Università degli Studi di Padova
Facoltà di Ingegneria

Finito di scrivere il giorno 22 luglio 2010 utilizzando LaTeX 2ε

UNIVERSITÀ DEGLI STUDI DI PADOVA
FACOLTÀ DI INGEGNERIA

—

DIPARTIMENTO DI INNOVAZIONE MECCANICA E GESTIONALE

—

TESI DI LAUREA TRIENNALE IN INGEGNERIA
DELL'AUTOMAZIONE

PIANIFICAZIONE DEL MOVIMENTO DI UN MANIPOLATORE PLANARE A TRE GRADI DI LIBERTÀ

RELATORE: CH.MO PROF. ING. GIULIO ROSATI

LAUREANDO: MEFO TEFO CHRISTIANE DIANE

ANNO ACCADEMICO 2009-2010

alla mia famiglia...

“

Indice

Sommario

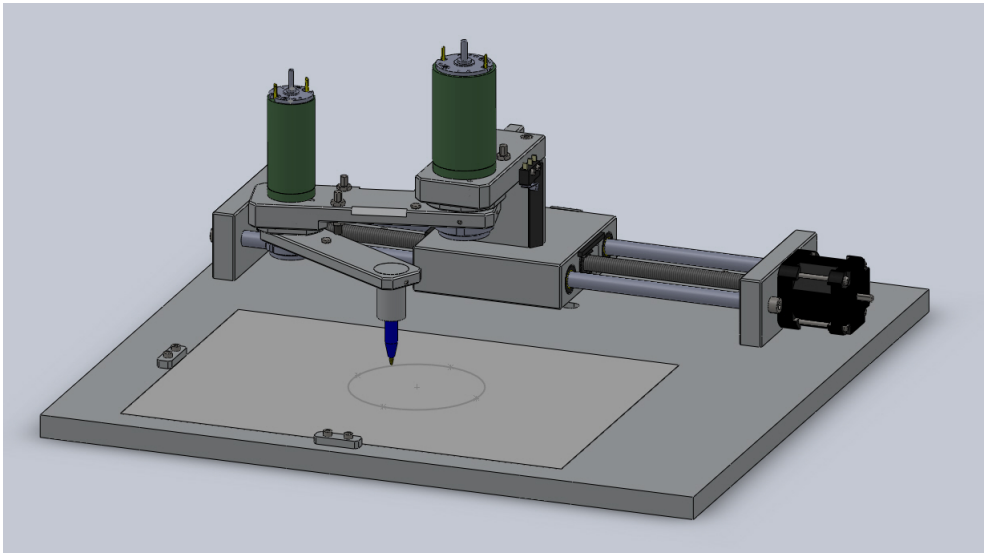

Figura 1: Manipolatore con tre gradi di libertà

Il progetto di tesi è stato basato sulla costruzione di un manipolatore planare a tre gradi di libertà composto da : un catrello che effettua un movimento rettilineo su una guida lineare collegato ad un motore stepper ; e da due link che ruotano attorno agli assi dei motori in continua a cui sono collegati .

Il progetto è stato realizzato da un gruppo di cinque studenti del corso di ingegneria dell'automazione Presso i laboratori del DIMEG e a ognuno di questi è stato attribuito specifici argomenti di tesi per la realizzazione e il funzionamento del manipolatore. La parte elettrica è stata quella di costruire i driver in tensione e corrente per il motore stepper e per i motori in continua. La parte meccanica riguardava il disegno in 3D e il montaggio dei pezzi del manipolatore.

Il mio lavoro nel gruppo è stato quello di implementare un software in Matlab per la pianificazione del movimento del manipolatore e la comunicazione dei dati con il programma creato in Simulink da uno studente del gruppo per il controllo del manipolatore.

Introduzione

Oggetto della tesi è un progetto software per la pianificazione del movimento di un manipolatore a tre gradi di libertà progettato e costruito da un gruppo di studenti . L'obiettivo della pianificazione delle traiettorie è quello di creare gli ingressi di riferimento per il sistema di controllo del moto [1] controllo che viene effettuato nel nostro progetto col programma SIMULINK su un altro computer per cui la necessità di comunicazione dei dati tramite l'uso delle pnet. Per permettere all'utente di specificare un numero di parametri che caratterizzano la traiettoria desiderata; abbiamo implementato un interfaccia utente usando la GUI in MATLAB.

Il lavoro di tesi sotto descritto tratta : della pianificazione di movimento di un manipolatore planare a tre gradi di libertà ; della creazione dell'interfaccia utente e della comunicazione dei dati tramite pnet con SIMULINK per il controllo di posizione.

Per poter vedere in modo realistico il movimento effettuato dal manipolatore abbiamo usato in Matlab delle funzioni che permettono la costruzione di oggetti per cui ; Nel capitolo 1 presenteremo la struttura di ogni parte del manipolatore disegnate in Matlab usando le stesse misure per la costruzione in 3D del robot .

Il calcolo delle traiettorie e della movimentazione del manipolatore sono illustrati nel capitolo 2 che tratta principalmente dell'argomento sulla pianificazione del movimento in cui calcoliamo tutte le cinematiche dirette e inverse ,tutti i tipi e leggi di moto applicate per ogni motore per la movimentazione del robot.

Per poter sfruttare al meglio il vantaggio di un grado di libertà in più del robot , abbiamo creato una funzione di ottimizzazione sul movimento del manipolatore il capitolo 3 ne fa una descrizione e espone anche i vincoli presente sul robot e

sul suo spazio di lavoro.

L'uso della Gui in MATLAB si è rivelato abbastanza necessario per creare un interfaccia utente in cui ogni funzione creata viene chiamata premendo o selezionando una casella desiderata dall'utente nel capitolo 4 vengono presentate e spiegate le funzioni usate per implementare l'interfaccia e il modo di usarlo .

Dato che il controllo di posizione del robot viene fatta da un altro programma che gira su un computer diverso ,nel nostro lavoro abbiamo anche realizzato la parte sull'invio e ricezione di dati attraverso i socket usando i pnet da questo il movimento del manipolatore viene effettuato con la posizione reale di ogni link ricevuta da SIMULINK . Nel capitolo 4 facciamo una spiegazione dettagliata presentando le funzioni e software utilizzate poi ne descriviamo le tappe di funzionamento del programma.

Capitolo 1

Struttura del manipolatore

In questo primo capitolo faremo una presentazione generale del manipolatore spiegando i tre gradi di libertà e descriveremo come sono stati ottenuti ogni componente del robot attraverso le funzioni Matlab utilizzate.

1.1 Presentazione generale del manipolatore a tre gradi di libertà

Un manipolatore è una macchina la cui struttura meccanica è normalmente costituita da una serie di elementi rigidi generalmente collegati in serie e connessi tra di loro mediante accoppiamenti di rotazione e/o di scorrimento relativo tra le componenti che costituiscono il robot .
Il manipolatore ha generalmente diversi gradi di libertà ed ha lo scopo di afferrare e/o movimentare oggetti(pezzi o utensili)e può essere comandato da un operatore, da un controllore elettronico programmabile o da altri sistemi [2].
La figura 1.2 viene usata nell'interfaccia in Matlab per visualizzare la posizione reale del robot ed essa è composta :

- Dalla piastra su cui viene appoggiato il manipolatore
- Dal manipolatore .
- Dal foglio su cui viene disegnato le traiettorie effettuate dal dispositivo d'estremità che risulta nel nostro caso essere una penna .

Il manipolatore nella figura come detto dall'inizio è di tipo planare quindi il movimento dell'end effector viene confinato solo nel piano orizzontale .Questo fattore dipende dalle caratteristiche strutturali del manipolatore . In effetti il movimento dell'end effector del nostro manipolatore dipende dalla posizione dei due membri e del carrello che sono montati in maniera seriale,però a differenza dei robot seriali che hanno una base fissa il manipolatore che abbiamo costrutto presenta il vantaggio di avere una base mobile che fa sì che l'area di lavoro da considerare risulta più ampia di quella del robot SCARA.
Lo spazio di lavoro di questo robot ad ogni posizione del carrello corrisponde a quella di un manipolatore con la base fissa però per sfruttare il vantaggio della base movibile abbiamo implementato nel capitolo 3 l'algoritmo per l'ottimizzazione del movimento del manipolatore spostando il carrello in modo opportuno asseconda della posizione del punto finale da raggiungere .Da questa considerazione tutti i punti che occupano l'aria non raggiungibile per il robot a base fissa risultano raggiungibile spostando la base .
Il posizionamento delle diverse coordinate ai giunti che descrivono il moto del robot nella figura 1.1

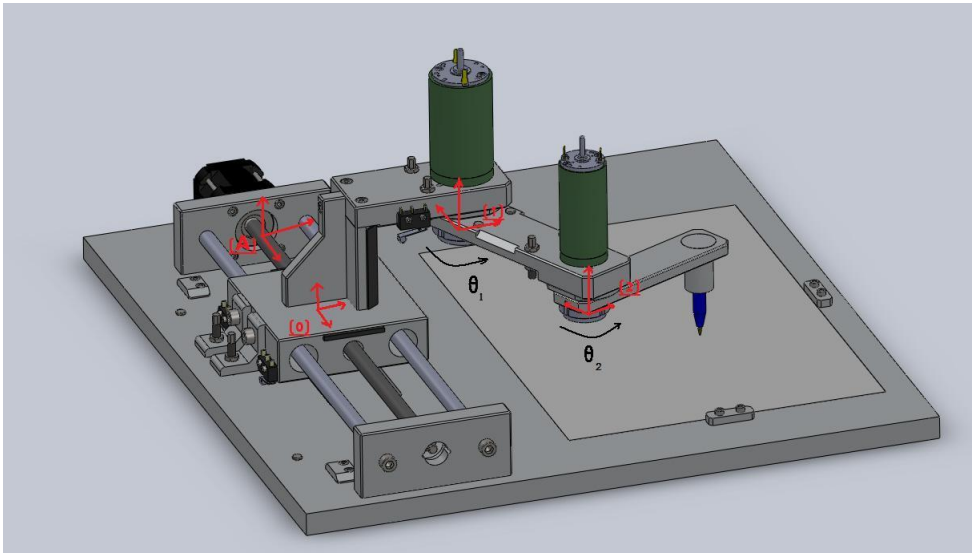

Figura 1.1: manipolatore con tutte le terne di riferimento e posizioni degli assi.

sono :

- Il primo si trova sul carrello e rappresenta la terna (0).
- Il secondo tra carrello e il primo membro e rappresenta la terna (1).
- Il terzo tra i due membri e rappresenta la terna (2).

