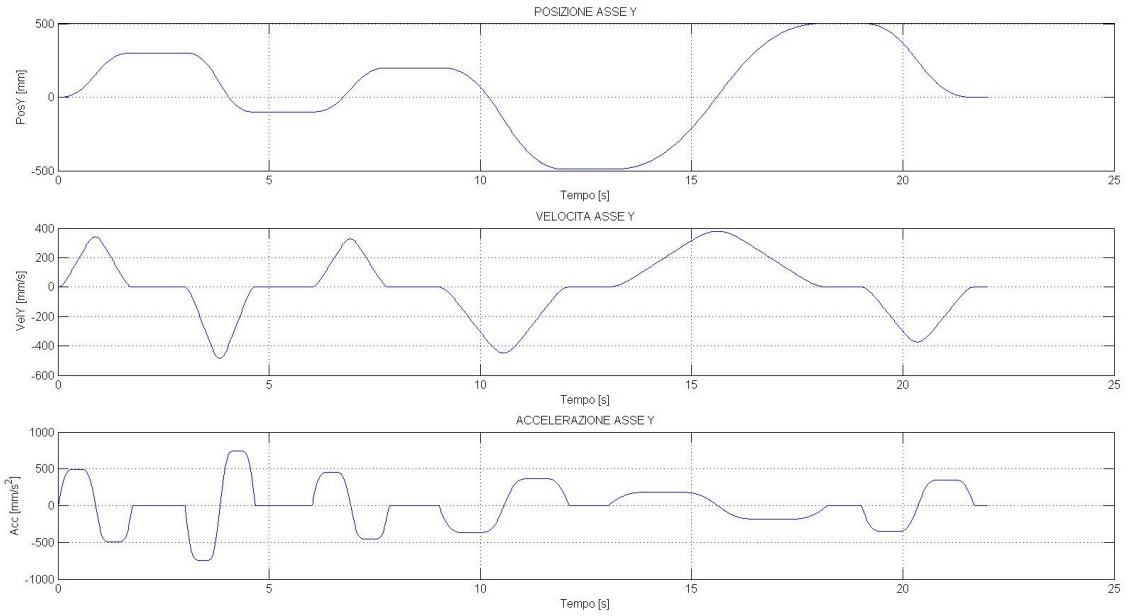

Fig. 5.25: Legge Freudestein 1-3-5: posizione, velocità ed accelerazione dell'asse Y

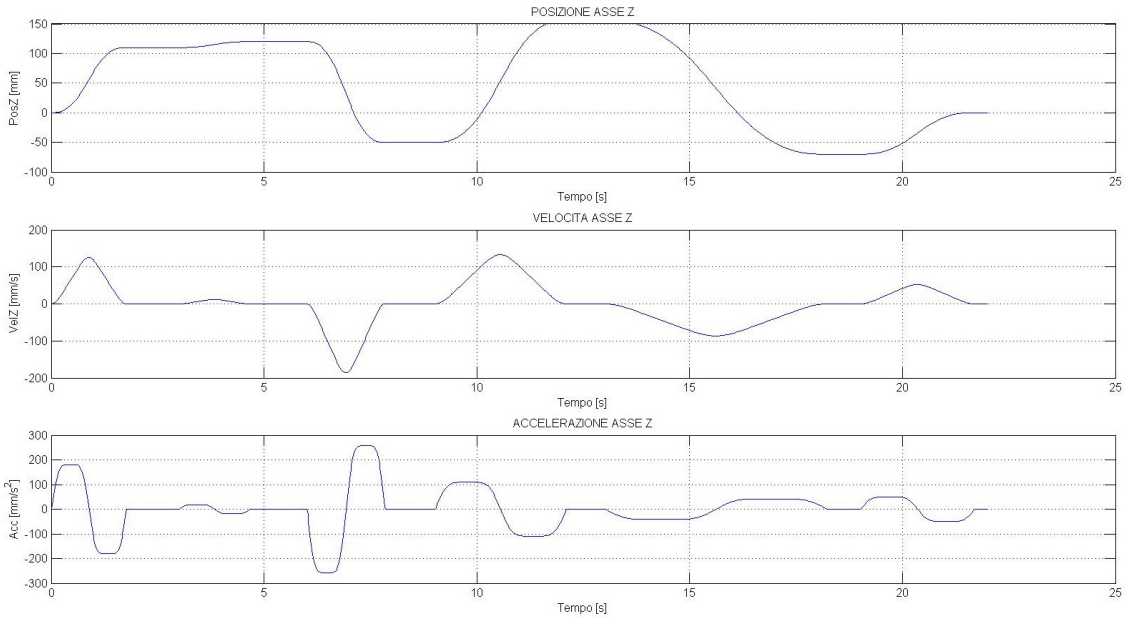

Fig. 5.26: Legge Freudestein 1-3-5: posizione, velocità ed accelerazione dell'asse Z