Queste coordinate mettono in evidenza la presenza di un motore elettrico collegato ad ogni giunto rotoidale o prismatico tramite quali sono ottenuti i moti relativi dei differenti elementi del robot. Come si può notare, nel nostro caso risultano in totale tre che giustificano i tre gradi di libertà del manipolatore .

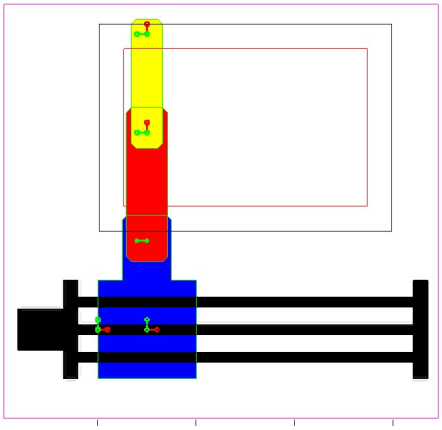

Figura 1.2: disegno con matlab

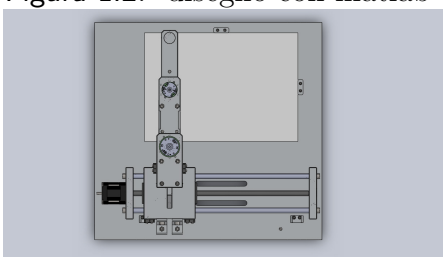

Figura 1.3: disegno con matlab

Figura 1.4: Manipolatore con bracci allineati

1.2 Descrizione di ogni componente

In questo paragrafo faremo una descrizione delle misure di ogni componente rigido del manipolatore disegnato in Matlab e dei diversi tipi di motore presenti su ogni giunto per la movimentazione del robot.

1.2.1 Guida lineare

Sulla figura 1.5 viene rappresentato la guida lineare che risulta essere la base fissa su cui è montato il manipolatore. La guida assieme al carrello costituiscono un

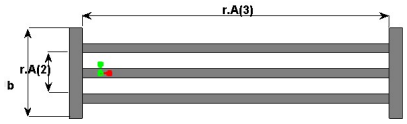

Figura 1.5: Disegno con Matlab

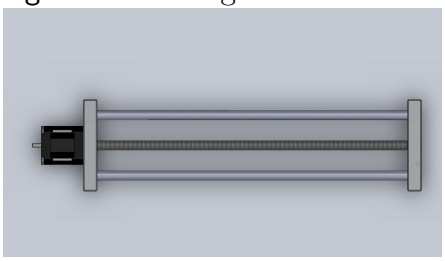

Figura 1.6: disegno con solid works

Figura 1.7: Guida lineare su cui è appoggiato il manipolatore

giunto prismatico questo dovuto alla presenza del motore passo-passo .
Come si vede dalla figura 1.5 ; Le misure della guida lineare sono salvate dentro la strutture del robot in diverse variabili in cui la lunghezza vale $r.A(3) = 300mm$ e la larghezza $r.A(1) = 66mm$.
Dato che la cinematica ai giunti viene calcolata rispetto ad ogni riferimento di ogni giunto di seguito le coordinate dell'estremità del manipolatore sono riportate rispetto al riferimento della guida che rappresenta l'origine di tutto il manipolatore essendo la parte fissa del robot. l'origine della guida si trova sulla vite del motore step e distante $d = 20mm$ dal blocco rettangolare di sinistra.
La lunghezza della guida utile per il movimento lineare del carrello è $L_{utile} = 200mm$ ovviamente minore di quello effettiva della guida dovuto alla presenza dei blocchi di lettura dei sensori induttivi presenti sul che servono per la calibrazione del manipolatore per il controllo di posizione essi sono situati a una di distanza $d_{sensoriinduttivo} = 50mm$.
Sulla guida sono anche presenti due blocchi che leggono i fini corsa meccanici

che permettono di togliere l'alimentazione a tutti i motori del manipolatore nel caso in cui il carrello si trovasse ad una certa posizione cioè che sorpassasse quelle posizioni posizione . Si trovano a una distanza di $d_{finecorsamecc} = 20mm$ dagli estremità della guida lineare in seguito verranno presi in considerazione nel capitolo 3 per l'ottimizzazione del movimento tenendo conto dei vincoli presenti sul manipolatore.

1.2.2 Il carrello

il carrello in figura 1.8 è un quadrato di $r.B(2) = 100mm$ di lato su cui è montato una struttura fissa di forma rettangolare alta (h=109mm) . Sulla figura che

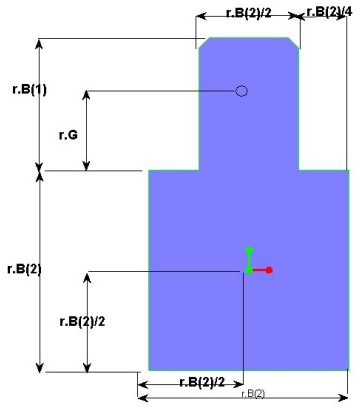

Figura 1.8: disegno con matlab

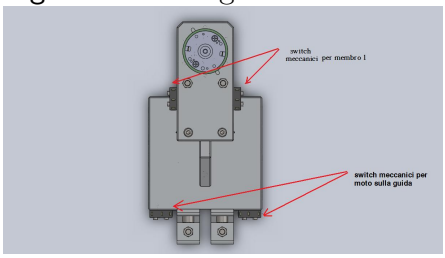

Figura 1.9: disegno 3D

Figura 1.10: Carrello

rappresenta la vista di sopra del carrello come nelle altre sono riportate :

- Le dimensioni del carrello usato per il disegno meccanico in 3D del manipolatore.

- Il riferimento del carrello rispetto a quello della guida lineare per il calcolo della cinematica diretta , si trova ad una distanza di $r.B(2)/2 = 50mm$ da questa e corrisponde nella configurazione del manipolatore in figura 1.2 alla posizione del blocco di lettura di un fine corsa meccanico.

- Il cerchietto per individualizzare il posizionamento del secondo motore in continua , si trova ad una distanza di $r.B(29/2 + r.G = 90mm$ dal riferimento del carrello e esso fa il collegamento del carrello al primo braccio del manipolatore.

Sul l'organo sopra il carrello sono state messe delle fine corse meccanici da tutte i due lati per fare in modo che il primo membro non superi un certo angolo cosi da evitare l'usura dei componenti dovuto ai strofinamenti tra di loro .
Il carrello del manipolatore in questa analisi è un organo movibile rispetto ai robot SCARA che hanno una base fissa con due membri. Per questo motivo che più avanti nel nostro lavoro abbiamo voluto sfruttare questo grado di libertà in più sullo spazio di lavoro.
Sulla guida lineare è attaccato il motore stepper con la vite su cui è montato una chiocciola alla quale è collegato il carrello e che permette a quest'ultimo di effettuare un movimento rettilineo lungo la guida lineare sopra descritta. Dall'analisi appena fatto possiamo confermare che l'insieme costituito dalla guida lineare-carrello forma un giunto prismatico che risulta a questo punto il primo giunto del manipolatore e quindi il primo grado di libertà dei tre come annunciato all'inizio di questo capitolo.

1.2.3 Il primo membro

Il primo membro è la seconda componente movibile e ha le stesse caratteristiche di movimento come quello del robot SCARA. La distanza dagli assi dei due motori inseriti è di $r.L(1) = 110mm$ lunghezza che viene usata per il calcolo delle cinematiche . Su quest'organo del manipolatore viene anche messo dei sensori induttivi per la calibrazione del movimento del secondo braccio .
Il motore che permette la movimentazione del primo membro è un motore in continua . L'insieme carrello-motore-membro in questo caso costituisce un giun-

Figura 1.11: disegno con matlab

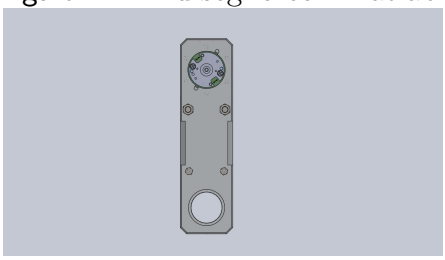

Figura 1.12: disegno 3D

Figura 1.13: Primo Membro

to rotoidale dato che il primo membro effettua un movimento rotatorio attorno all'asse del motore .
Il suo moto è condizionato dalla presenza degli switch meccanici figura 1.9 sul carrello che fanno in modo che il braccio non possa fare un angolo di $r.th2(1) = 125gradi$ o minore di $r.th2(2) = -125gradi$ vedere figura 1.17 in cui è rappresentato l'angolo massima che può fare il primo membro .