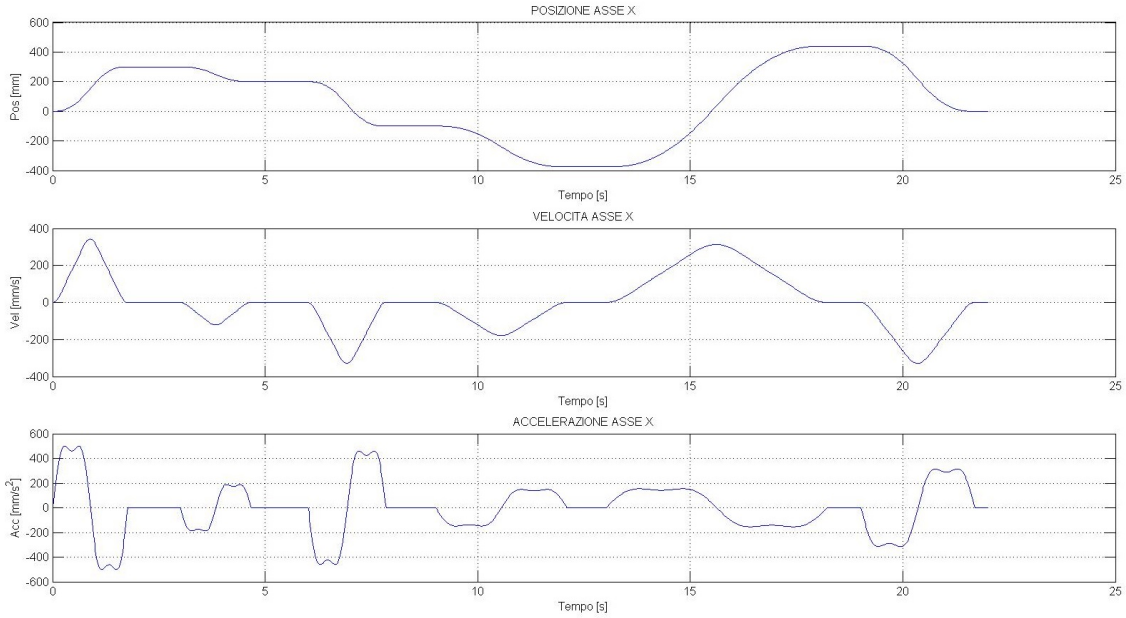

Fig. 5.27: Legge Gutman 1-3: posizione, velocità ed accelerazione dell'asse X

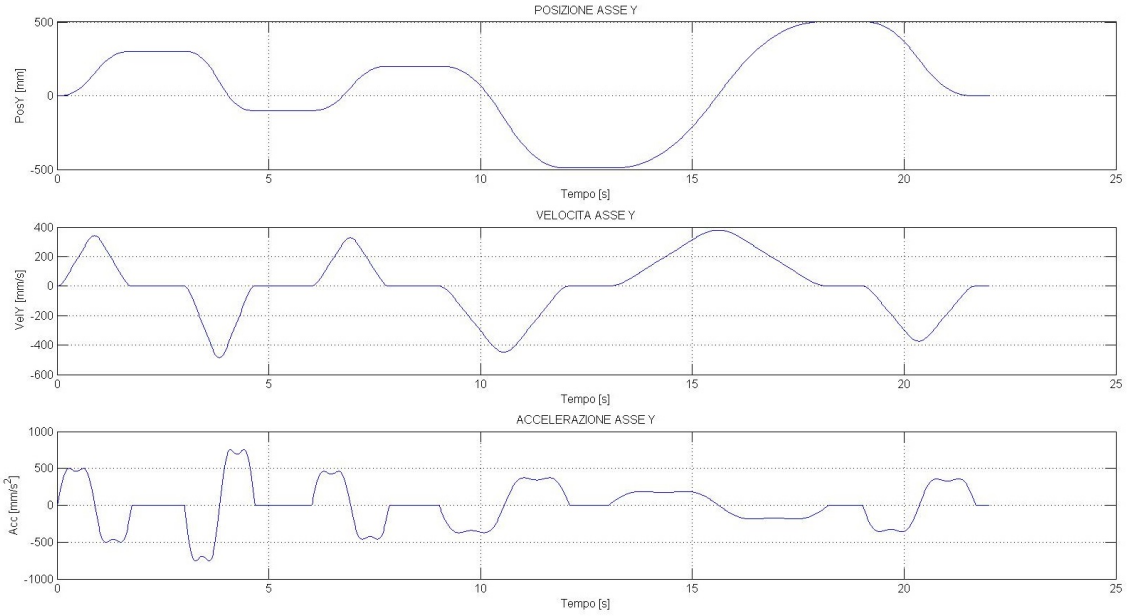

Fig. 5.28: Legge Gutman 1-3: posizione, velocità ed accelerazione dell'asse Y

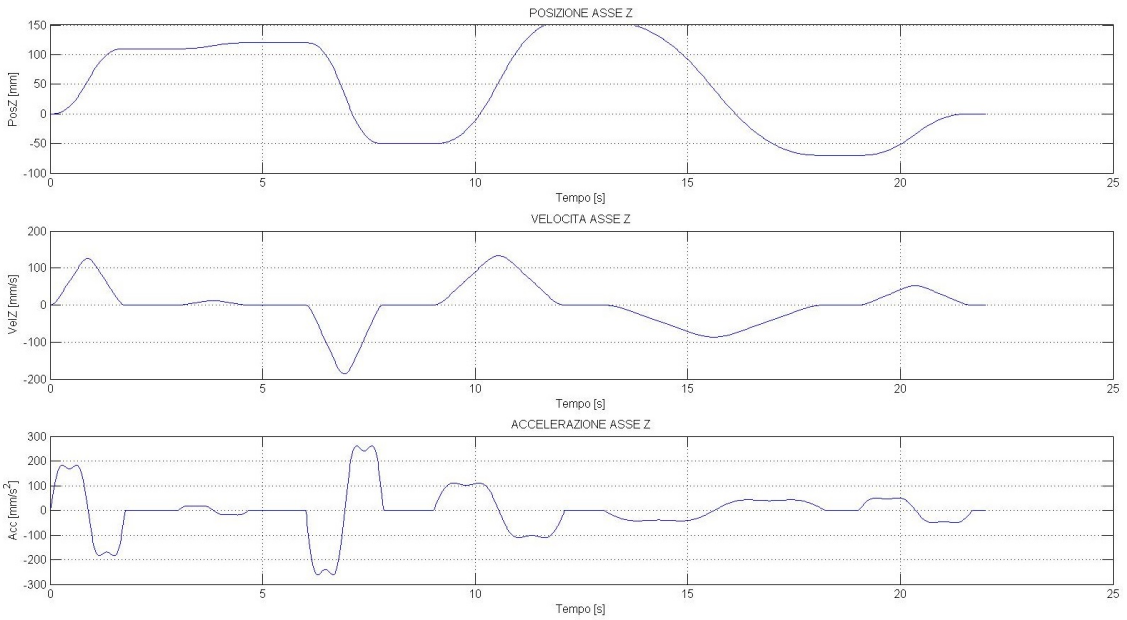

Fig. 5.29: Legge Gutman 1-3: posizione, velocità ed accelerazione dell'asse Z

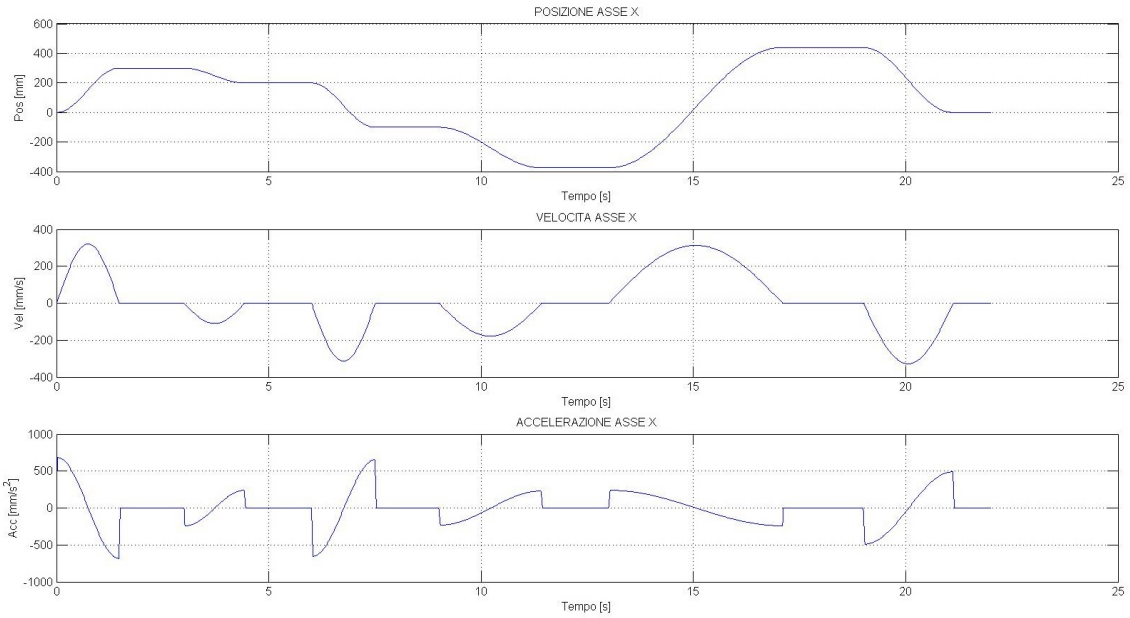

Fig. 5.30: Legge Armonica: posizione, velocità ed accelerazione dell'asse X

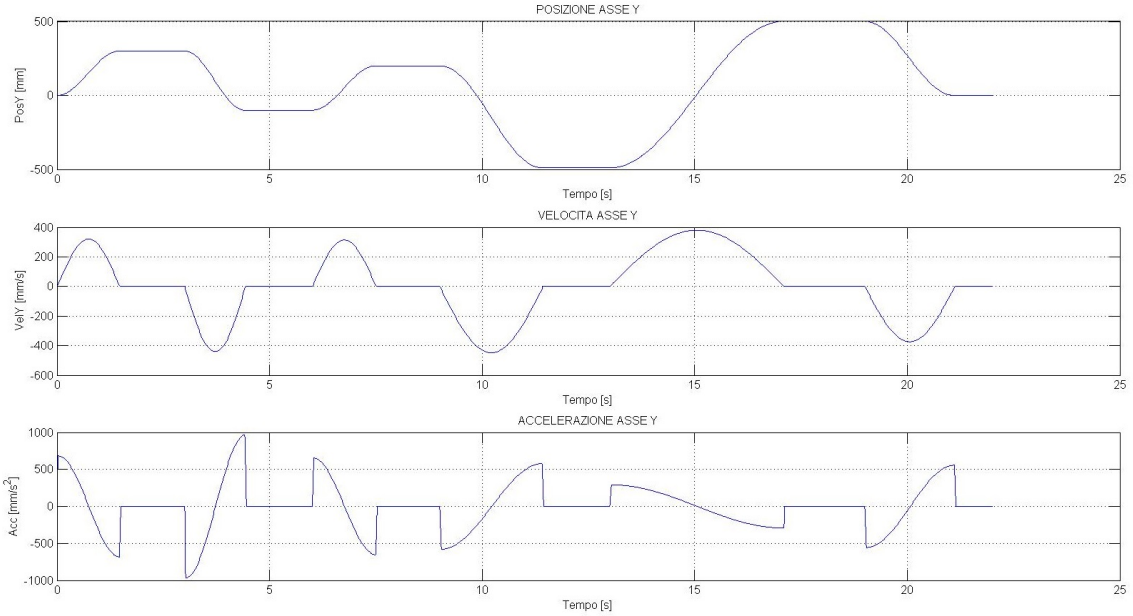

Fig. 5.31: Legge Armonica: posizione, velocità ed accelerazione dell'asse Y

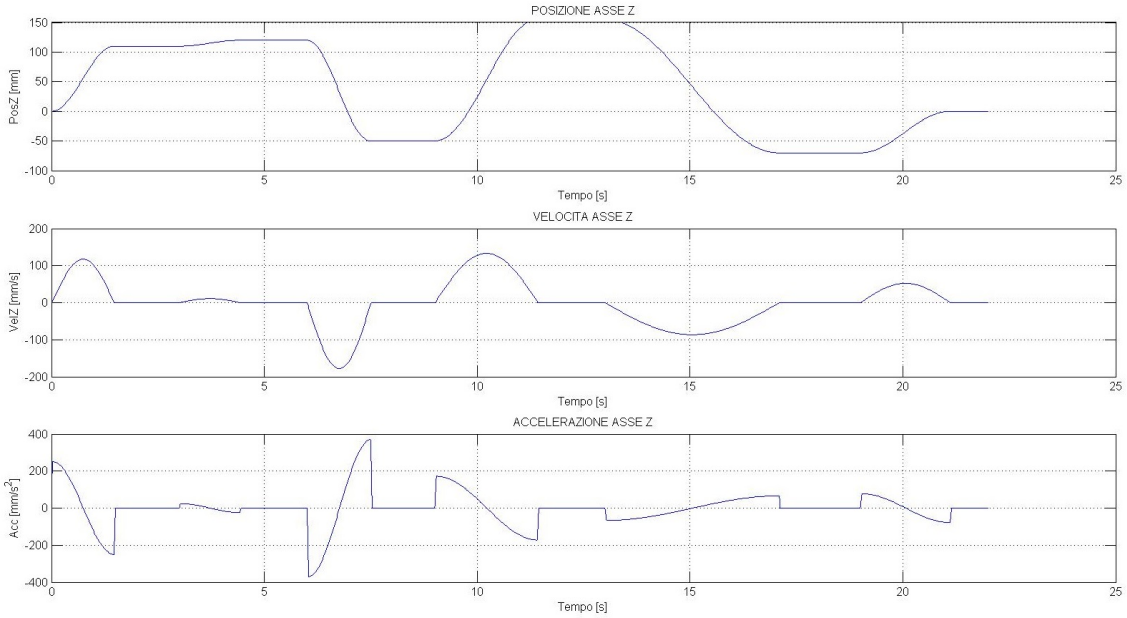

Fig. 5.32: Legge Armonica: posizione, velocità ed accelerazione dell'asse Z

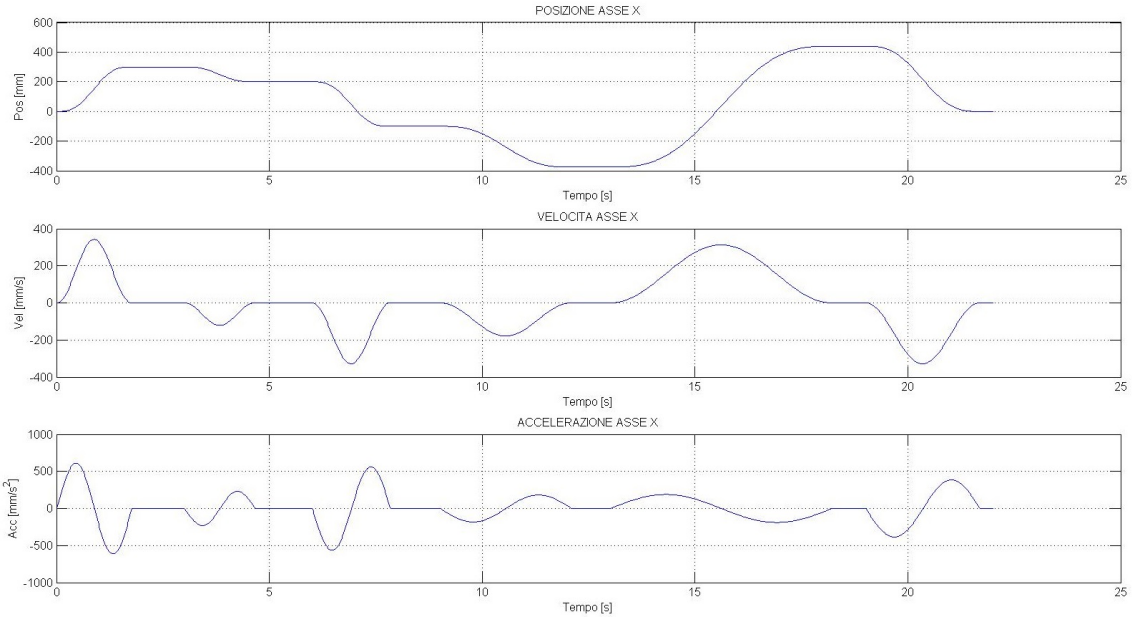

Fig. 5.33: Legge Cicloidale: posizione, velocità ed accelerazione dell'asse X

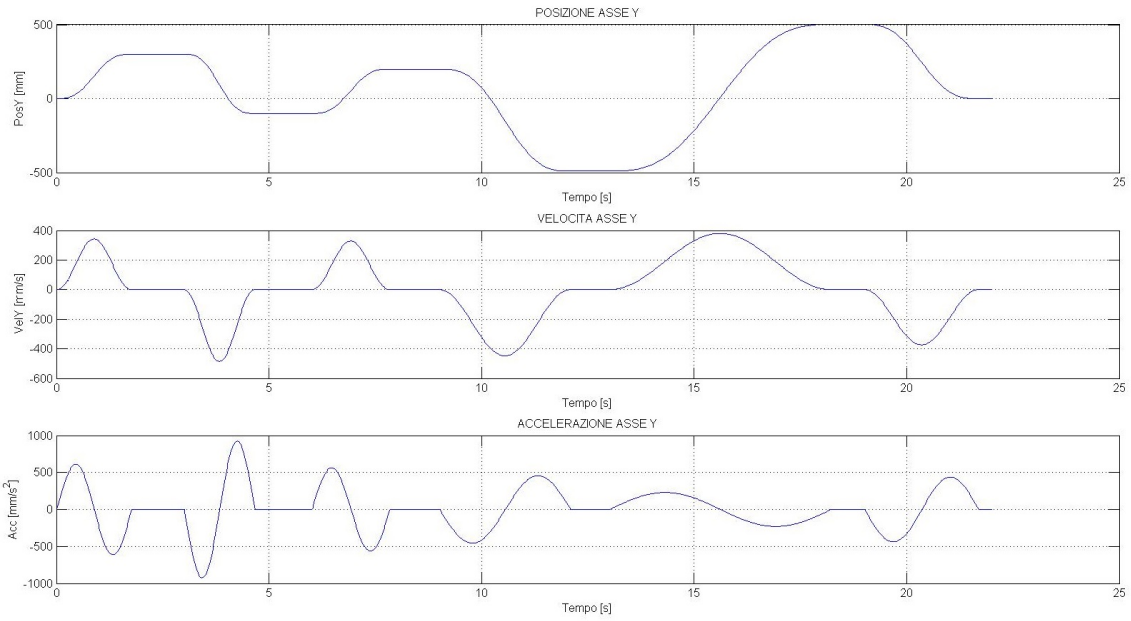

Fig. 5.34: Legge Cicloidale: posizione, velocità ed accelerazione dell'asse Y

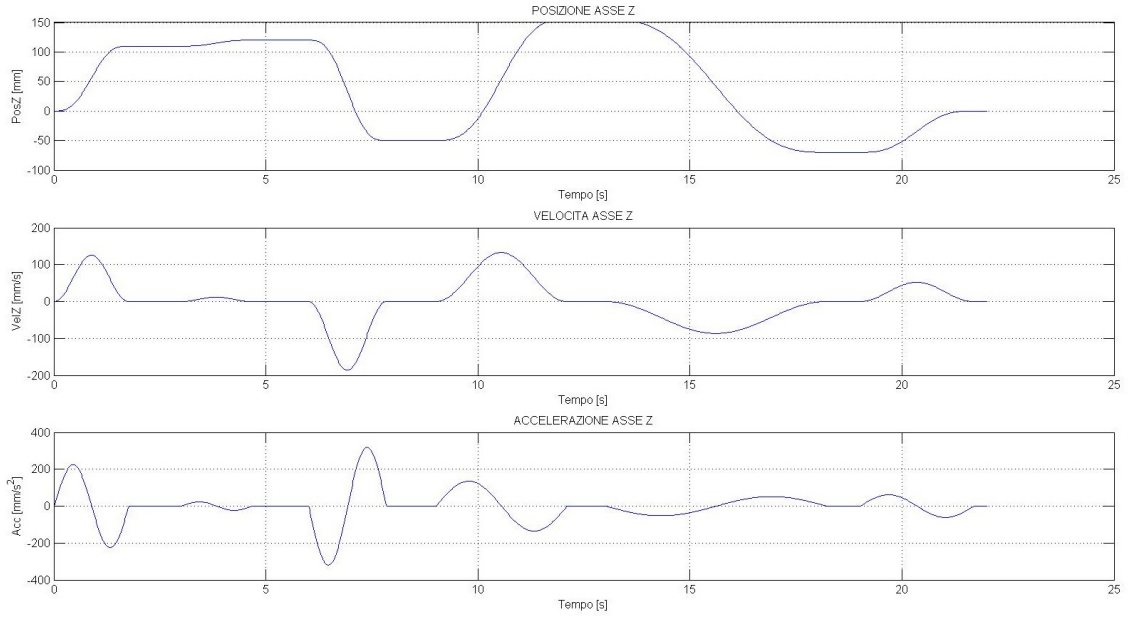

Fig. 5.35: Legge Cicloidale: posizione, velocità ed accelerazione dell'asse Z

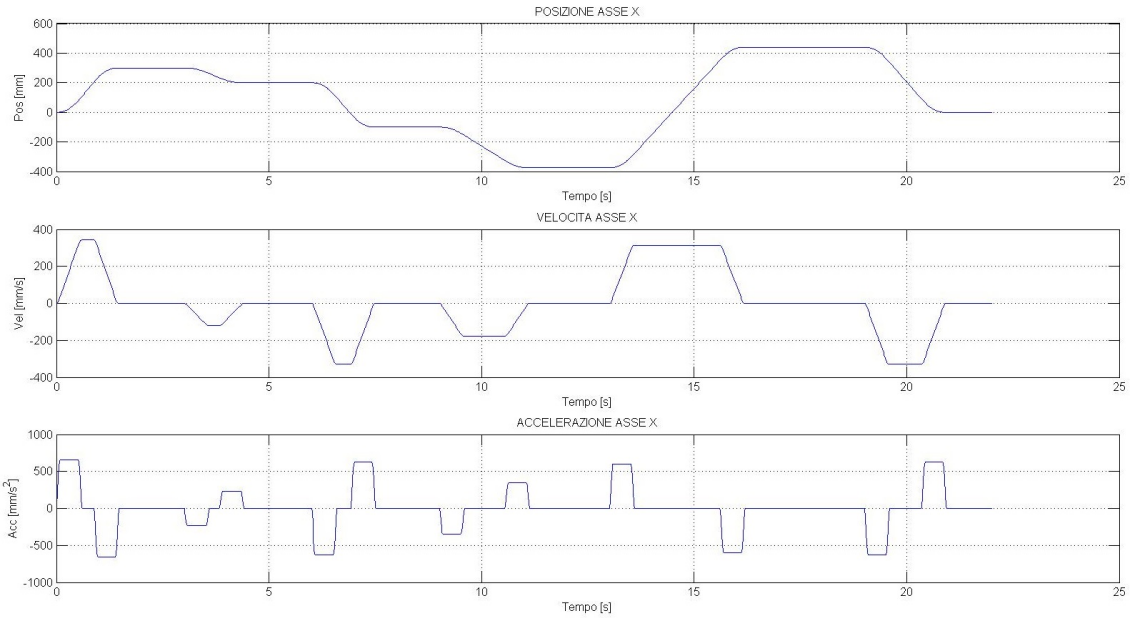

Fig. 5.36: Legge Trapezoidale in Accelerazione: posizione, velocità ed accelerazione dell'asse X

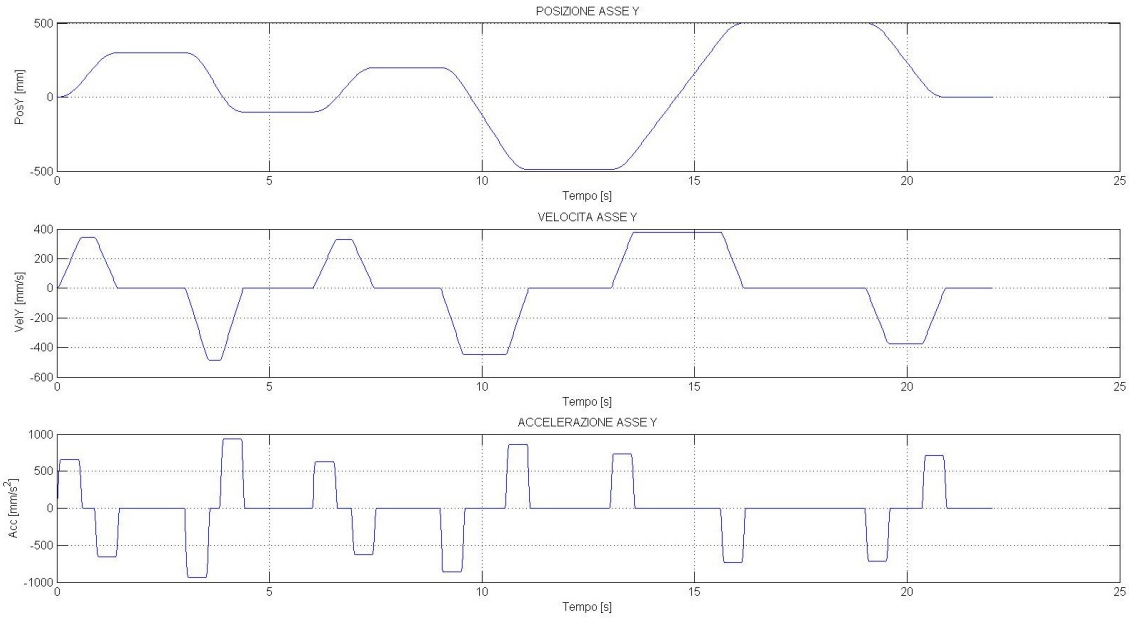

Fig. 5.37: Legge Trapezoidale in Accelerazione: posizione, velocità ed accelerazione dell'asse Y

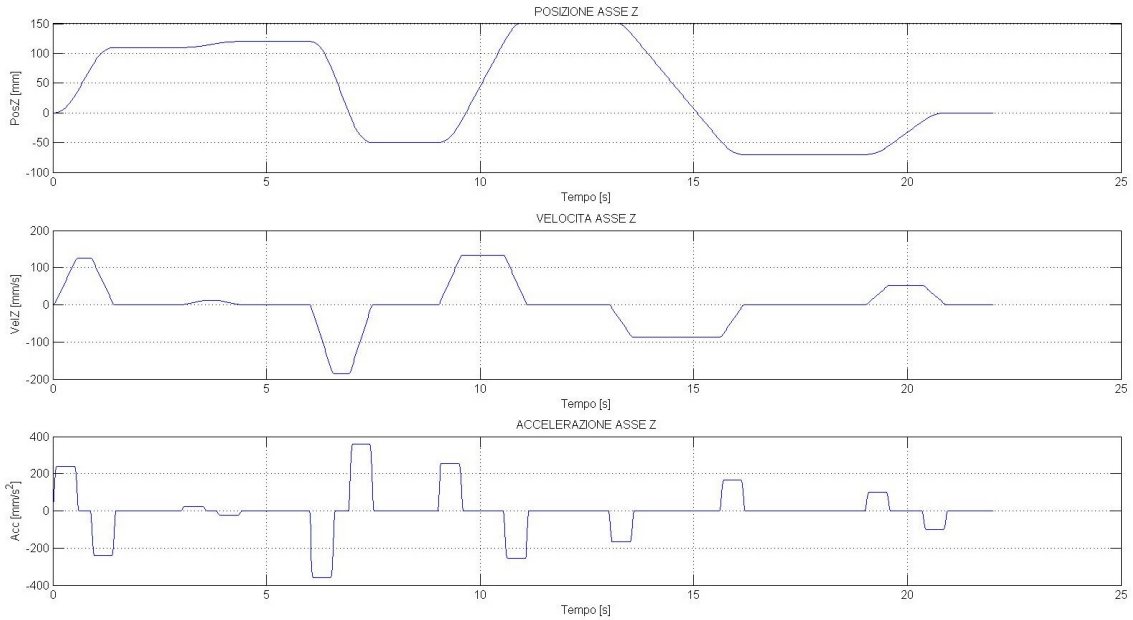

Fig. 5.38: Legge Trapezoidale in Accelerazione: posizione, velocità ed accelerazione dell'asse Z

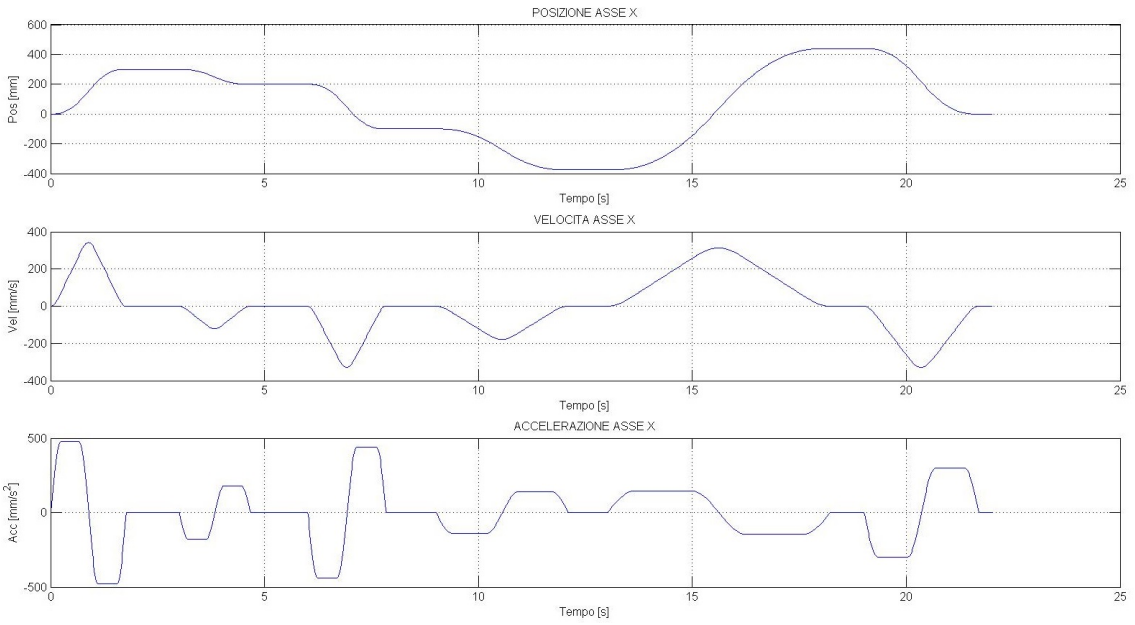

Fig. 5.39: Legge Trapezoidale Modificata: posizione, velocità ed accelerazione dell'asse X

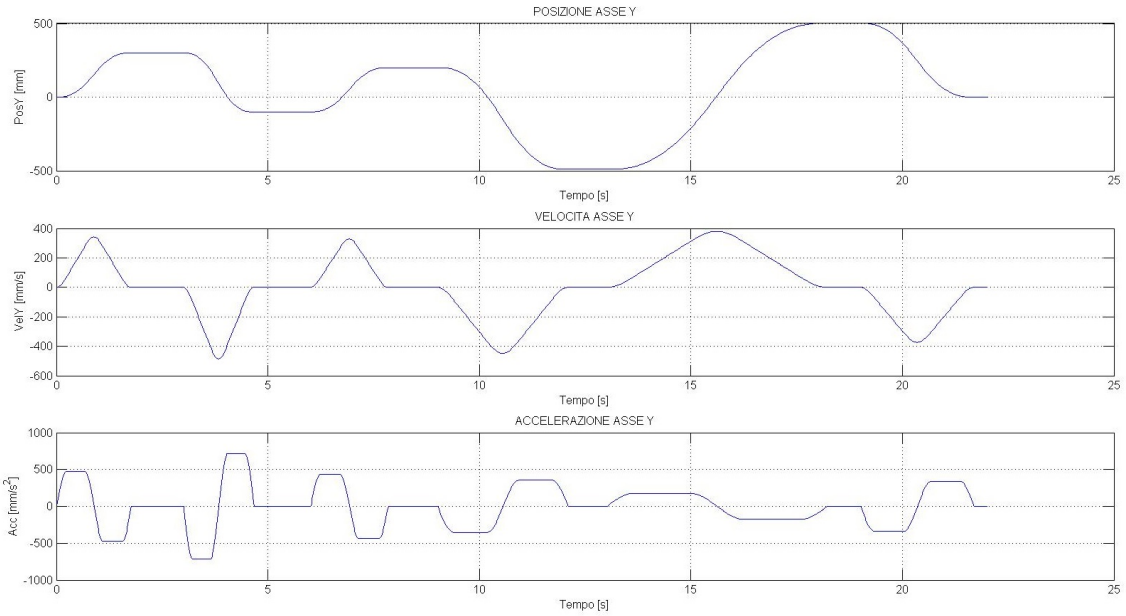

Fig. 5.40: Legge Trapezoidale Modificata: posizione, velocità ed accelerazione dell'asse Y

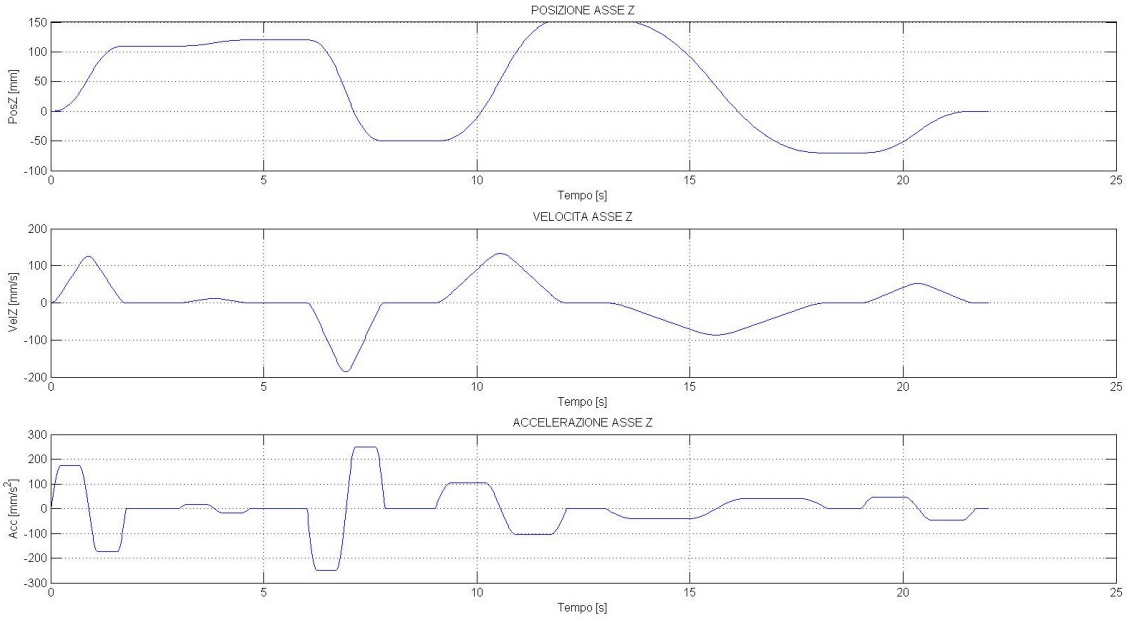

Fig. 5.41: Legge Trapezoidale Modificata: posizione, velocità ed accelerazione dell'asse Z

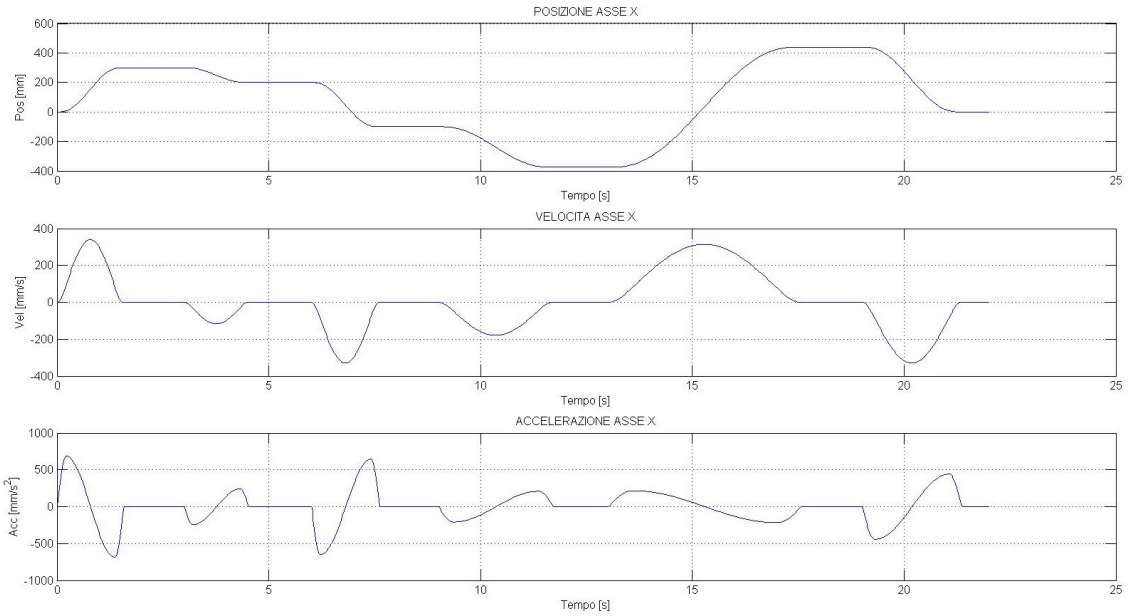

Fig. 5.42: Legge Sinusoidale Modificata: posizione, velocità ed accelerazione dell'asse X

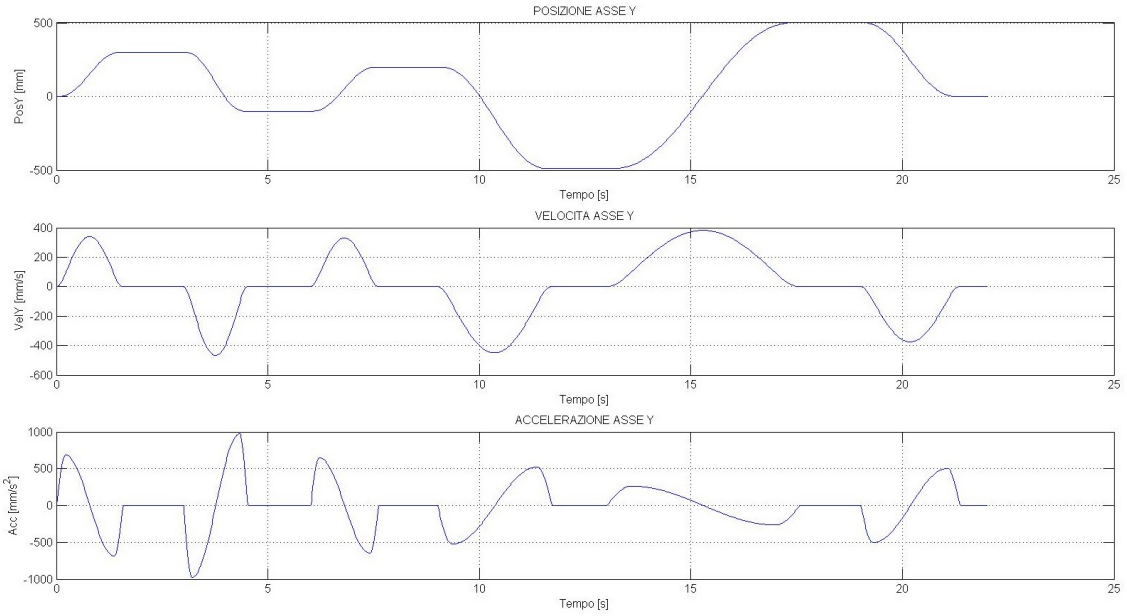

Fig. 5.43: Legge Sinusoidale Modificata: posizione, velocità ed accelerazione dell'asse Y