1.2.4 Il secondo membro

Essi ha una lunghezza più piccola del primo e ci si trova all'estremità l'end effector la cui asse è distante da quello del secondo motore di $r.L(2) = 100mm$ .L'insieme primo membro-motore-secondo membro costituisce un giunto rotoidale in effetti il membro due effettua un movimento rotoidale attorno all'asse del secondo motore. Il movimento di quest'ultimo organo del manipolatore dipende da quello del primo membro e viene anche condizionato dalla presenza degli switch meccanici presenti sul primo membro che fanno si che non puo andare oltre un certo angolo i valori limiti degli angoli sono salvati dentro il vettore $r.th2$.

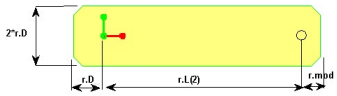

Figura 1.14: disegno con matlab

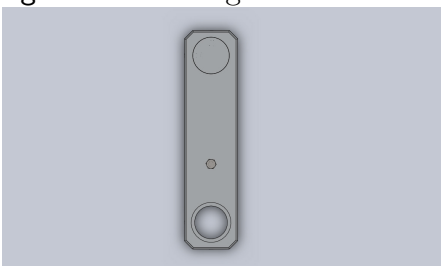

Figura 1.15: disegno 3D

Figura 1.16: Secondo Membro

1.3 Funzioni usate in Matlab per il disegno del manipolatore

le componenti descritte sopra sono ricavate in Matlab usando la funzione patch nella libreria di Matlab . Questa funzione è utile per il disegno degli oggetti. Nella funzione creaMP è presente la definizione di tutte le misure delle varie componenti sotto forma di matrice come esempio abbiamo :

```
% CARRELLO
r.B = [66 100] ; % B(2) lung e larg. B(1) lung testa
r.P00 =[[-r.B(2)/2 -r.B(2)/2 0 1]'
[r.B(2)/2 -r.B(2)/2 0 1]'
[r.B(2)/2 r.B(2)/2 0 1]'
[1/4*r.B(2) r.B(2)/2 0 1]'
[1/4*r.B(2) r.B(2)/2+r.B(1)-a 0 1]'
[1/4*r.B(2)-a r.B(2)/2+r.B(1) 0 1]'
[-1/4*r.B(2)+a r.B(2)/2+r.B(1) 0 1]'
[-1/4*r.B(2) r.B(2)/2+r.B(1)-a 0 1]'
[-1/4*r.B(2) r.B(2)/2 0 1]'
[-r.B(2)/2 r.B(2)/2 0 1]'];
% MEMBRO 1
r.C = [21 25]; % 2*C(1) larghezza C(2) spazio tra secondo motore e fine membro 1
r.P11 = [[-r.C(1) -r.C(1)+a 0 1]'
[-r.C(1)+a  -r.C(1) 0 1]'
[r.L(1)+r.C(2)-a  -r.C(1) 0 1]'
[r.L(1)+r.C(2) -r.C(1)+a 0 1]'
[r.L(1)+r.C(2) r.C(1)-a 0 1]'
```

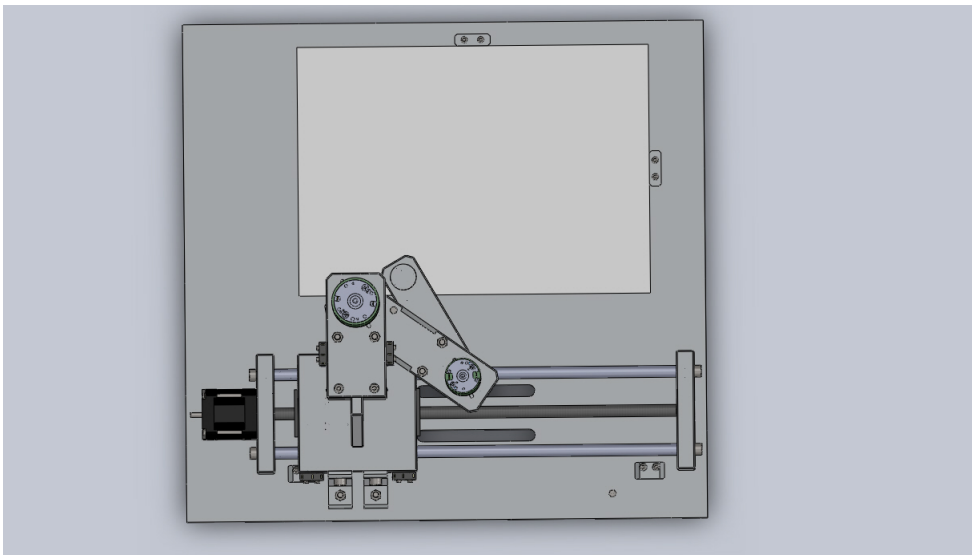

Figura 1.17: organi del manipolatore in configurazione limite

```
[r.L(1)+r.C(2)-a r.C(1) 0 1]'
[-r.C(1)+a  r.C(1) 0 1]'
[-r.C(1) r.C(1)-a 0 1]'];
% MEMBRO 2
r.D = 16 ; % 2*D largezza membro 2
r.mod = 15 ; % spazio tra la testa e l'end effettor
r.P22 = [[-r.D -r.D+a 0 1]'
[-r.D+a -r.D 0 1]'
[r.L(2)+r.mod-a -r.D 0 1]'
[r.L(2)+r.mod -r.D+a 0 1]'
[r.L(2)+r.mod r.D-a 0 1]'
[r.L(2)+r.mod-a r.D 0 1]'
[-r.D+a r.D 0 1]'
[-r.D r.D-a 0 1]'];
```

poi la funzione initdisMP2 che costruisce ogni componenti chiamando la funzione patch ed attribuisce ad ognuno un riferimento che si sono avverati utile per il calcolo delle cinematiche. qua viene riportato il codice in Matlab per la costruzione degli oggetti.

```
% disegna il carrello
r.fig.obj = patch(r.P00(1,:),r.P00(2,:),'b','FaceAlpha'
,.5,'EdgeColor','g');
set(r.fig.obj,'Parent',r.fig.link0) %rif. carrello
% disegna il membro 1
```

```
r.fig.obj = patch(r.P11(1,:),r.P11(2,:),'r','FaceAlpha'
,.5,'EdgeColor','g');
set(r.fig.obj,'Parent',r.fig.link1) %rif. membro 1
% disegna il membro 2
r.fig.obj = patch(r.P22(1,:),r.P22(2,:),'y','FaceAlpha'
,.5,'EdgeColor','g');
set(r.fig.obj,'Parent',r.fig.link2) %rif. membro 2
% disegna la guida
r.fig.obj = patch(r.Xg,r.Yg,'k','FaceAlpha',.5);
set(r.fig.obj,'Parent',r.fig.link4) % rif. guida
```

Dalla definizione delle misure e dalla costruzione delle oggetti siamo passati alla tappa della movimentazione per cui di seguito il prossimo argomento sara dedicato fondamentalmente alla pianificazione del movimento del nostro manipolatore per cui analizzeremo il calcolo di tutte le cinematiche .

Capitolo 2

Pianificazione del movimento

Con la pianificazione della traiettoria si intende stabilire la modalità con cui si vuole che evolve il movimento del manipolatore, da una postura iniziale ad una postura finale. Si tratta di definire sia il percorso geometrico sia la legge di moto da realizzare (ossia la dipendenza temporale di posizioni, velocità ed accelerazioni) [3]. Per la pianificazione l'utente specifica un numero ristretto di parametri:

- Per il percorso: punti estremi , eventuali punti intermedi.
- la legge di moto desiderata.

Per stabilire il movimento del manipolatore si intende stabilire il movimento che ogni asse deve eseguire; Per cui effettueremo la Pianificazione del movimento di ogni link. Poi di seguito studieremo prima di tutto le cinematiche dirette e inverse di ogni link importante per la movimentazione poi faremo la pianificazioni delle traiettorie nello spazio operativo e nello spazio dei giunti usando per tutti quanti lo stesso tipo di legge oraria .

2.1 Cinematica diretta

Con cinematica diretta si indica il problema che consiste nel calcolare quale moto assume l'organo terminale del robot quando viene data una certa configurazione dei giunti. La risoluzione di questi problemi consiste semplicemente nel calcolo delle matrici di posizione dell'end-effector dato il valore delle sue coordinate ai giunti ,

Note le coordinate ai giunti cioè la posizione del carrello T_x per l'accoppiamento prismatico e dei due membri Θ_1 e Θ_2 (figura 2.1)per l'accoppiamento rotoidale come mostrato in figura 2.1.
Per trovare la posizione del l'organo terminale rispetto alla guida che è la base fissa del nostro manipolatore abbiamo calcolato la matrice $T_{3A} = T_{0A}T_{10}T_{21}T_{32}$.