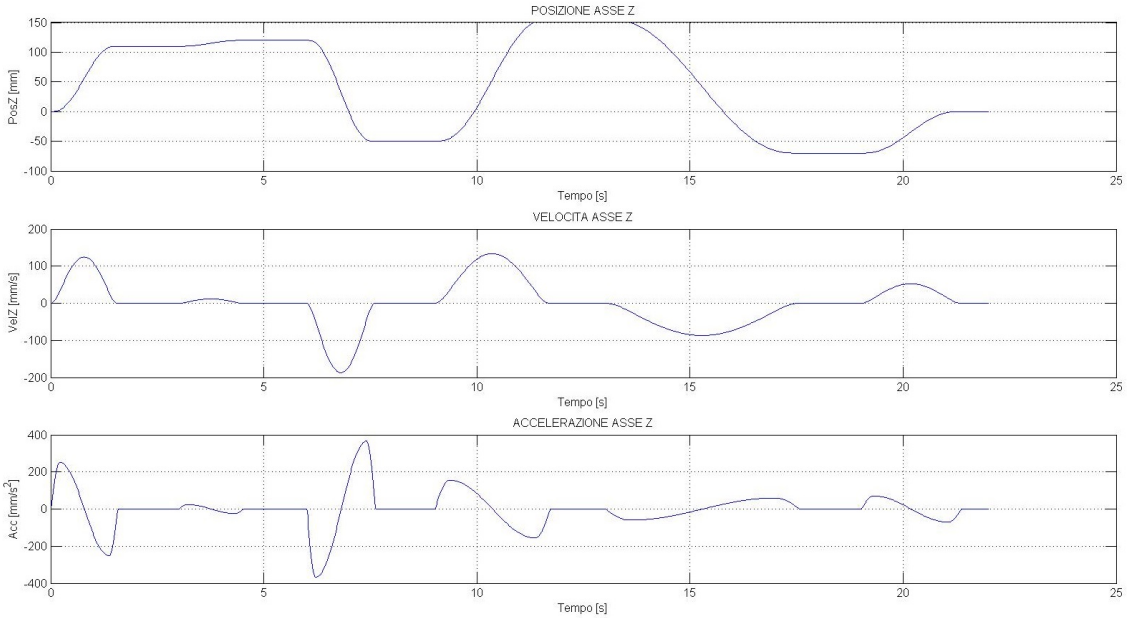

Fig. 5.44: Legge Sinusoidale Modificata: posizione, velocità ed accelerazione dell'asse Z

5.7 Analisi dei Risultati

Una volta che il blocco funzionale ha calcolato la traiettoria desiderata con i relativi limiti di velocità e accelerazione impostati, essa viene campionata ed inviata ad intervalli regolari di 16ms al controllore Adept SmartController, il quale farà alterare la posizione istantanea dell'organo terminale in base allo step calcolato. Il tempo che intercorre tra l'invio del riferimento di posizione da parte dell'xPC e il completamento effettivo dell'alterazione purtroppo non è nullo, ma assume un valore costante in base alle caratteristiche dinamiche del manipolatore e di quelle prestazionali della filiera di trasmissione. Si è potuto constatare che, con il setup sperimentale utilizzato, la somma dei ritardi dovuti alla trasmissione via rete dei dati, la loro elaborazione da parte del controllore e la successiva movimentazione dell'organo terminale porta ad avere un ritardo tra l'istante di invio del riferimento di posizione e il suo effettivo raggiungimento da parte del robot di circa 80ms (figura 5.45). Si fa presente che la caratteristica più limitante per il problema dei ritardi è la banda passante del manipolatore stesso, valutata all'incirca sui 64ms.

La precisione di movimentazione è in ogni caso molto buona, ed è stata fatta inoltre una valutazione sulla bontà del calcolo della posizione dell'organo terminale tramite il comando *HERE* integrato nell'ambiente Adept Desktop. Tale verifica è stata realizzata confrontando la posizione restituita dal comando *HERE* e quella ritornata dal blocco del calcolo della cinematica diretta di posizione implementato offline sull'xPC. Quest'ultimo è sicuramente un metodo più preciso in quanto necessita solamente dei valori letti dagli encoder del manipolatore calcolando la posizione dell'end-effector in forma chiusa, la quale risulta perciò esatta a meno degli errori iniziali sugli encoder. Ne sono risultati grafici estremamente simili, visibili in figura 5.46,i quali differiscono per quantità così esigue da poter considerare i due metodi praticamente equivalenti dal punti di vista qualitativo, ma preferendo alla fine quello offline in quanto riduce sensibilmente il carico di lavoro sul controllore. Si fa tener presente che qualora si lavorasse anche con le rotazioni il procedimento offline non è utilizzabile per il calcolo della posizione angolare con il metodo in forma chiusa, obbligando ad una soluzione iterativa per approssimazioni successive,

riducendo per cui il vantaggio dell'utilizzo del calcolo offline, preferendo la funzione $HERE$ nettamente meno complicata.

Vengono riportati di seguito i grafici relativi alle prove di movimentazione effettuate con i riferimenti calcolati al paragrafo precedente.

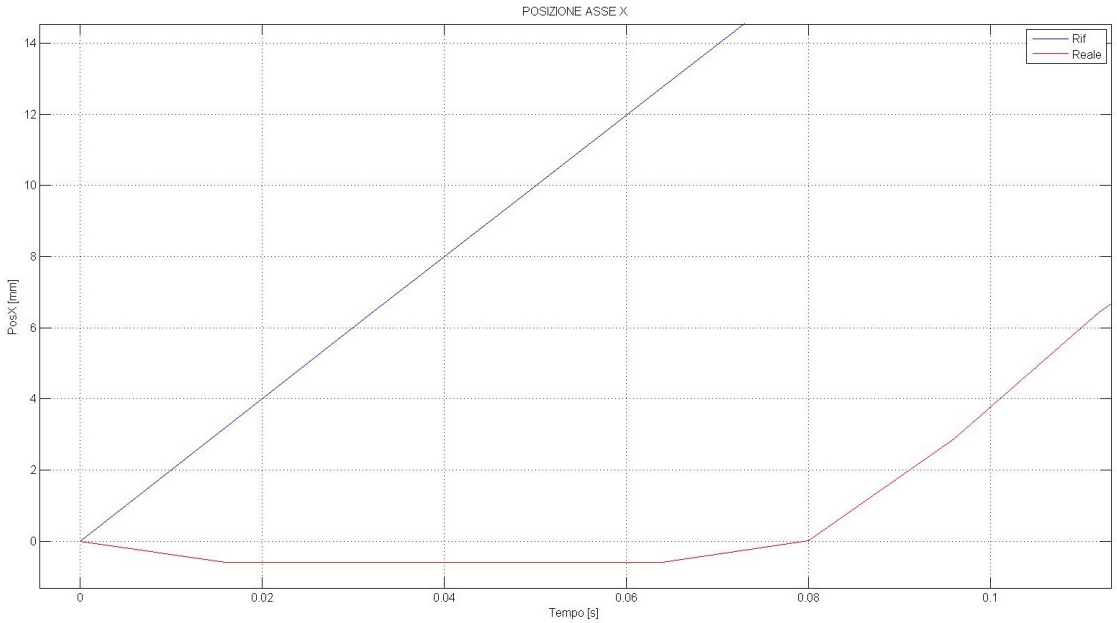

Fig. 5.45: Ritardo di movimentazione osservato tra il riferimento e la posizione reale

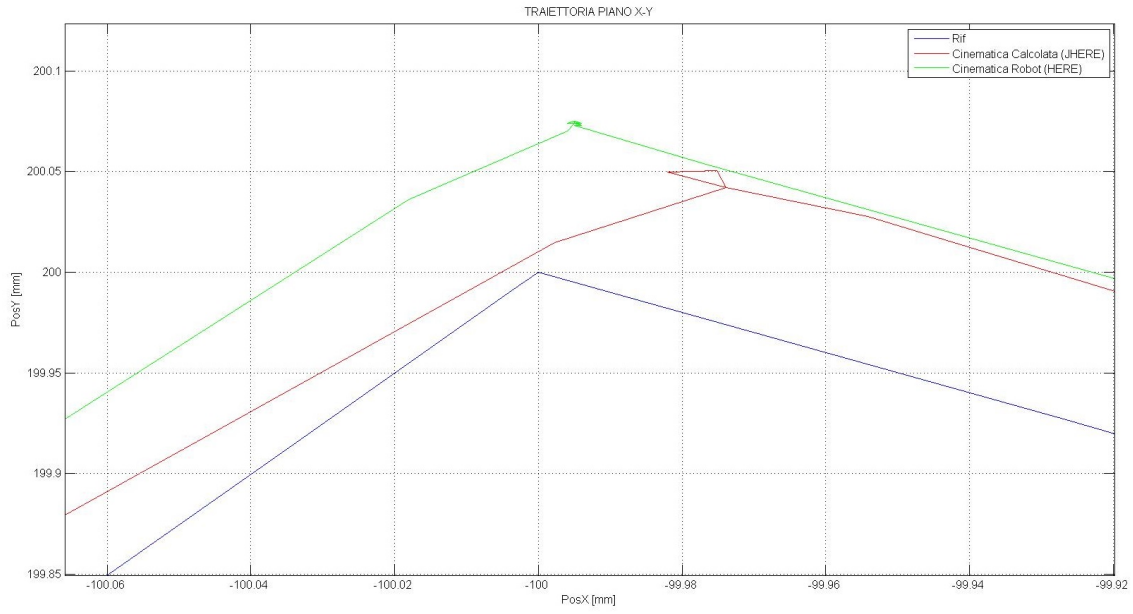

Fig. 5.46: Differenza tra il riferimento di posizione, la posizione effettiva restituita dal comando $HERE$ e quella calcolata offline

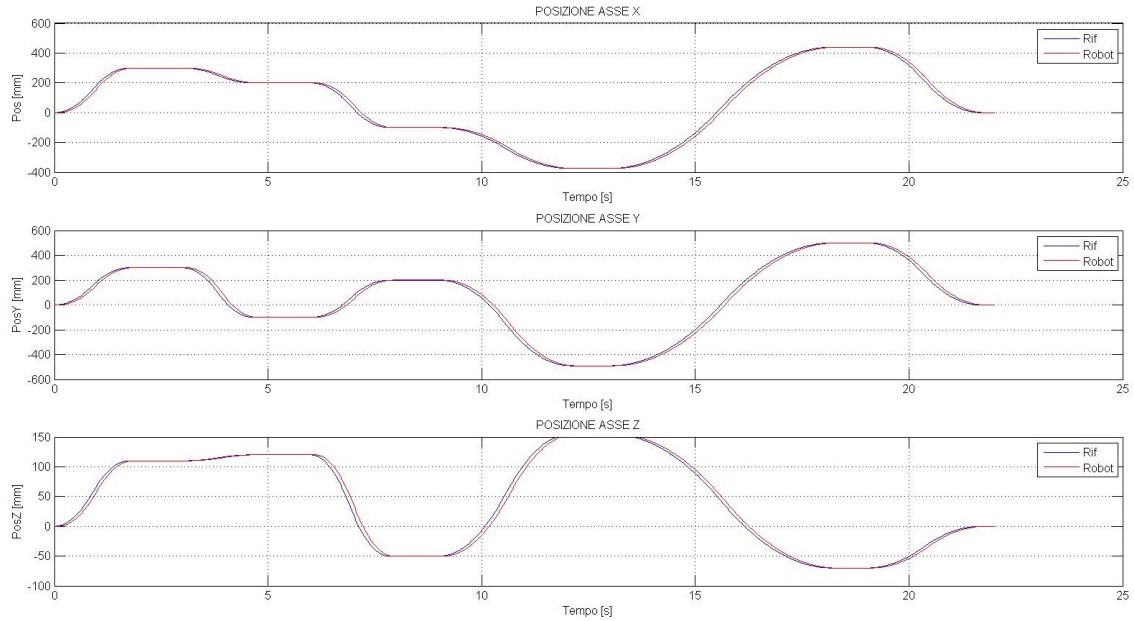

Fig. 5.47: Legge Triangolare: confronto tra il riferimento e la posizione reale

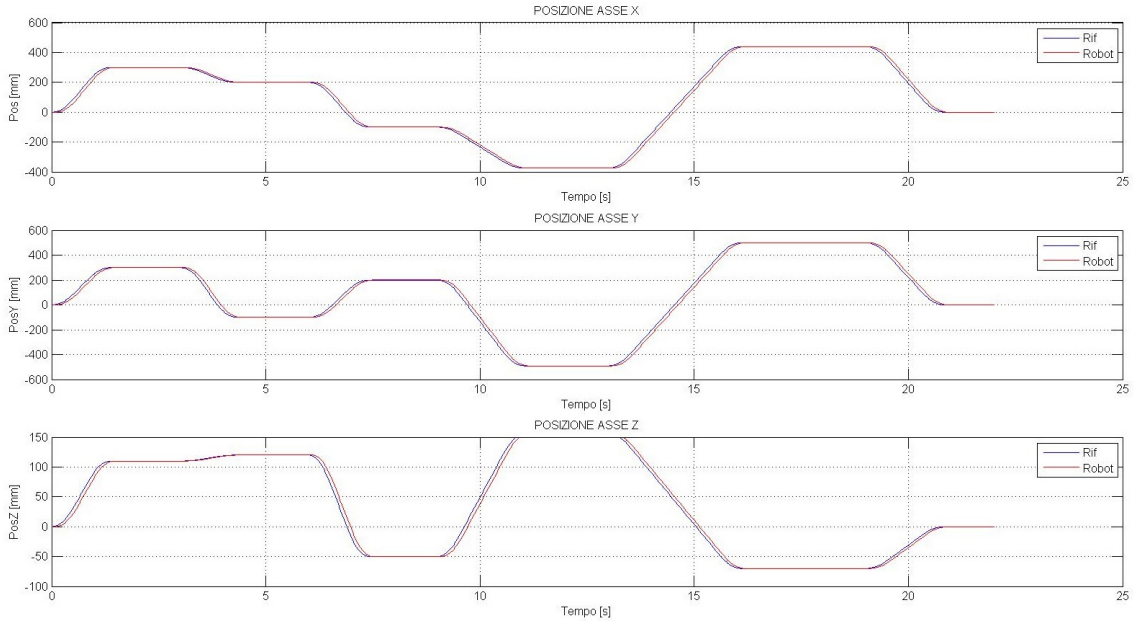

Fig. 5.48: Legge Trapezoidale in Velocità: confronto tra il riferimento e la posizione reale

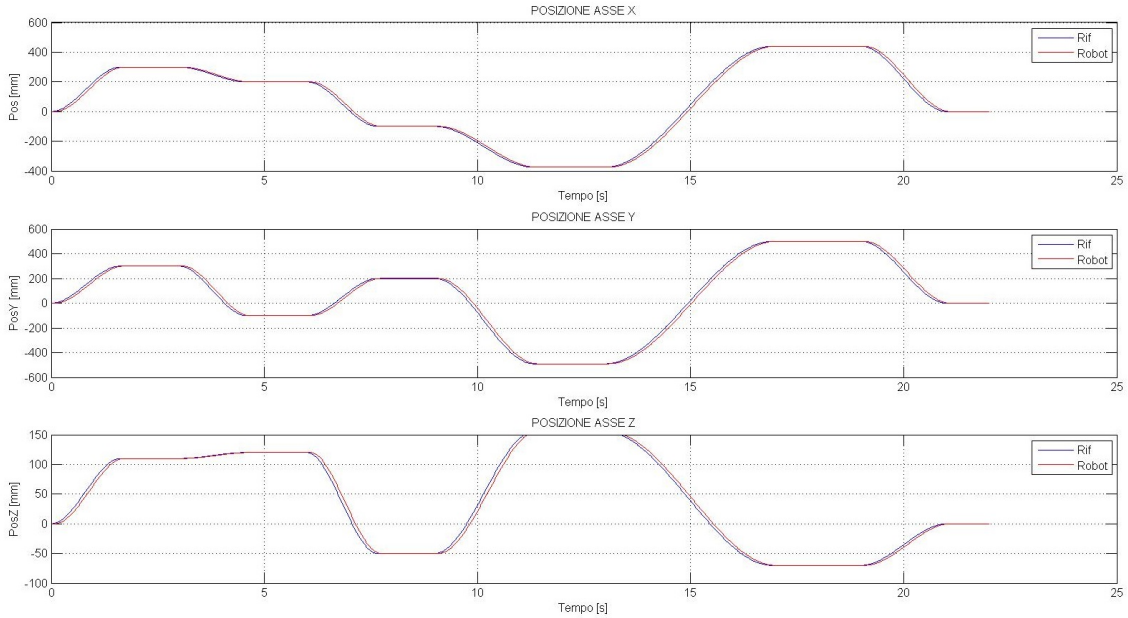

Fig. 5.49: Legge Polinomiale 3° grado: confronto tra il riferimento e la posizione reale

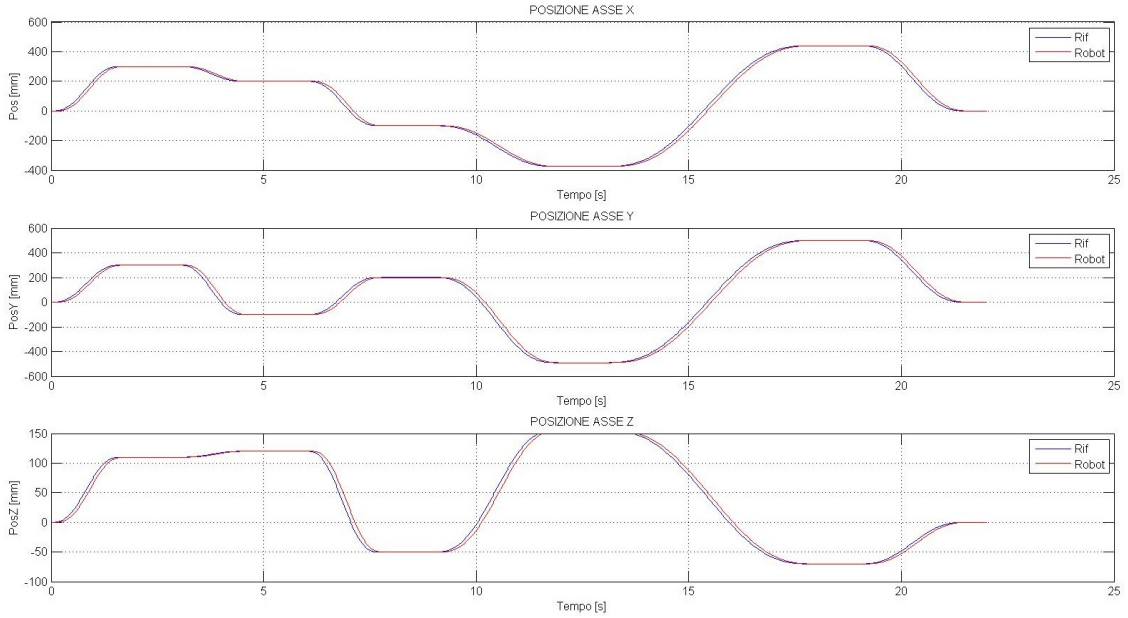

Fig. 5.50: Legge Polinomiale 5° grado: confronto tra il riferimento e la posizione reale

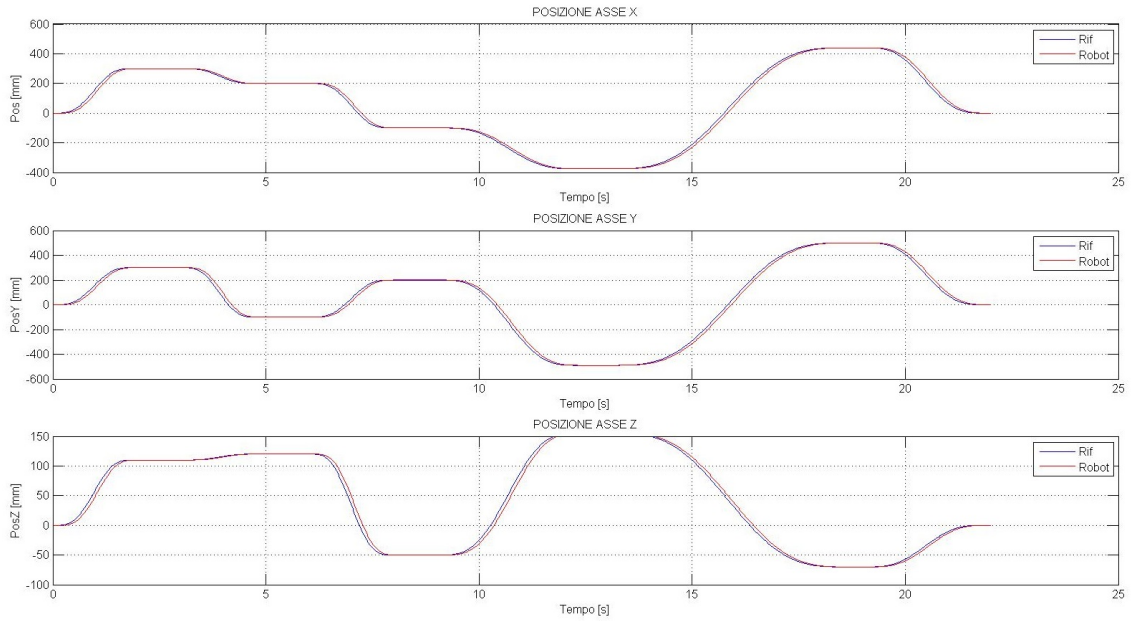

Fig. 5.51: Legge Polinomiale 7° grado: confronto tra il riferimento e la posizione reale

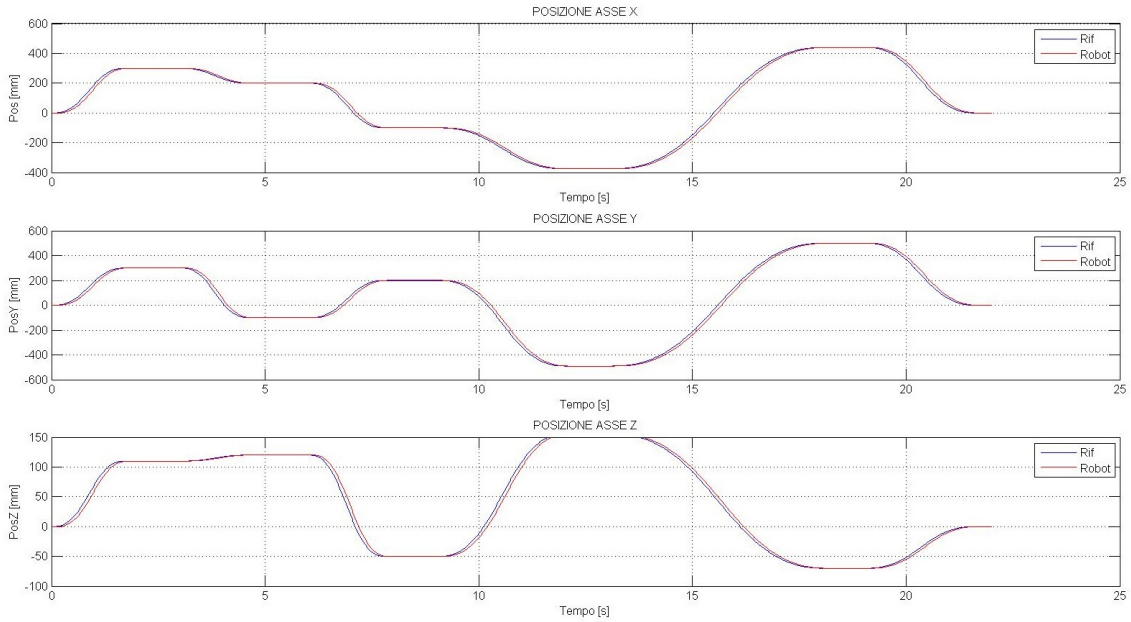

Fig. 5.52: Legge Freudestein 1-3: confronto tra il riferimento e la posizione reale

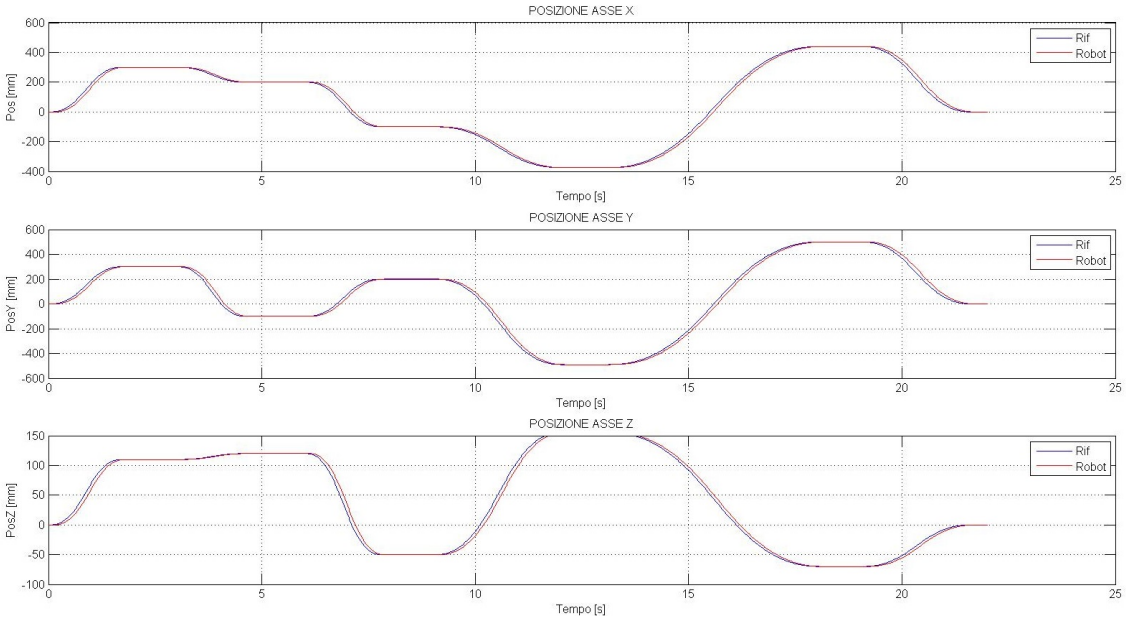

Fig. 5.53: Legge Freudestein 1-3-5: confronto tra il riferimento e la posizione reale