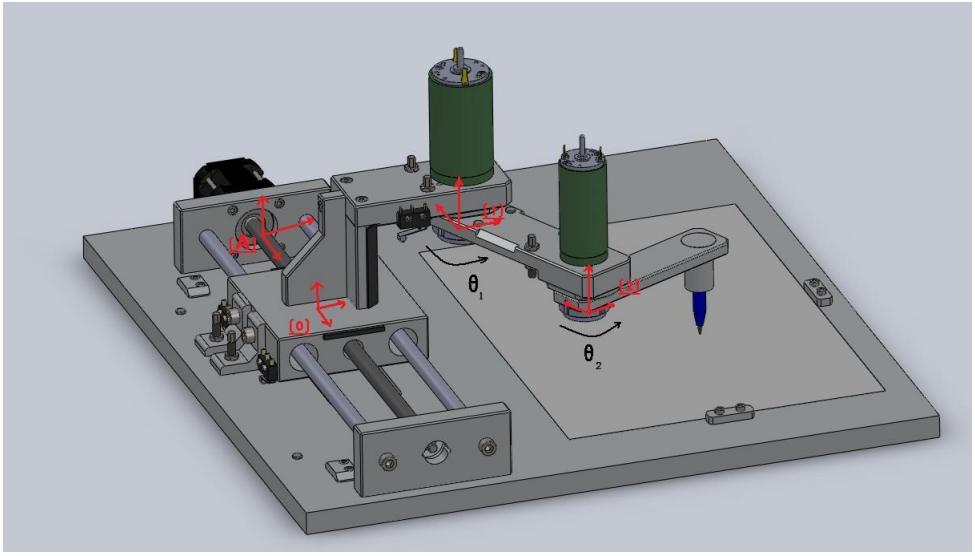

Figura 2.1: manipolatore con tutte le terne di riferimento e posizioni degli assi.

abbiamo fatto coincidere ad ogni accoppiamento (prismatico e rotoidale) del manipolatore con un asse coordinato così abbiamo ottenuto le seguenti matrici di posizione di ogni terne:

- **Accoppiamento prismatico** tra la guida lineare e il carrello vedere figura 2.1.

$$T_{0A} = \left[\begin{array}{ccc|c} 1 & 0 & 0 & Tx \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

  Rappresenta la matrice posizione della terna (0) del carrello rispetto alla terna fissa(A) sulla guida .

- **Primo Accoppiamento rotoidale** tra carrello e primo membro

$$T_{01} = \left[\begin{array}{ccc|c} \cos\vartheta_1 & -\sin\vartheta_1 & 0 & 0 \\ \sin\vartheta_1 & \cos\vartheta_1 & 0 & B_2/2 + G \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

Rappresenta la matrice di posizione della terna (2) rispetto alla terna (1) dove la posizione angolare tra le due terne viene descritta dalla matrice di rotazione e l'altro blocco definisce la posizione dell'origine (0) rispetto al alla terna (A) roto-traslazione del primo membro membro rispetto al carrello.

- **Secondo accoppiamento rotoidale** tra il primo e il secondo membro.

$$T_{21} = \left[\begin{array}{ccc|c} \cos\vartheta_2 & -\sin\vartheta_2 & 0 & L_1 \\ \sin\vartheta_2 & \cos\vartheta_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

Abbiamo assegnato anche all'estremità del manipolatore una terna (3) La cui matrice di posizione rispetto alla terna (2) del secondo giunto rotoidale è la seguente.

$$T_{32} = \left[\begin{array}{ccc|c} 1 & 0 & 0 & L_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

L'obbiettivo di quest'analisi essendo di trovare la matrice di posizione tra la terna (3) dell'end-effector rispetto alla terna (A) della base fissa del manipolatore; ricaviamo la matrice T_{3A} :

$$T_{3A} = \left[\begin{array}{ccc|c} c_1c_2 - s_1s_2 & -(s_1c_2 + s_2c_1) & 0 & L_2(c_1c_2 - s_1s_2) + L_1c_1 + T_x \\ s_1c_2 + s_2c_1 & c_1c_2 - s_1s_2 & 0 & L_2(s_1c_2 + s_2c_1) + L_1s_1 \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

ove $s_1 = \sin\vartheta_1 \quad s_2 = \sin\vartheta_2 \quad c_1 = \cos\vartheta_1 \quad c_2 = \cos\vartheta_2$. Trovando La matrice T_{3A} possiamo adesso convertire le coordinate dell'estremità del manipolatore dal suo sistema di riferimento a quello della guida lineare. cioè la posizione ad ogni istante dell'end effector rispetto a una coordinata fissa viene data dalla seguente relazione conoscendo già a priori le coordinate ai giunti a quelli istanti sono (Tx $\vartheta_1 \quad \vartheta_2$) :

$$P_A = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = T_{3A} P_3$$

visto che la terna (3) corrisponde all'end-effector abbiamo in questo caso

$$P_3 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

da cui

$$P_A = \begin{bmatrix} L_2(c_1c_2 - s_1s_2) + L_1c_1 + T_x \\ L_2(s_1c_2 + s_2c_1) + L_1s_1 \\ 0 \\ 1 \end{bmatrix}$$

2.1.1 Implementazione cinematica diretta con Matlab

La funzione cindirMP2 permette di calcolare la cinematica diretta del manipolatore e ha come ingresso le variabili r e i :

- dove r rappresenta La struttura del robot che contiene la posizione di ogni asse definito nel vettore r.q in cui il primo elemento rappresenta la posizione della terna di riferimento del carrello ; il secondo quella del primo membro e la terza posizione è quella della terna del terzo membro.

- dove i rappresenta la posizione i-esima del vettore r.q .

Questa funzione ritorna tutte le matrici di posizione e la posizione dell'end effector ,definite precedentemente usando la funzione makehgtform della libreria di Matlab
Qualche linee di codice usato:

```
r.T0A =  makehgtform('translate',[r.q(i,1) 0 0]);
r.T10 =  makehgtform('translate',[0 r.B(2)/2+r.G 0]) *...
.. makehgtform('zrotate',r.q(i,2));
r.T21 =  makehgtform('translate',[r.L(1) 0 0]) * ...
...makehgtform('zrotate',r.q(i,3));
r.T32 =  makehgtform('translate',[r.L(2) 0 0]);

r.P = r.T3A * [0 0 0 1]';
```

2.2 Cinematica inversa

Con cinematica inversa in generale si tratta di calcolare le coordinate ai giunti e delle sue derivate una volta assegnate le matrice di posizione. Noi ci siamo fermati al calcolo delle coordinate ai giunti dato che il controllo implementato poi da Simulink è di posizione.L'analisi di posizione è stata fatta risolvendo un sistema non lineare che ha un numero di incognite pari al numero dei giunti cioè 3 ricordando le 3 gradi di libertà descritti nel capitolo 1 .
Conoscendo la matrice T_{3A} ci proponiamo di calcolare la posizione di ogni asse T_x e $\vartheta_1 \quad \vartheta_2$ prima di tutto notiamo che nella matrice T_{3A} si ha $T_{11} = T_{22} \quad T_{12} = -T_{21}$ in base alle formule trigonometriche abbiamo $T_{11} = c_1c_2 - s_1s_2 = \cos(\vartheta_1 + \vartheta_2)$ e $T_{12} = s_1c_2 + s_2c_1 = \sin(\vartheta_1 + \vartheta_2)$ ponendo $\Psi = \vartheta_1 + \vartheta_2$ la matrice T_{A3} ha la seguente struttura (2.2)

$$T_{3A} = \left[\begin{array}{ccc|c} \cos\Psi & -\sin\Psi & 0 & x \\ \sin\Psi & \cos\Psi & 0 & y \\ 0 & 0 & 1 & z \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

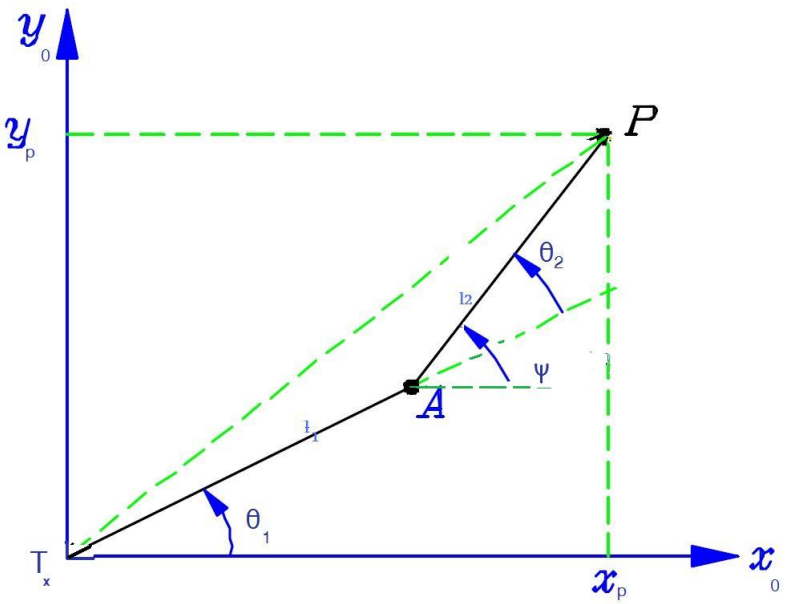

Figura 2.2: schema cinematico in pianta e definizione dell'asse Ψ.