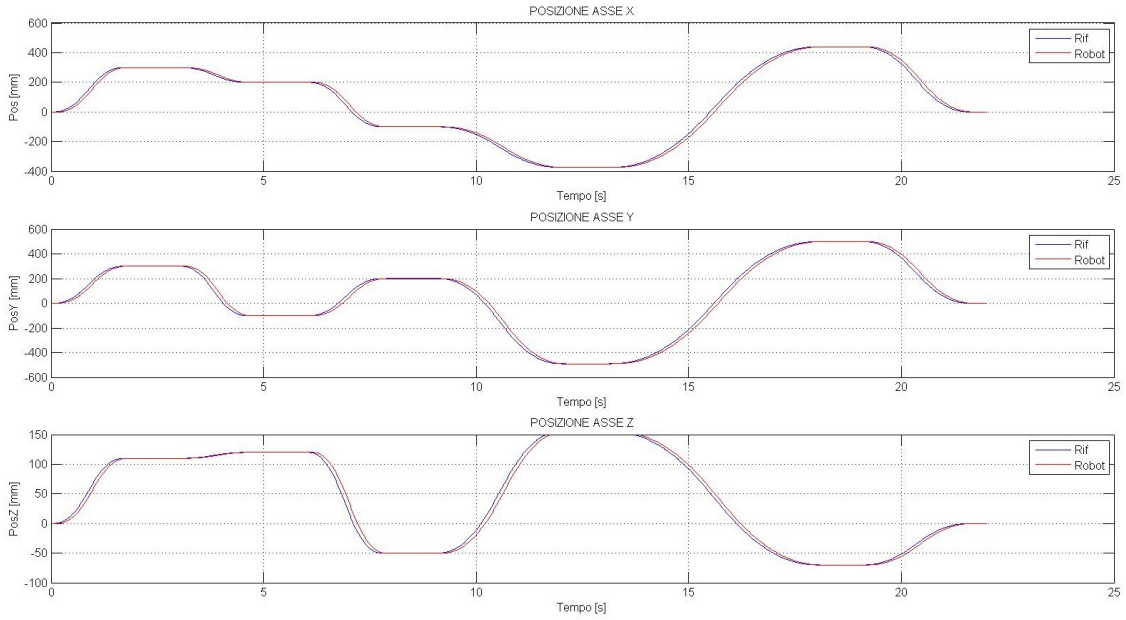

Fig. 5.54: Legge Gutman 1-3: confronto tra il riferimento e la posizione reale

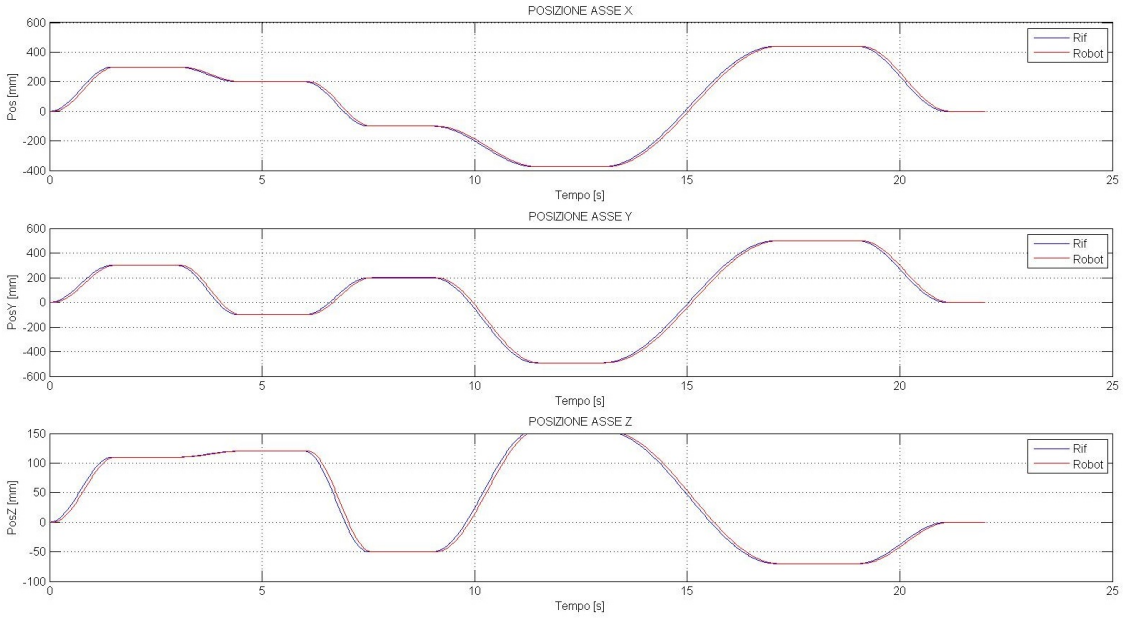

Fig. 5.55: Legge Armonica: confronto tra il riferimento e la posizione reale

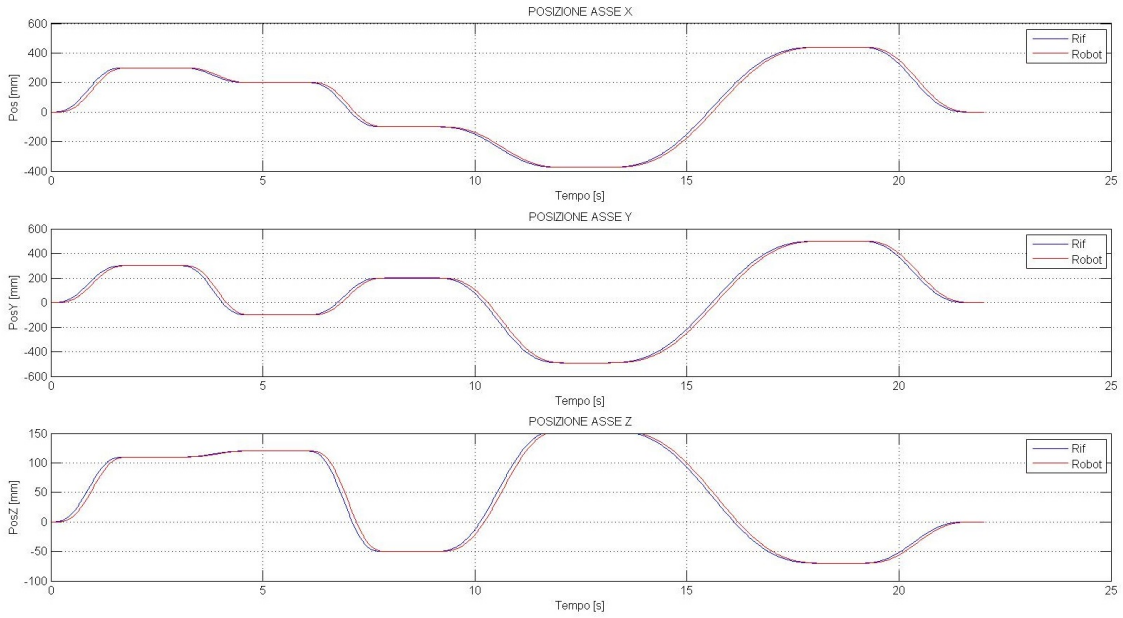

Fig. 5.56: Legge Cicloidale: confronto tra il riferimento e la posizione reale

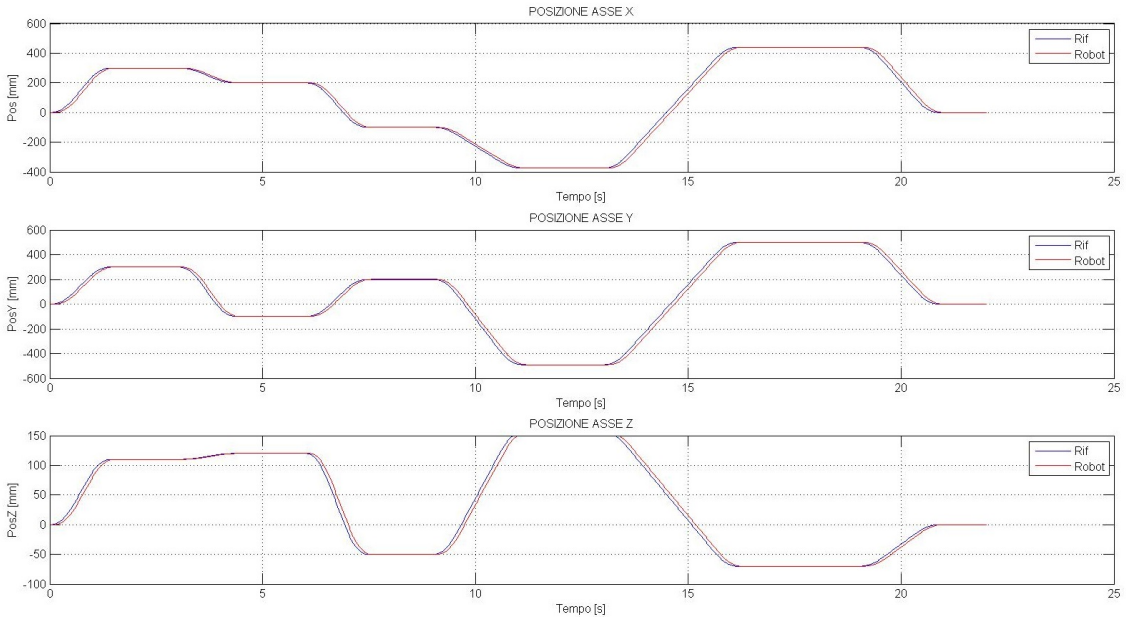

Fig. 5.57: Legge Trapezoidale in Accelerazione: confronto tra il riferimento e la posizione reale

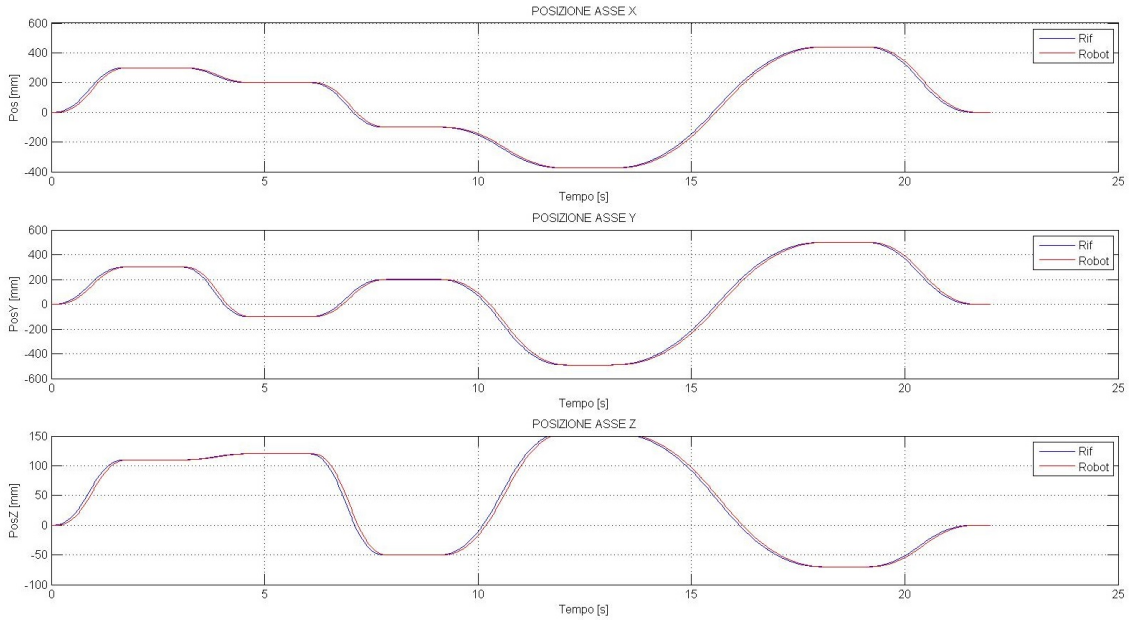

Fig. 5.58: Legge Trapezoidale Modificata: confronto tra il riferimento e la posizione reale

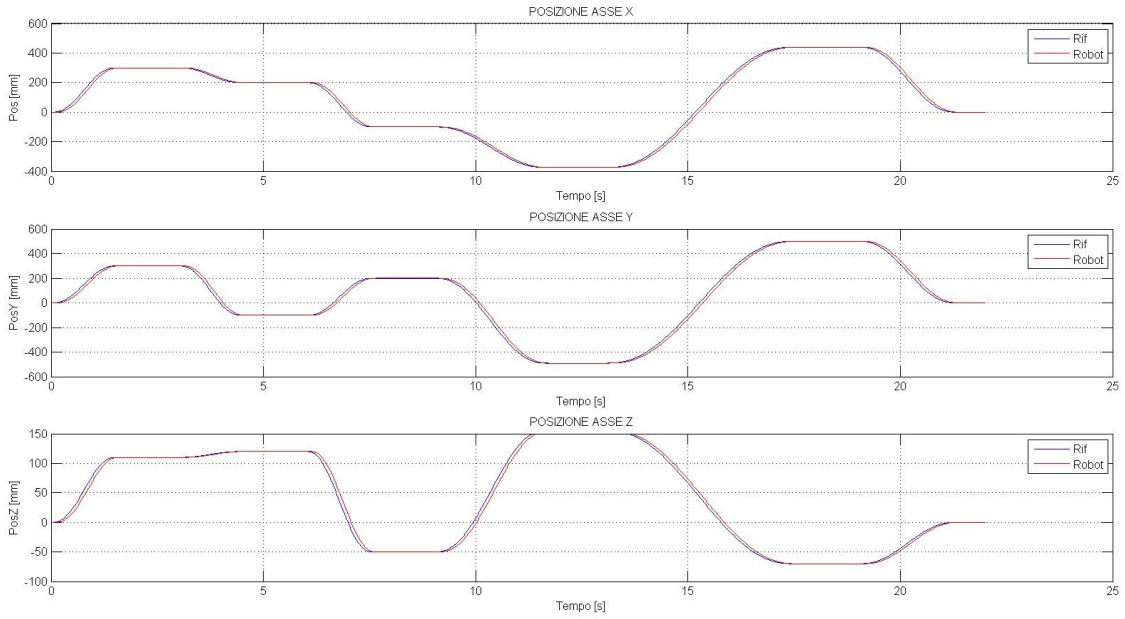

Fig. 5.59: Legge Sinusoidale Modificata: confronto tra il riferimento e la posizione reale

Conclusioni

Il lavoro presentato in tale elaborato ha avuto l'obiettivo di sviluppare delle strategie di controllo innovative utilizzabili sui manipolatori paralleli dell'azienda Adept. Lo scopo è stato quello di sfruttare dei particolari comandi, messi a disposizione dall'ambiente di sviluppo Adept Desktop ed implementabili sul controllore Adept SmartController, che han permesso di modificare il comportamento classico dei comandi standard generalmente utilizzati per la movimentazione del manipolatore parallelo Adept, rendendo possibile una gestione del moto più accurato in quei casi dove la componente real-time del controllo è quella preponderante.

La prima parte del lavoro, concentrata sullo sviluppo del codice V+ che viene eseguito direttamente sul controllore dedicato, è stata la fase più delicata. La ricerca della corretta sintassi dei comandi, la maggior parte priva di grosse documentazioni a riguardo, ha reso necessario un continuo lavoro di sperimentazione pratica sul manipolatore stesso, in modo da verificare le molteplici possibilità di configurazione e di sincronizzazione del controllore. Grosso sforzo ha richiesto proprio la fase di sincronia, affinata testando diverse combinazioni dei parametri interni del controllore al fine di ottenere efficienza nella gestione del moto garantendo al tempo stesso una fluidità e precisione notevole.

Una volta trovato il miglior compromesso e verificato il software di base realizzato, ci si è concentrati sullo sviluppo del programma al livello superiore per la gestione vera e propria dei riferimenti di posizione. Esso è stato realizzato in ambiente Matlab Simulink ed eseguito dall'xPC target della National Instruments. Anche in questo caso, la ricerca della corretta sincronizzazione tra il computer real-time e il controllore non è stata di rapida risoluzione, ed ha richiesto diverso tempo per la sua messa a punto. In particolare si è dovuto prestare attenzione alla sincronia con cui i dati di riferimento, campionati ad una ben determinata frequenza dall'xPC, dovevano venire inviati al controllore in modo da ottenere delle alterazioni del manipolatore sincronizzate perfettamente, evitando possibili perdite di pacchetti indispensabili per la continuità del moto.

Interfacciati correttamente i diversi moduli del controllo ci si è concentrati sullo sviluppo dei due blocchi funzionali addetti al calcolo della cinematica diretta di posizione del manipolatore ed al calcolo in tempo reale della traiettoria da far realizzare all'organo terminale. Questa è stata la fase più teorica dell'esperienza, caratterizzata dall'implementazione di algoritmi più o meno semplici ma nei quali si è dovuta prestare sempre una particolare attenzione, al fine di evitare comportamenti dannosi per il manipolatore in seguito a piccoli errori di programmazione.

Sicuramente la parte più appagante è stata quella finale di verifica. In essa sono state testate tutte le leggi di moto implementate, analizzando il comportamento reale del manipolatore. In particolare ci si è concentrati sulla capacità dell'organo terminale di seguire i profili imposti, verificando la correttezza e bontà del moto sia dal punto di vista della precisione spaziale sia di quella puramente temporale. Essendo un controllo realizzato principalmente per realtà real-time, i ritardi con cui il manipolatore esegue le operazioni richieste è un aspetto fondamentale. I risultati ottenuti hanno confermato le aspettative che ci si erano poste all'inizio del lavoro, rispecchiando certi limiti fisici del sistema Adept oltre i quali non si è potuti andare. Si deve ricordare infatti che il manipolatore adoperato è stato progettato per ben determinati scopi industriali, mentre l'obiettivo perseguito durante l'esperienza è stato quello, in qualche modo, di convertire o adattare le caratteristiche di tale robot per scopi non strettamente legati al suo naturale impiego nelle linee di produzione.

Concludendo, il lavoro svolto ha portato allo sviluppo di un controllo innovativo per i manipolatori Adept con ottimi risultati e ampie potenzialità d'impiego. Esso infatti può

trovare impiego in tutti quei settori dove la movimentazione di particolari categorie di carichi deve avvenire nella maniera più dolce possibile, evitando l'instaurarsi di indesiderate oscillazioni pur non tralasciando la rapidità di esecuzione.

Inoltre il software realizzato, essendo per sua natura modulare, può venir agilmente integrato in sistemi di controllo già implementati attraverso l'ambiente Matlab Simulink o anche solamente sfruttandone la parte di codice fondamentale da caricare sul controllore Adept.

Ringraziamenti

Un sentito ringraziamento va al prof. Giovanni Boschetti, che mi ha seguito e consigliato durante le fasi più delicate della tesi sempre con particolare attenzione.
Ringrazio la mia famiglia che mi ha sostenuto in questi anni di studio, tutti gli amici di corso e di paese, i compagni di sciate ed in particolar modo la mia ragazza Eleonora che mi ha supportato e sopportato durante la preparazione degli esami.
Infine un grazie speciale a Fox, inossidabile compagno di studi, per aver reso meno pesanti gli anni universitari con la sua compagnia.

Bibliografia

1. K. L. Lazzari, ≪Sviluppo di un indice dinamico per la previsione del comportamento di manipolatori paralleli≫, *Università degli studi di Padova,Dipartimento di Tecnica e Gestione dei Sistemi Industriali, 2012*. Tesi di laurea Magistrale.