Confrontando le due matrici T_{3A} elemento per elemento si pongonono sedici equazioni le cui soluzioni ci permettono di trovare i valori di $\vartheta_1 \quad \vartheta_2 \quad T_x$ richiesti : Da queste equazione abbiamo scartato le equazioni identicamente nulle e quelle non indipendenti conoscendo le coordinate dell'end-effector così esistono 2 configurazioni del manipolatore per arrivare nella posizione desiderata. 2.3)

Da queste considerazioni ci ritroviamo alla fine a due equazioni con tre incognite da risolvere:

$$\begin{cases} x = L_2(c_1c_2 - s_1s_2) + L_1c_1 + T_x \\ y = L_2(s_1c_2 + s_2c_1) + L_1s_1 \end{cases} \tag{2.1}$$

Dato che la base del carrello non è fissa abbiamo scelto di fissare in modo opportuno i valori di T_x che corrisponde alla posizione del carrello asseconda delle coordinate dell'end-effector questo argomento è trattato integralmente nel capitolo 3 successivo. pertanto è sempre valido l'affermazione che ad ogni valori T_x scelto corrispondono due possibili configurazioni del manipolatore.

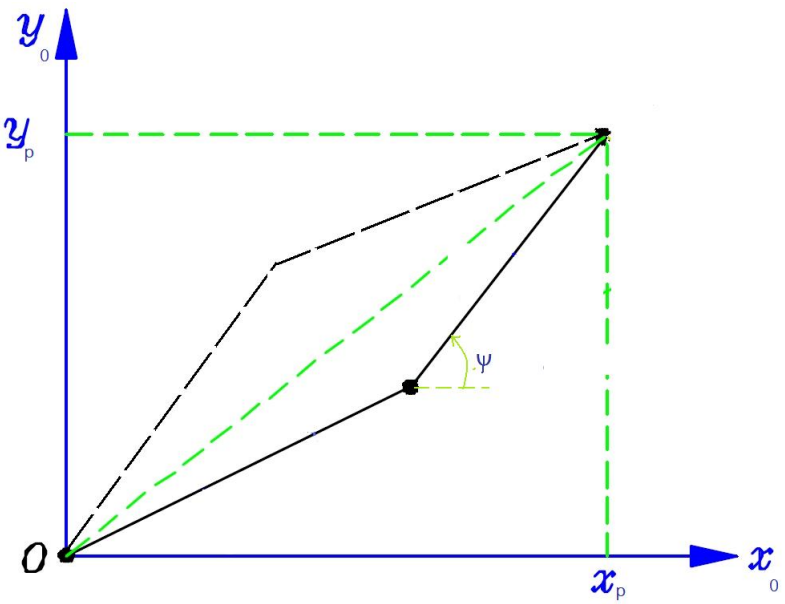

Figura 2.3: le due soluzioni del problema cinematico inverso ad ogni posizione del carrello

Calcolo della cinematica inversa

Si tratta quindi di determinare le posizione degli assi del manipolatore fissato una posizione dell'end effector del manipolatore In questo caso con sideriamo di lavorare nel sistema di riferimento (1).Esistono molti approcci per il calcolo del problema cinematico inverso noi abbiamo applicato il seguente approccio :
Si osservi la figura 2.4)

Applicando il teorema di Carnot al triangolo OPA, vale la seguente equazione:

$$OP^2 = OA^2 + AP^2 - 2OAAP\cos(\pi - \vartheta_2)$$

ovvero

$$x_p^2 + y_p^2 = l_1^2 + l_2^2 - 2l_1 l_2 \cos(\pi - \vartheta_2)$$

L'angolo ϑ_1 si ricava per differenza:

$$\begin{cases} \cos(\pi - \vartheta_2) = -\frac{x_p^2 + y_p^2 - l_1^2}{2l_1 l_2} \\ \vartheta_1 = P\hat{O}x - P\hat{O}B \end{cases} \tag{2.2}$$

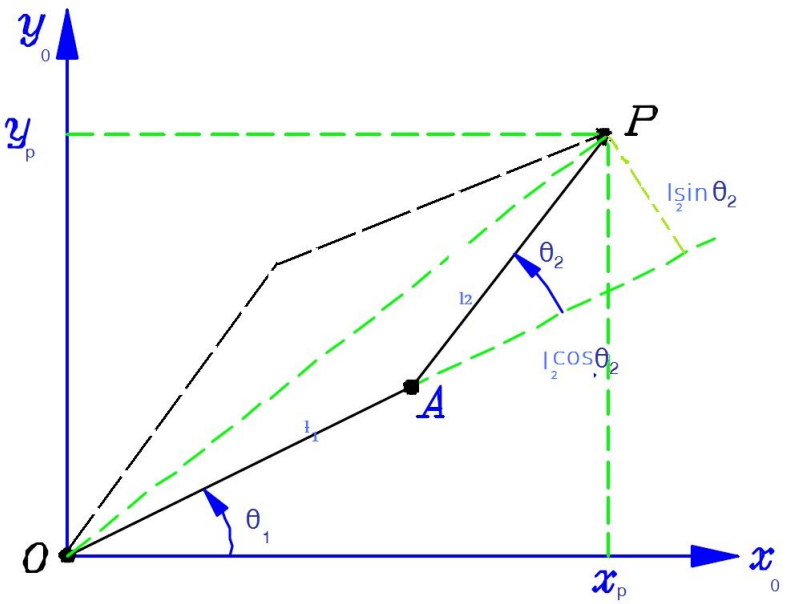

Figura 2.4: cinematica inversa manipolatore in posizione Tx

$$\begin{cases} \vartheta_2 = \pm \arccos(-\frac{x_p^2+y_p^2-l_1^2}{2l_1l_2}) \\ \vartheta_1 = a\tan 2(y_p, x_p) - a\tan 2(l_2\sin(\vartheta_2), l_1 + l_2\cos(\vartheta_2)) \end{cases} \tag{2.3}$$

La funzione "atan2" è la tangente inversa a 4 quadranti. L'equazione che consente di pervenire all'angolo ϑ_2 presenta due soluzioni a causa della funzione"arccos()"; per ciascuna delle due configurazione si ottiene una corrispondente soluzione di α.

2.2.1 Implementazione cinematica inversa con Matlab

La funzione cinIn permette di calcolare la cinematica inversa del manipolatore Essa ha come ingresso le variabili r,XY e gomito :

- dove r rappresenta La struttura del robot che contiene la posizione T_x del carrello rispetto alla terna (A)
- XY coordinate dell'end effector rispetto all'origine della terna (A).
- gomito rappresenta la configurazione del gomito che è '0' se il manipolatore deve arrivare sul punto con gomito sinistro e '1' con quello destro

Come uscita :

- il vettore alfa che contiene le posizioni degli assi (th_1)e (th_2) nei giunti.

Nel codice della funzione di cinIn ;prima di calcolare le diverse posizioni degli assi nei giunti rotoidali corrispondendo ad ogni coordinate dei vari punti selezionati ; abbiamo trovato il valore finale Tx con il metodo di ottimizzazione poi abbiamo riportato le coordinate dell'end-effector nel riferimento della terna (1) in modo da lavorare come se il manipolatore fosse a solo due gradi di libertà .

```
Xp = XY(:,1)-(r.Tx);
Yp = XY(:,2)-(r.B(2)/2+r.G);
```

2.3 Spazio operativo e spazio dei giunti

2.3.1 Spazio dei giunti

Per la pianificazione delle traiettorie nello spazio dei giunti si vuole specificare l'andamento desiderato per la posizione, la velocità e l'accelerazione dei singoli giunti . Quindi interessa solo che gli assi si portino da una posizione iniziale ad una finale (e non ha interesse il movimento risultante nello spazio operativo)
lo scopo di questa operazione è di generare una funzione q(t) di posizione in funzione del tempo per ogni giunto del manipolatore che interpola i valori assegnati per le variabili di giunto, rispettando i vincoli imposti che assicurano il posizionamento corretto del l'end effector nel tempo prestabilito. Nel nostro caso questi vincoli riguardano la posizione del punto desiderato da raggiungere fuori dal limite dallo spazio raggiungibile del manipolatore.

Moto punto-punto

Per questo tipo di moto è lasciato all'utente di specificare un punto estremo che rappresenta la posizione finale del manipolatore rispetto a quello attuale considerato come quella iniziale. Il tempo di transito da una posizione all'atra viene calcolato con l'algoritmo di ottimizzazione esposto nel capitolo 3.

Nel codice Matlab la funzione trajj11 permette di definire tutte le posizione ,le velocità ed accelerazione di ogni link in funzione del tempo ed ha come ingressi i parametri sulla posizione iniziale e quella finale di uno giunto del manipolatore;sul tempo di accelerazione e decelerazione desiderato e il tipo di legge di moto da effettuare. in figura 2.5) viene visualizzato la posizione del carrello ad ogni istante trovati con il moto punto-punto effettuato dal manipolatore da una posizione iniziale ad una finale.

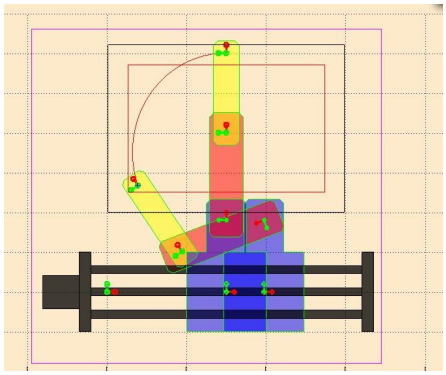

Figura 2.5: robot dopo aver effettuato un moto da un punto iniziale ad uno finale

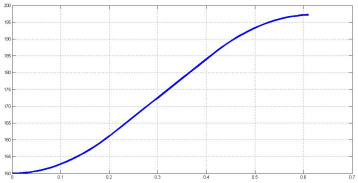

Figura 2.6: posizione del Carrello ad ogni istante

Figura 2.7: Moto Punto punto

Moto sul percorso assegnato

In questo caso viene anche specificato dei punti intermedi che vengono poi interpolati mediante dei polinomi.

- **Interpolazione mediante polinomi**
  Il problema di determinare una traiettoria che passi per n punti può essere risolto in modo univoco adottando una funzione polinomiale di grado n-1,

del tipo:

$$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + ...a_{n-1} t^{n-1}$$

Dati i valori t_i q_i $i = 1...n$ si costruiscono i vettori q, a e la matrice T (di Vandermonde) come :

$$q = \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_{n-1} \\ q_n \end{bmatrix} = \begin{bmatrix} 1 & t_1 & \dots & t_1^{n-1} \\ 1 & t_2 & \dots & t_2^{n-1} \\ & & \vdots & \\ 1 & t_{n-1} & \dots & t_{n-1}^{n-1} \\ 1 & t_n & \dots & t_n^{n-1} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \\ a_{n-2} \end{bmatrix} = T_a \qquad (2.4)$$

ne consegue che $a = T^{-1}q$ la matrice T è sempre invertibile se $t_i > t_{i-1}$ con $i = 1 \dots$n

Un vantaggio evidente dell'interpolazione polinomiale è che la funzione q(t) ha derivate continue di ordine qualsiasi all'interno dell'intervallo [t1 tn]. Tuttavia il metodo non è efficiente dal punto di vista numerico: all'aumentare del numero n di punti aumenta il numero condizionante k (rapporto tra il massimo ed il minimo valor singolare) della matrice T di Vandermonde, rendendo il problema della sua inversione mal condizionato numericamente.

Esistono anche altri metodi, più efficienti, per calcolare i coefficienti del polinomio, ma le difficoltà numeriche permangono per valori elevati di n.

Anche prescindendo dalle difficoltà numeriche, l'interpolazione di n punti mediante un unico polinomio di grado n-1 presenta degli svantaggi:

1. il grado del polinomio dipende da n e, per elevati valori di n, la quantità di calcoli da eseguire può essere notevole;
2. la variazione di un solo punto (t_i, q_i) implica il ricalcolo dell'intero polinomio;
3. l'aggiunta di un punto finale (t_{n+1} $, q_{n+1}$) implica l'utilizzo di un polinomio di grado maggiore ed il ricalcolo di tutti i coefficienti
4. la soluzione che si ottiene presenta in generale oscillazioni indesiderate

Un'alternativa è, anziché considerare un unico polinomio di grado n-1, utilizzare n-1 polinomi di grado p (tipicamente inferiore), ognuno dei quali definito in un tratto della traiettoria. Il grado p dei polinomi è normalmente preso uguale a 3 (tratti di traiettoria cubica). Una primo, ovvio, modo di procedere consiste nell'assegnare posizioni e velocità in tutti i punti e calcolare i coefficienti delle cubiche tra due punti consecutivi.

Spline

L'interpolazione mediante cubiche genera una traiettoria che presenta accelerazione discontinua nei punti di passaggio. Per ovviare a questo problema, sempre mantenendo interpolanti cubiche, si deve rinunciare ad imporre specifici valori di velocità nei punti intermedi, limitandosi ad imporre la continuità in due tratti contigui di posizione, velocità ed accelerazione.
in figura 2.8) viene visualizzato la posizione del carrello ad ogni istante trovati con il moto punto-punto effettuato dal manipolatore da una posizione iniziale ad una finale.

La traiettoria che si ottiene con questo procedimento prende il nome di spline (smooth path line).

Si può dimostrare che la spline è la funzione interpolante a curvatura minima, a parità di condizioni di continuità sulle derivate.

- **Spline: condizioni da imporre**
  Poiché con n punti si hanno n-1 polinomi del tipo:

$$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$

  ciascuno dei quali ha 4 coefficienti, il numero totale di coefficienti da calcolare è $4(n-1)$. Le condizioni da imporre sono:

  - $2(n-1)$ condizioni di passaggio per punti (ogni cubica deve interpolare i punti alle sue estremità);
  - $n-2$ condizioni sulla continuità delle velocità nei punti intermedi

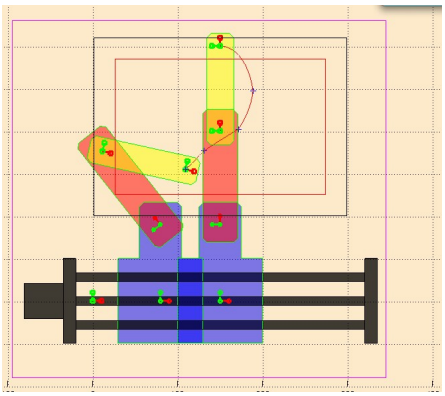

Figura 2.8: robot dopo aver effettuato un moto su più punti

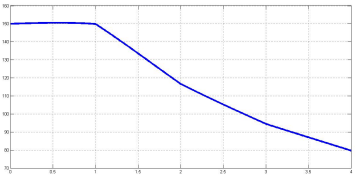

Figura 2.9: posizione del Carrello ad ogni istante

Figura 2.10: spline cubiche

– $n-2$ condizioni sulla continuità delle accelerazioni nei punti intermedi

Si hanno quindi:

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

gradi di libertà residui.

Una modalità (non unica) di utilizzare questi 2 gradi di libertà consiste nell'assegnare opportune condizioni iniziali e finali sulla velocità.

- **Spline: posizione analitica del problema** Si desidera determinare una funzione:

$$q(t) = \{q_k(t), \quad t \in [t_k, t_{k+1}], k = 1 \dots n-1\}$$

$$q(\tau) = a_{k0} + a_{k1}\tau + a_{k2}\tau^2 + a_{k3}\tau^3$$

Con le condizioni :

$$q_k(0) = q_k, q_k(T_k) = q_{k+1} \quad k = 1 \dots n-1$$

$$\dot{q}_k(0) = \dot{q}_k, q_k(T_k) = v_{k+1} \quad k = 1 \dots n-2$$

$$\ddot{q}_k(0) = \ddot{q}_k, \qquad k = 1 \dots n-2$$

$$\dot{q}_1(0) = v_1, \dot{q}_{n-1}(T_{n-1}) = v_n$$

dove le quantità v_k $k = 1 \dots n-1$non sono specificate.
Il problema consiste nel ricavare i coefficienti a_{ki}

- **Spline: algoritmo** Si assumano inizialmente note le velocità v_k, $k = 2 \dots n-1$ nei punti intermedi. In questo modo, per ogni polinomio cubico si hanno quattro condizioni al contorno su posizione e velocità, che danno origine al sistema:

$$\begin{cases} q_k(0) = a_{k0} = q_k \\ \dot{q}_k(0) = a_{k1} = v_k \\ q(T_k) = a_{k0} + a_{k1}T_k + a_{k2}T_k^2 + a_{k3}T_k^3 = q_{k+1} \\ \dot{q}(T_k) = a_{k1} + 2a_{k2}T_k + 3a_{k3}T_k^2 = v_{k+1} \end{cases} \tag{2.5}$$

che risolto dà :

$$\begin{cases} a_{k0} = q_k \\ a_{k1} = v_k \\ a_{k2} = \frac{1}{T_k}\left[\frac{3(q_{k+1}-q_k)}{T_k} - 2v_k - v_{k+1}\right] \\ a_{k3} = \frac{1}{T_k^2}\left[\frac{2(q_k-q_{k+1})}{T_k} + v_k + v_{k+1}\right] \end{cases} \tag{2.6}$$

Naturalmente le velocità v_k, $k = 2 \dots n-1$ vanno calcolate. Imponiamo la continuità dell'accelerazione nei punti intermedi:

$$\ddot{q}_k(T_k) = 2a_{k2} + 6a_{k3}T_k = 2a_{k+1,2} = \ddot{q}_{k+1}(0), k = 1 \dots n-2$$

Sostituendo le espressioni per i termini $a_{k2}, a-k3, a_{k+1,2}$e moltiplicando per $(T_kT_{k+1})/2$ si ottiene:

$$T_{k+1}v_k + 2(T_{k+1}+T_k)v_{k+1} + T_kv_{k+2} = \frac{3}{T_kT_{k+1}}\left[T_k^2(q_{k+2}-q_{k+1}) + T_{k+1}^2(q_{k+1}-q_k)\right]$$

Mettendo in forma matriciale si ottiene un equazione del tipo : $Av = c$ dove le costanti c_k del vettore c dipendono solo dalle posizioni intermedie e dalle durate dei segmenti, grandezze note.

- La matrice A è a struttura dominante diagonale e risulta sempre invertibile per $T_k > 0$.
- Inoltre la matrice A è a struttura tridiagonale, per cui esistono tecniche numeriche efficienti (metodo di Gauss-Jordan) per la sua inversione.
- Una volta nota l'inversa di A si possono calcolare le velocità $v_2 \dots v_{n-1}$ come:

$$v = A^{-1}c$$

il che risolve completamente il problema.