2. A. Auricchio, ≪Controllo non time-based per la movimentazione di liquidi≫, *Università degli studi di Padova,Dipartimento di Tecnica e Gestione dei Sistemi Industriali, 2013*. Tesi di laurea Magistrale.

3. L. Agostini, ≪Kinematic Analysis of a 3-DOF parallel manipulator≫, *Università degli studi di Padova,Dipartimento di Tecnica e Gestione dei Sistemi Industriali, 2008*. Tesi di laurea Triennale.

4. ≪Telemanipulation System: Adept Robot for Teleoperation≫, *DML Dexterous Manipulation Laboratory*, www-cdr.stanford.edu/dml/tele_projects/telesys_adept.htm

5. ≪Adept Quattro s650H Robot User's Guide≫, rev.B, Marzo 2010

6. Adept Technology Inc., www.adept.com

Appendice A

Codice Adept Desktop

```
.PROGRAM unic()

stop_move = FALSE
temp1 = 0
temp2 = 0
temp3 = 0
tempo = 0
tempo_here = 0
tempo_alter = 0
step_x = 0
step_y = 0
step_z = 0
step_alpha = 0

; Definizione della locazione di Home
SET home = TRANS(0,0,-900,0,0,0)
TOOL TRANS(0,0,0,0,180,180)

; Creazione dell'array della posizione di Home
DECOMPOSE pos_home[] = home

; Coordinate della posizione di home
xhome = pos_home[0]
yhome = pos_home[1]
zhome = pos_home[2]
alphahome = pos_home[5]
$xhome = $LNGB(xhome*1000)
$yhome = $LNGB(yhome*1000)
$zhome = $LNGB(zhome*1000)
$alphahome = $LNGB(alphahome*1000)

;Inizializzazione variabili
time_xpc = 0
durata = 30*1000
lunrx = 1
luntx = 2
$ip_target = "147.162.95.23"

t = 0
b = 1E-04
c = 0
d = 1E-04
e = 0
f = 1E-03
g = 0
```

```
h = 1E-03
i = 0

stored_x = 0
stored_y = 0
stored_z = 0
stored_alpha = 0
des_step_x = 0
des_step_y = 0
des_step_z = 0
des_step_alpha = 0
step_x = 0
step_y = 0
step_z = 0
step_alpha = 0

posx = 0
posy = 0
posz = 0
posalpha = 0

; Limiti dello spazio di lavoro
z_limit_up = -740
z_limit_middle = -985
z_limit_down = -1240
limit_r1 = 675
limit_r2 = 350
r_var = 0
next_x = xhome
next_y = yhome
next_z = zhome
next_alpha = alphahome

max_step_x = 160
max_step_y = 160
max_step_z = 160
max_step_alpha = 160



SPEED 20, 20
ACCEL 20, 20
MOVES home

; aspetta che sia arrivato in Home
DO
UNTIL STATE(2) == 2

ALTON () 3
CPON ALWAYS
SPEED 100, 100 ALWAYS
```

```
ACCEL (0) 100, 100

;comando di movimentazione principale su cui effettuare le alterazioni
MOVE HERE

ATTACH (lunrx) "UDP"
FOPEN (lunrx, 16) "/LOCAL_PORT 65001 /BUFFER_SIZE 1024"

ATTACH (luntx) "UDP"
FOPEN (luntx, 0) $ip_target, "/REMOTE_PORT 65000"


   ;Partenza del ciclo di ricezione-movimentazione

WHILE stop_move == FALSE DO
    temp1 = TIMER(0)

    READ (lunrx) $pacchetto_dati

    rxa = LNGB($pacchetto_dati)
    rxb = LNGB($pacchetto_dati,5)
    rxc = LNGB($pacchetto_dati,9)
    rxd = LNGB($pacchetto_dati,13)
    rxe = LNGB($pacchetto_dati,17)
    checksum = LNGB($pacchetto_dati,21)-rxa-rxb-rxc-rxd-rxe

    ; VERIFICA CHECHSUM
    IF (checksum <> 0) THEN     ; verifico correttezza del pacchetto
TYPE "Errore trasmissione"     ; in caso di errore scrivo il messaggio

    ELSE


time_xpc = rxa
set_dx = rxb/1000
set_dy = rxc/1000
set_dz = rxd/1000
set_dalpha = rxe/1000*180/PI

    ;Fermo la movimentazione alla fine della trasmissione
    ; IF time_xpc == durata THEN
    ; stop_move = TRUE
;END


b = set_dx
d = set_dy
f = set_dz
h = set_dalpha
```

```
    ; Calcolo gli incrementi effettivi

des_step_x = b-c
stored_x = stored_x+des_step_x
IF stored_x >= 0 THEN
    step_x = MIN(max_step_x,stored_x)
ELSE
    step_x = MAX(-max_step_x,stored_x)
END
stored_x = stored_x-step_x
c = b


des_step_y = d-e
stored_y = stored_y+des_step_y
IF stored_y >= 0 THEN
    step_y = MIN(max_step_y,stored_y)
ELSE
    step_y = MAX(-max_step_y,stored_y)
END
stored_y = stored_y-step_y
e = d


des_step_z = f-g
stored_z = stored_z+des_step_z
IF stored_z >= 0 THEN
    step_z = MIN(max_step_z,stored_z)
ELSE
    step_z = MAX(-max_step_z,stored_z)
END
stored_z = stored_z-step_z
g = f


des_step_alpha = h-i
stored_alpha = stored_alpha+des_step_alpha
IF stored_alpha >= 0 THEN
    step_alpha = MIN(max_step_alpha,stored_alpha)
ELSE
    step_alpha = MAX(-max_step_alpha,stored_alpha)
END
stored_alpha = stored_alpha-step_alpha
i = h



    ; Calcolo della posizione futura
next_x = next_x+step_x
next_y = next_y+step_y
next_z = next_z+step_z
```

```
next_alpha = next_alpha+step_alpha

    ; Controllo dei limiti
IF ((next_z < z_limit_down) OR (next_z > z_limit_up)) THEN
    HALT
ELSE
    IF ((next_z < z_limit_up) AND (next_z > z_limit_down)) THEN
IF (SQRT(SQR(next_x)+SQR(next_y)) > limit_r1) THEN
    HALT
END
    ELSE
r_var = limit_r2-302/255*(-next_z-986)
IF (SQRT(SQR(next_x)+SQR(next_y)) > r_var) THEN
    HALT
END
    END

END
    END ;checksum


    temp2 = TIMER(0)
    ALTER () step_x, step_y, step_z, 0, 0, step_alpha
    tempo_alter = TIMER(0)-temp2


    ; MISURAZIONE DELLA POSIZIONE CORRENTE E ELABORAZIONE DATI

    temp3 = TIMER(0)
    JHERE j1, j2, j3, j4
    DECOMPOSE posizione_robot[] = HERE
    JHERE j1, j2, j3, j4
    tempo_here = TIMER(0)-temp3
    posx = posizione_robot[0]
    posy = posizione_robot[1]
    posz = posizione_robot[2]
    posalpha = posizione_robot[5]
    robot_dx = posx-xhome
    robot_dy = posy-yhome
    robot_dz = posz-zhome
    robot_dalpha = posalpha-alphahome

; Conversione in stringhe di 4 byte
    $time_xpc = $LNGB(time_xpc)
    $robot_dx = $LNGB(INT(robot_dx*1000))
    $robot_dy = $LNGB(INT(robot_dy*1000))
    $robot_dz = $LNGB(INT(robot_dz*1000))
    $robot_dalpha = $LNGB(INT(robot_dalpha*1000*PI/180))
    $j1 = $LNGB(INT(j1*1000))
    $j2 = $LNGB(INT(j2*1000))
    $j3 = $LNGB(INT(j3*1000))
```

```
    $j4 = $LNGB(INT(j4*1000))


    ;;ASSEMBLAGGIO DEL PACCHETTO DATI
     ;;TRASMISSIONE DEL PACCHETTO VIA ETHERNET PROTOCOLLO UDP

    $data_out = $time_xpc+$robot_dx+$robot_dy+$robot_dz+$robot_dalpha+
    $j1+$j2+$j3+$j4+$xhome+$yhome+$zhome

; Trasmissione dati
    WRITE (luntx) $data_out
    tempo = TIMER(0)-temp1
    WAIT

END

; Chiusura modulo UDP per la comunicazione
FCLOSE (lunrx)
DETACH (lunrx)
FCLOSE (luntx)
DETACH (luntx)

ALTOFF

.END
```

Appendice B

Codice Blocchi Funzionali Simulink

B.1 Generatore Riferimenti

```
function [t0,xi,yi,zi,alphai,xf,yf,zf,alphaf]= fcn(t)

%INIZIALIZZAZIONE VARIABILI
%posizione iniziale
t0=0;
xi=0;
yi=0;
zi=0;
alphai=0;

%posizione finale
xf=0;
yf=0;
zf=0;
alphaf=0;


if(t<=3)
    t0=0;
    %posizione iniziale
    xi=0;
    yi=0;
    zi=0;
    alphai=0;
    %posizione finale
    xf=300;
    yf=300;
    zf=110;
    alphaf=1;


elseif (t<=6)
    t0=3;
    %posizione iniziale
    xi=300;
    yi=300;
    zi=110;
    alphai=1;
    %posizione finale
    xf=200;
    yf=-100;
    zf=120;
    alphaf=-1;
```

```
elseif (t<=9)
    t0=6;
    %posizione iniziale
    xi=200;
    yi=-100;
    zi=120;
    alphai=-1;
    %posizione finale
    xf=-100;
    yf=200;
    zf=-50;
    alphaf=1;

elseif (t<=13)
    t0=9;
    %posizione iniziale
    xi=-100;
    yi=200;
    zi=-50;
    alphai=1;
    %posizione finale
    xf=-375;
    yf=-490;
    zf=155;
    alphaf=-1;

elseif (t<=19)
    t0=13;
    %posizione iniziale
    xi=-375;
    yi=-490;
    zi=155;
    alphai=-1;
    %posizione finale
    xf=440;
    yf=500;
    zf=-70;
    alphaf=1;

elseif (t<=22)
    t0=19;
    %posizione iniziale
    xi=440;
    yi=500;
    zi=-70;
    alphai=1;
    %posizione finale
    xf=0;
    yf=0;
    zf=0;
```

```
    alphaf=0;


else %se non ho più nuovi riferimenti
    t0=t;
    %posizione finale
    xf=0;
    yf=0;
    zf=0;
    alphaf=0;
    %posizione iniziale
    xi=xf;
    yi=yf;
    zi=zf;
    alphai=alphaf;

end

end
```