Abbiamo implementato questo algoritmo in Matlab con le funzioni :

- percorso che permette di selezionare piu punti da seguire usando la funzione ginput dentro un ciclo while che ne esce se e solo se viene premuto il tasto destro del mouse.
- pos-splin ha come ingresso i coefficienti di ogni polinomi cubici e ritorna indietro tutte le posizione da effettuare dall'end-effector
- splin-point ritorna i coefficienti dei vari polinomi cubici.

2.3.2 Spazio operativo

si definisce il percorso dell'organo terminale del manipolatore nel comune spazio cartesiano.
Occorre in ogni caso uno stadio di inversione cinematica per passare allo spazio di attuazione. In questo caso abbiamo preso anche in conto il movimento che effettua la base prima abbiamo trovato le coordinate dei punti che descrivono ogni traiettorie considerando il movimento solo nello spazio dei giunti rotoidali e di seguito prima di risolvere il problema cinematico inverso abbiamo trovato tutte le posizioni del carrello da una posizione iniziale a d una finale usando la

funzione di ottimizzazione così il movimento del manipolatore coinvolge tutti gli assi.

Codice per definizione il moto in linea retta

Un segmento nello spazio operativo effettuato dai bracci viene definita come :

$$p(s) = p_i + \frac{s}{\|p_f - p_i\|}(p_f - p_i)$$

dove:

- p_i p_f rappresentano le coordinate dei punti iniziale e finale.
- s descrive la legge di percorrenza in funzione del tempo

. per farsi che il robot raggiunge il punto finale partendo da quello iniziale eseguendo una traiettoria rettilinea 2.11)

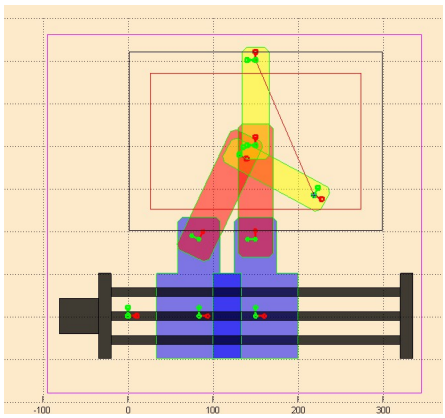

Figura 2.11: robot dopo aver effettuato una traiettoria rettilinea

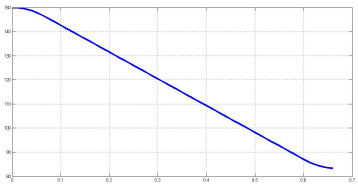

Figura 2.12: posizione del Carrello ad ogni istante

Figura 2.13: Segmento

Abbiamo creato in Matlab la funzione rett che prende in ingress: i punti iniziali e finali e la legge temporale per ogni giunto e ritorna come uscita tutti i punti

che descrivono il segmento da eseguire. Viene riportato qua sotto il codice della funzione descritta.

```
xPr = x_in(1) + ( x - x_in(1) ) * s ;
yPr = x_in(2) + ( y - x_in(2) ) * s ;
```

Per fare muovere anche la base prima abbiamo pianificato il movimento del carrello partendo da una posizione iniziale ad una ottimizzata

Codice per definizione del moto lungo una traiettoria circolare

2.14)

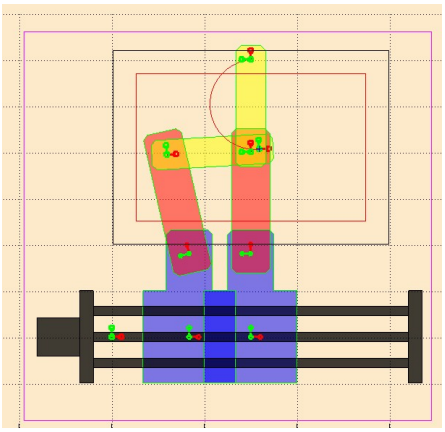

Figura 2.14: robot dopo aver effettuato una traiettoria circolare

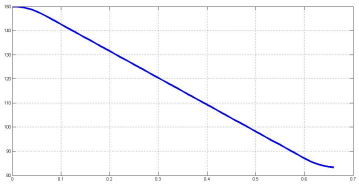

Figura 2.15: posizione del Carrello ad ogni istante

Figura 2.16: semicerchio

Supponendo che il moto avvenga su un cerchio generico di cento p_c e orientato in modo generico,è sempre possibile definire una terna con origine in p_c.indicando R la matrice di rotazione che definisce l'orientamento della terna solidale al

cerchio, le coordinate del punto che si muove su di esso sono :

$$p(s) = p_c + R \begin{bmatrix} \rho\cos(s/\rho) \\ \rho\sin(s/\rho) \\ 0 \end{bmatrix} \tag{2.7}$$

ove ρ è il raggio della circonferenza e

```
centro = (p_i + p_f)/2 ;
xPr =   centro(1) +R*(A11*cos(alfa*s) + A12*sin(alfa*s) ) ;
yPr =  centro(2)+ R*(A21*cos(alfa*s) + A22*sin(alfa*s) ) ;
```

Capitolo 3

Ottimizzazione del movimento

L'obbiettivo dell'ottimizzazione del movimento è quello di fare muovere il meno possibile i giunti rotoidali facendo lavorare di più quello prismatico.
Per potere fare questo il metodo di ottimizzazione si è basato sulla minimizzazione del tempo di movimentazione dei giunti rotoidali da una posizione iniziale a quella successiva questo viene fatto anche tenendo conto:

- dei vincoli presenti sul manipolatore cioè i diversi switch meccanici descritti in 1 che bloccano il movimento del manipolatore se un organo va oltre al limite imposto.3.1)

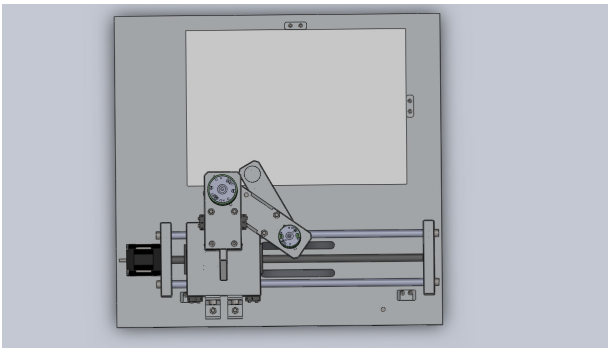

Figura 3.1: disegno 3D del manipolatore in configurazione limite.

- dalla posizione del manipolatore rispetto al punto da raggiungere .

3.1 Descrizione del metodo iterativo usato

È un algoritmo iterativo che permette di ridurre o aumentare la distanza da percorrere dal carrello asseconda che il suo tempo di movimento sia maggiore o minore del massimo di tempo scelto tra gli altri due organi quando il manipolatore si muove da una posizione iniziale ad una finale. l'algoritmo viene descritto come segue :

3.1.1 Calcolo del limite di spostamento del carrello

Per calcolare il limite di spostamento del carrello rispetto alla posizione del punto da raggiungere ;abbiamo prima riportato le coordinate del punto selezionato rispetto all'origine della terna di riferimento (1) del primo membro questo viene calcolato attraverso il codice seguente.

```
Xp = XY(1)-Xr;
Yp = XY(2)-r.T1A(2,4);
```

ove :

1. XY rappresenta le coordinate del punto rispetto alla terna di riferimento fisso (A) della guida lineare.

2. X_r è la posizione del robot cioè del carrello rispetto all'asse fissa da notare poi che corrisponde anche a quella della terna (1).

3. $r.T_{1A}$ è la matrice di posizione della terna(1) rispetto alla terna (A)

Poi abbiamo calcolato le posizioni massime e minime che occuperebbe il manipolatore per arrivare al tale punto (figura 3.2)

$$xmax = x_{pt} + \sqrt{(L_1 + L_2)^2 - y_{pt}^2}$$

$$xmin = x_{pt} - \sqrt{(L_1 + L_2)^2 - y_{pt}^2}$$

formula implementato con Matlab usando il codice:

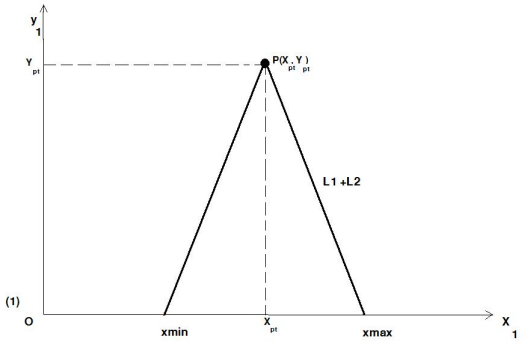

Figura 3.2: posizioni dei massimi e minimi trovati rispetto alla terna (A).

```
L_stesi = r.L(1)+r.L(2);
dist = sqrt((L_stesi)^2-Yp^2);
% calcolo limiti di spostamento per la base
xmax = XY(1)+dist;
xmin = XY(1)-dist;
```

In cui :

1. L_stesi lunghezza massima del braccio esteso

2. x_{max} x_{min} posizione massima e minima limite del carrello rispetto alla terna (A) di riferimento fisso per cui il manipolatore arriva sul punto con il braccio esteso (vedere figura 3.2) .