B.2 Generatore della Legge di Moto

```
function [a,x,y,z,alpha]=fcn(t,indice,lambda,lambda_tr,t0,xi,yi,zi,
                          alphai,xf,yf,zf,alphaf,ap_max,app_max)

%%INIZIALIZZAZIONE VARIABILI
%Sample time
s_time=0.016;
T=10;
t1=0;
t2=0;
t3=0;
Cv=0;
Ca=0;
%inizializzo la posizione di riferimento e la velocità
a=0;
alpha=0;
ap=0;
xp=0;
yp=0;
zp=0;
alphap=0;
%%FINE INIZIALIZZAZIONE

%%CALCOLO AMPIEZZE
amp_x=xf-xi;
amp_y=yf-yi;
amp_z=zf-zi;
amp_alpha=alphaf-alphai;
%Calcolo ampiezza del vettore spaziale 'a'
amp_a=sqrt(amp_x^2+amp_y^2+amp_z^2);
```

```
%TRIANGOLARE VELOCITA' (Default)
if (indice<=1 || indice>13)
    Cv=2;   %1/lambda
    Ca=4;    %1/(lambda^2)
    if((sqrt(amp_a/app_max*Ca))>(amp_a/ap_max*Cv))
        T=sqrt(amp_a/app_max*Ca);
        ap_max=amp_a/T*Cv;
    else
        T=amp_a/ap_max*Cv;
        app_max=amp_a/(T^2)*Ca;
    end

    alphap_max=amp_alpha/T*Cv;
    alphapp_max=amp_alpha/(T^2)*Ca;

    q=(t-t0)/T;
    if (q<=1)
        if (q<=1/2)
            ap=app_max*(t-t0);
            a=ap*(t-t0)/2;
            alphap=alphapp_max*(t-t0);
            alpha=alphap*(t-t0)/2;
        else
            ap=app_max*T-app_max*(t-t0);
            a=ap_max*T/2-ap*(T-(t-t0))/2;
            alphap=alphapp_max*T-alphapp_max*(t-t0);
            alpha=alphap_max*T/2-alphap*(T-(t-t0))/2;
        end
    end
end

%TRAPEZOIDALE VELOCITA'
if(indice==2)

    if (lambda<=0)
        t1=ap_max/app_max;
        t3=t1;
        t2=amp_a/ap_max-t1;
        T=t1+t2+t3;
        lambda=t1/T;
        if (lambda>1/2)
            lambda=1/2;
            Cv=1/(1-lambda);
            Ca=1/(lambda*(1-lambda));
            if ((sqrt(amp_a/app_max*Ca))>(amp_a/ap_max*Cv))
                T=sqrt(amp_a/app_max*Ca);
                ap_max=amp_a/T*Cv;
            else
                T=amp_a/ap_max*Cv;
```

```
                app_max=amp_a/(T^2)*Ca;
            end
        end

    else
        if (lambda>1/2)
            lambda=1/2;
        end
        Cv=1/(1-lambda);
        Ca=1/(lambda*(1-lambda));
        if ((sqrt(amp_a/app_max*Ca))>(amp_a/ap_max*Cv))
            T=sqrt(amp_a/app_max*Ca);
            ap_max=amp_a/T*Cv;
        else
            T=amp_a/ap_max*Cv;
            app_max=amp_a/(T^2)*Ca;
        end
    end

    alphap_max=amp_alpha/T*Cv;
    alphapp_max=amp_alpha/(T^2)*Ca;

    q=(t-t0)/T;
    if (q<=1)
        if (q<=lambda)
            ap=app_max*(t-t0);
            a=ap*(t-t0)/2;

            alphap=alphapp_max*(t-t0);
            alpha=alphap*(t-t0)/2;
        elseif (q<=(1-lambda))
            ap=ap_max;
            a=1/2*app_max*(T*lambda)^2+ap*((t-t0)-T*lambda);

            alphap=alphap_max;
            alpha=1/2*alphapp_max*(T*lambda)^2+alphap*((t-t0)-T*lambda);
        else
            ap=ap_max-app_max*((t-t0)-(T-T*lambda));
            a=1/2*app_max*(T*lambda)^2+ap_max*(T-2*T*lambda)+
             ap_max*((t-t0)-(T-T*lambda))-1/2*app_max*((t-t0)-
              (T-T*lambda))^2;

            alphap=alphap_max-alphapp_max*((t-t0)-(T-T*lambda));
            alpha=1/2*alphapp_max*(T*lambda)^2+alphap_max*(T-2*T*lambda)+
             alphap_max*((t-t0)-(T-T*lambda))-1/2*alphapp_max*((t-t0)-
              (T-T*lambda))^2;
        end
    end
end

%POLINOMALE 3° GRADO
```

```
if(indice==3)
    Cv=1.5;
    Ca=6;
    if((sqrt(amp_a/app_max*Ca))>(amp_a/ap_max*Cv))
        T=sqrt(amp_a/app_max*Ca);
    else
        T=amp_a/ap_max*Cv;
    end
    alphap_max=amp_alpha/T*Cv;
    alphapp_max=amp_alpha/(T^2)*Ca;
    q=(t-t0)/T;
    if (q<=1)
          a=amp_a*(3*q^2-2*q^3);
          alpha=amp_alpha*(3*q^2-2*q^3);
    end
end

%POLINOMALE 5° GRADO
if(indice==4)
    Cv=1.875;
    Ca=5.7733;
    if((sqrt(amp_a/app_max*Ca))>(amp_a/ap_max*Cv))
        T=sqrt(amp_a/app_max*Ca);
    else
        T=amp_a/ap_max*Cv;
    end
    alphap_max=amp_alpha/T*Cv;
    alphapp_max=amp_alpha/(T^2)*Ca;
    q=(t-t0)/T;
    if (q<=1)
          a=amp_a*(10*q^3-15*q^4+6*q^5);
          alpha=amp_alpha*(10*q^3-15*q^4+6*q^5);
    end
end

%POLINOMIALE 7° GRADO
if(indice==5)
    Cv=2.1875;
    Ca=7.5107;
    if((sqrt(amp_a/app_max*Ca))>(amp_a/ap_max*Cv))
        T=sqrt(amp_a/app_max*Ca);
    else
        T=amp_a/ap_max*Cv;
    end
    alphap_max=amp_alpha/T*Cv;
    alphapp_max=amp_alpha/(T^2)*Ca;
    q=(t-t0)/T;
    if (q<=1)
          a = amp_a*( 35*q^4 - 84*q^5 + 70*q^6 - 20*q^7 );
          alpha = amp_alpha*( 35*q^4 - 84*q^5 + 70*q^6 - 20*q^7 );
    end
```

```
end

%FREUDESTEIN 1-3
if(indice==6)
    Cv=2;
    Ca=5.3856;
    if((sqrt(amp_a/app_max*Ca))>(amp_a/ap_max*Cv))
        T=sqrt(amp_a/app_max*Ca);
    else
        T=amp_a/ap_max*Cv;
    end
    alphap_max=amp_alpha/T*Cv;
    alphapp_max=amp_alpha/(T^2)*Ca;
    q=(t-t0)/T;
    if (q<=1)
        a=amp_a*(q-1/(2*pi)*(27/28*sin(2*pi*q)+1/84*sin(6*pi*q)));
        alpha=amp_alpha*(q-1/(2*pi)*(27/28*sin(2*pi*q)+1/84*sin(6*pi*q)));
    end
end

%FREUDESTEIN 1-3-5
if(indice==7)
    Cv=2;
    Ca=5.0603;
    if((sqrt(amp_a/app_max*Ca))>(amp_a/ap_max*Cv))
        T=sqrt(amp_a/app_max*Ca);
    else
        T=amp_a/ap_max*Cv;
    end
    alphap_max=amp_alpha/T*Cv;
    alphapp_max=amp_alpha/(T^2)*Ca;
    q=(t-t0)/T;
    if (q<=1)
        a = amp_a*(q-1125/1192/(2*pi)*(sin(2*pi*q)+1/54*sin(6*pi*q) +
         1/1250*sin(10*pi*q)) );
        alpha = amp_alpha*(q-1125/1192/(2*pi)*(sin(2*pi*q) +
         1/54*sin(6*pi*q) + 1/1250*sin(10*pi*q)));
    end
end

%GUTMAN 1-3
if(indice==8)
    Cv=2;
    Ca=5.1296;
    if((sqrt(amp_a/app_max*Ca))>(amp_a/ap_max*Cv))
        T=sqrt(amp_a/app_max*Ca);
    else
        T=amp_a/ap_max*Cv;
    end
    alphap_max=amp_alpha/T*Cv;
    alphapp_max=amp_alpha/(T^2)*Ca;
```

```
    q=(t-t0)/T;
    if (q<=1)
        a = amp_a*( q - 1/pi*( 15/32*sin(2*pi*q) + 1/96*sin(6*pi*q)));
        alpha=amp_alpha*(q-1/pi*(15/32*sin(2*pi*q)+1/96*sin(6*pi*q)));
    end
end

%ARMONICA
if(indice==9)
    Cv=1.5708;
    Ca=4.9348;
    if((sqrt(amp_a/app_max*Ca))>(amp_a/ap_max*Cv))
        T=sqrt(amp_a/app_max*Ca);
    else
        T=amp_a/ap_max*Cv;
    end
    alphap_max=amp_alpha/T*Cv;
    alphapp_max=amp_alpha/(T^2)*Ca;
    q=(t-t0)/T;
    if (q<=1)
        a = amp_a/2 * (1 - cos(pi*q));
        alpha = amp_alpha/2 * (1 - cos(pi*q));
    end
end

%CICLOIDALE
if(indice==10)
    Cv=2;
    Ca=6.2832;
    if((sqrt(amp_a/app_max*Ca))>(amp_a/ap_max*Cv))
        T=sqrt(amp_a/app_max*Ca);
    else
        T=amp_a/ap_max*Cv;
    end
    alphap_max=amp_alpha/T*Cv;
    alphapp_max=amp_alpha/(T^2)*Ca;
    q=(t-t0)/T;
    if (q<=1)
        a = amp_a*( q - 1/(2*pi)*sin(2*pi*q) );
        alpha = amp_alpha*( q - 1/(2*pi)*sin(2*pi*q) );
    end
end

%ACCELERAZIONE TRAPEZOIDALE
if(indice==11 || indice==12 || indice==13)
    %Calcolo dei coefficienti necessari
    Cv=1/(1-lambda);
    Ca=1/(lambda*(1-lambda_tr)*(1-lambda));
    %Trapezoidale normale
    if (lambda<=0)
        t1=ap_max/app_max;
```

```
    t3=t1;
    t2=amp_a/ap_max-t1;
    T=t1+t2+t3;
    lambda=t1/T;
    if (lambda>1/2)
        lambda=1/2;
        Cv=1/(1-lambda);
        Ca=1/(lambda*(1-lambda_tr)*(1-lambda));
        if ((sqrt(amp_a/app_max*Ca))>(amp_a/ap_max*Cv))
            T=sqrt(amp_a/app_max*Ca);
            ap_max=amp_a/T*Cv;
        else
            T=amp_a/ap_max*Cv;
            app_max=amp_a/(T^2)*Ca;
        end
    end

else
    if (lambda>1/2)
        lambda=1/2;
    end
    Cv=1/(1-lambda);
    Ca=1/(lambda*(1-lambda_tr)*(1-lambda));
    if ((sqrt(amp_a/app_max*Ca))>(amp_a/ap_max*Cv))
        T=sqrt(amp_a/app_max*Ca);
        ap_max=amp_a/T*Cv;
    else
        T=amp_a/ap_max*Cv;
        app_max=amp_a/(T^2)*Ca;
    end
end

q0=0;
q1=lambda_tr*lambda;
q2=lambda-lambda_tr*lambda;
q3=lambda;
q4=1-lambda;
q5=q4+lambda_tr*lambda;
q6=1-lambda_tr*lambda;
q7=1;

if(indice==12)  %Trapezoidale modificata lambda=1/2,lambda_tr=1/8
  Cv=2;
  Ca=4.9;
  if((sqrt(amp_a/app_max*Ca))>(amp_a/ap_max*Cv))
    T=sqrt(amp_a/app_max*Ca);
  else
    T=amp_a/ap_max*Cv;
  end
  q0=0;
  q1=1/8;
```

```
  q2=3/8;
  q3=0.5;
  q4=q3;
  q5=1-q2;
  q6=1-q1;
  q7=1;
elseif(indice==13)  %Sinusoidale modificata lambda_tr=1/8
  Cv=1.75;
  Ca=5.5;
  if((sqrt(amp_a/app_max*Ca))>(amp_a/ap_max*Cv))
    T=sqrt(amp_a/app_max*Ca);
  else
    T=amp_a/ap_max*Cv;
  end
  q0=0;
  q1=1/8;
  q2=q1;
  q3=0.5;
  q4=q3;
  q5=1-q1;
  q6=q5;
  q7=1;
end

%Tempo normalizzato
q=(t-t0)/T;

d1=q1-q0;
d2=q2-q1;
d3=q3-q2;
d4=q4-q3;
d5=q5-q4;
d6=q6-q5;
d7=q7-q6;

a11 =2*d1/pi+d2+2*d3/pi;
a12=-2*d5/pi-d6-2*d7/pi;
a21=2*d1/pi*((pi-2)/pi*d1+d2/2)+(2*d1/pi+d2)*(d2/2+(pi-2)/pi*d3)+
(2*d1/pi+d2+2*d3/pi)*(2*d3/pi+d4+2*d5/pi);
a22= (2*d7/pi+d6)*((pi-2)/pi*d5+d6/2)+2*d7/pi*(d6/2+(pi-2)/pi*d7);

A = -a12/(a11*a22-a12*a21);
B = a11/(a11*a22-a12*a21);

%Velocità nei punti notevoli
V1 = 2*d1/pi*A;
V2 = A*d2 + V1
V3 = 2/pi*d3*A+V2;
V4 = V3;
V5 = V4-B*2*d5/pi;
V6 = V5-B*d6;
```

```
    %Spostamenti punti notevoli
    S1=2*d1^2/pi*A-(2*d1/pi)^2*A;
    S2=A/2*d2^2+V1*d2+S1;
    S3=(2/pi*d3)^2*A+V2*d3+S2;
    S4=S3+A*d4*(2*d1/pi+d2+2*d3/pi);
    S5=S4+A*d5*(2*d1/pi+d2+2*d3/pi)-B*2*d5^2/pi*(1-2/pi);
    S6=S5+V5*d6-B*d6^2/2;

    if ((t-t0)<=T)

        if (q<=q1)
            a=amp_a*(2*d1/pi*A*(q-2*d1/pi*sin(pi*q/2/d1)));
            alpha=amp_alpha*(2*d1/pi*A*(q-2*d1/pi*sin(pi*q/2/d1)));

        elseif (q<=q2)
            a=amp_a*(A/2*(q-q1)^2+V1*(q-q1)+S1);
            alpha=amp_alpha*(A/2*(q-q1)^2+V1*(q-q1)+S1);

        elseif (q<=q3)
            a=amp_a*((((2/pi*d3)^2)*(A*(1-cos(pi*(q-q2)/2/d3)))+
             V2*(q-q2)+S2));
            alpha=amp_alpha*((((2/pi*d3)^2)*(A*(1-cos(pi*(q-q2)/2/d3)))+
            V2*(q-q2)+S2));
        elseif (q<=q4)
            a=amp_a*(V3*(q-q3)+S3);
            alpha=amp_alpha*(V3*(q-q3)+S3);
        elseif (q<=q5)
            a=amp_a*(-B*2/pi*d5*(q-q4-2/pi*d5*sin(pi*(q-q4)/2/d5))+
             V4*(q-q4)+S4);
            alpha=amp_alpha*(-B*2/pi*d5*(q-q4-2/pi*d5*sin(pi*(q-q4)/2/d5))+
            V4*(q-q4)+S4);
        elseif (q<=q6)
            a=amp_a*((S5+V5*(q-q5)-B*(q-q5)^2/2));
            alpha=amp_alpha*((S5+V5*(q-q5)-B*(q-q5)^2/2));
        else
            a=amp_a*((((2/pi*d7)^2)*B*(cos(pi*(q-q6)/2/d7)-1)+V6*(q-q6)+S6));
            alpha=amp_alpha*((((2/pi*d7)^2)*B*(cos(pi*(q-q6)/2/d7)-1)+
             V6*(q-q6)+ S6));
        end
    end
end


if ((t-t0)<=T)
%   %Posizione
    x=xi+a*amp_x/amp_a;
    y=yi+a*amp_y/amp_a;
    z=zi+a*amp_z/amp_a;
    alpha=alphai+alpha;
else
    x=xf;
```

```
    y=yf;
    z=zf;
    alpha=alphaf;
end

end
```

B.3 Cinematica Diretta di Posizione

```
function [x0,y0,z0,alpha]= fcn(q1g,q2g,q3g,q4g,xhome,yhome,zhome)

%Dimensioni Manipolatore
ai=0.375
bi=0.825
leq=ai*cos(pi/4)
OP=0.275
Zoff1=0.104775
Zoff2=0.086511
l=0.08075
l1=0.04425
l2=0.025
h=0.11
h1=0.025
dxtool=0.08075/2

%Converto le posizioni dei motori in radianti
q1=q1g*pi/180
q2=q2g*pi/180
q3=q3g*pi/180
q4=q4g*pi/180

%calcolo dei punti di attacco dei motori
xp1=-OP*cos(pi/4)
xp2=OP*cos(pi/4)
xp3=OP*cos(pi/4)
xp4=-OP*cos(pi/4)

yp1=-OP*cos(pi/4)
yp2=-OP*cos(pi/4)
yp3=OP*cos(pi/4)
yp4=OP*cos(pi/4)

zp=-Zoff1


%Calcolo dei punti fittizzi di attacco dei motori traslati delle dimensioni
 della tavoletta finale
xpf1=xp1+l+l2
xpf2=xp2-l1
xpf3=xp3-l1
```

```
xpf4=xp4+l+l2


ypf1=yp1+0.08
ypf2=yp2+0.08
ypf3=yp3-0.08
ypf4=yp4-0.08

zpf=zp-Zoff2



%Calcolo dei centri sfere
x1=-leq*cos(q1)+xpf1
x2=leq*cos(q2)+xpf2
x3=leq*cos(q3)+xpf3
x4=-leq*cos(q4)+xpf4

y1=-leq*cos(q1)+ypf1
y2=-leq*cos(q2)+ypf2
y3=leq*cos(q3)+ypf3
y4=leq*cos(q4)+ypf4

z1=-ai*sin(q1)+zpf
z2=-ai*sin(q2)+zpf
z3=-ai*sin(q3)+zpf
z4=-ai*sin(q4)+zpf

%Calcolo dei parametri
A=(y2-y1)/(x1-x2)
B=(z2-z1)/(x1-x2)
C=(x1^2-x2^2+y1^2-y2^2+z1^2-z2^2)/(2*(x1-x2))
D=((x2-x3)*B+(z2-z3))/((x3-x2)*A+(y3-y2))
E=(2*C*(x2-x3)+x3^2-x2^2+y3^2-y2^2+z3^2-z2^2)/(2*A*(x3-x2)+2*(y3-y2))
F=A^2*D^2+B^2+2*A*B*D+D^2+1
G=2*(A^2*E*D+A*B*E+A*C*D+B*C+D*E-A*D*x3-B*x3-D*y3-z3)
H=A^2*E^2+C^2+E^2+2*A*C*E-2*A*E*x3-2*C*x3-2*E*y3+x3^2+y3^2+z3^2-bi^2

%Calcolo delle coordinate del punto finale e converto in [mm]
zt=((-G-sqrt(G^2-4*F*H))/(2*F))
yt=D*zt+E
xt=A*yt+B*zt+C

x0=xt*1000-xhome
y0=yt*1000-yhome
z0=zt*1000-zhome
alpha=0
```