Data la presenza dei blocchi di lettura degli switch meccanici sulla guida lineare ,abbiamo fatto un reset dei massimo e minimo trovato considerando il vincolo imposto sul movimento della base questo è stato fatto facendo il controllo sulla posizione dei massimo e minimo trovati. Condizione implementato attraverso il seguente codice.

```
% reset del minimo
if xmin<r.XLIM(1)
    xmin = r.XLIM(1);
end
% reset del massimo
if xmax>r.XLIM(2)
```

```
    xmax = r.XLIM(2);
end
```

1. la prima condizione è quella di verificare se la posizione minima trovata è minore del limite consentito per il movimento del carrello lungo la guida cioè verificare se il carrello si troverebbe prima dello switch presente al lato sinistro della guida se succedesse questo allora viene assegnato come posizione minima quella dello switch r.XLIM(1) così per essere sicuri che il manipolatore non andrebbe oltre il limite previsto in modo da evitare che viene staccata l'alimentazione dei motori.

2. la seconda invece riguarda la posizione massima calcolata e verifica se questa è maggiore della posizione del secondo switch r.XLIM(2) posto alla destra della guida. Se fosse il caso allora viene assegnato come valore massimo quella posizione. vedere figura 3.5

3.1.2 Calcolo distanza minima-massima e distanza punto selezionato e dell'end-effector

La distanza minima o massima rappresenta la distanza del robot dal valore massimo o minimo calcolato prima .Questa distanza si rivelerà utile di seguito per verificare quanto lontano o vicino si trova il carrello dalla posizione del punto da raggiungere sull'asse delle ascisse cioè la distanza da effettuare dal manipolatore per arrivare al punto .

$$\begin{aligned} dxmin &= xmin - Xr; \\ dxmax &= xmax - Xr; \end{aligned} \tag{3.1}$$

Le distanze sia del punto selezionato che dell'end-effector rispetto alla terna (1) di riferimento del manipolatore vengono calcolati come segue :

$$d_p = \sqrt{X_p^2 + Y_p^2}$$

$$d_{patt} = \sqrt{X_{patt}^2 + Y_{patt}^2}$$

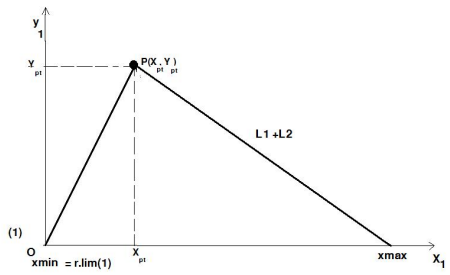

Figura 3.3: posizione minima

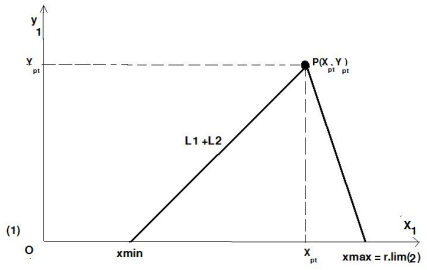

Figura 3.4: posizione massima

Figura 3.5: posizioni massimi e minime del carrello corrispondente ai limiti di movimentazione sulla guida

ove X_p Y_p sono le coordinate del punto e X_{patt} =r.T31(1,4) Y_{patt} =r.T31(2,4); quelli dell'end-effector; tutti calcolati rispetto all'origine della terna (1). Il codice in Matlab per calcolare tale distanze è il seguente: Il confronto tra questi due valore permetterà di seguito di sapere se si deve o non incrementare il valore di posizione del carrello per arrivare al punto.

3.1.3 Insiemi che vincolano il movimento del carrello lungo la guida

Gli insiemi considerati come lo mostrano la figura 3.8 sono 4 tra cui due quarti di cerchi e due rettangoli da ambe i due lati nello spazio di lavoro.Se il punto selezionato appartiene ad uno di questi ;il carrello rimarrà fermo o potrà muoversi

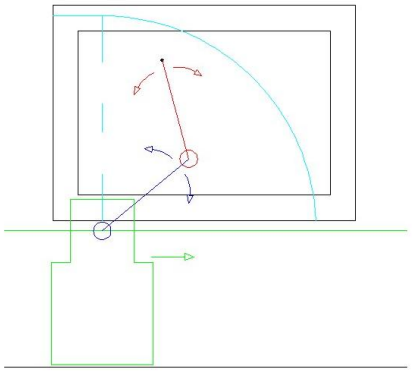

Figura 3.6: insieme di sinistra

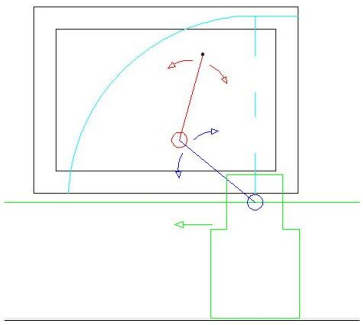

Figura 3.7: insieme di destra

Figura 3.8: posizioni massimi e minime del carrello corrispondente ai limiti di movimentazione sulla guida

solo da una parte destra o sinistra. ricordiamo che l'analisi di questi insiemi è dovuto a alla presenza sulla guida lineare dei blocchetti fissi che leggono gli switch meccanici presenti sul carrello che impongono un limite di movimentazione del carrello che ci hanno costretti a fare i reset dei massimi e minimi calcolati in funzione della posizione del punto da raggiungere. La verifica che un punto appartiene o non a uno di quelli insieme viene fatto attraverso le variabili con valore '1' se il punto appartiene a tale insieme altrimenti '0'.

punto si trova nel quarto di cerchio o nel rettangolo sinistro

Dato che il valore minimo viene assegnato al quello limite se $xmin < r.LIM(1)$ come spiegato nel paragrafo precedente.
Tutti I punti per cui il valore minimo risulta minore di quello limite formano normalmente tutti l'interno del cerchio con caratteristiche

$$\begin{cases} R = L_1 + L_2 \\ c = [XLIM(1), \quad B_2/2 + G] \end{cases} \tag{3.2}$$

Dove le coordinate del centro 'c' costituiscono quelle dell'origine della terna (1) rispetto alla terna(A) quando il bordo sinistro del carrello corrisponde alla posizione del corrispondente blocco di lettura sinistro sulla guida. Però considerando lo spazio di scrittura cioè l'altezza del foglio nella nostra applicazione l'insieme considerata si riduce a quello in figura 3.9 Dall'analisi di questi insiemi abbiamo

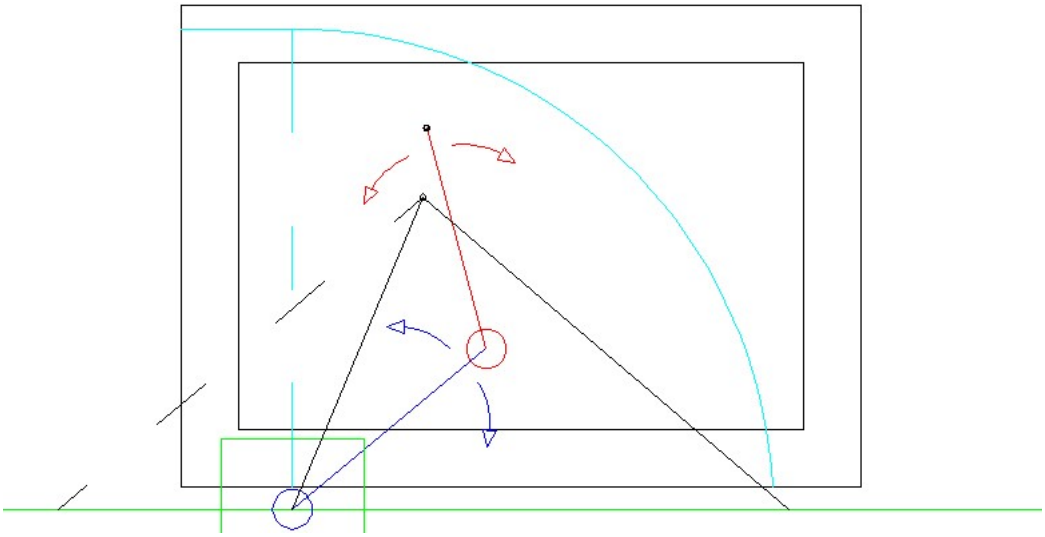

Figura 3.9: punto si trova nel quarto di cerchio o nel rettangolo sinistro rappresentazione del minimo risultante rispetto a quello che dovrebbe essere.

imposto che in questa situazione il carrello starà fermo o si muoverà solo verso destra per evitare di oltrepassare il limite segnato;cioè incrementeremo il valore di posizione del carrello .
Il codice che permette di verificare se il punto appartiene o no a quest'insieme è il seguente:

```
\footnotesize
\begin{verbatim}
 ----insiemi considerati----
%1)punto si trova nell'arco di sinistra
    % cerchio di sinistra
 cer_sin =  sqrt((XY(1)-r.XLIM(1))^2+(XY(2)-(r.B(2)/2+r.G))^2);
if((cer_sin <= L_stesi)&& (XY(1) >=r.XLIM(1)))
    Arc_sin = 1;
else
     Arc_sin = 0;
end

%3) punto si trova o no nel rettangolo di sinistro
 if(XY(1) <=r.XLIM(1))
     rett_sin = 1;
 else
      rett_sin = 0;
 end
```

punto si trova nel quarto di cerchio destro o nel rettangolo di destro

Quest'insieme risulta simmetrica a l'insieme trovato prima e viene considerato in questo caso quando il valore massimo trovato risulta maggiore della posizione dello switch meccanico presente alla destra della guida 3.10

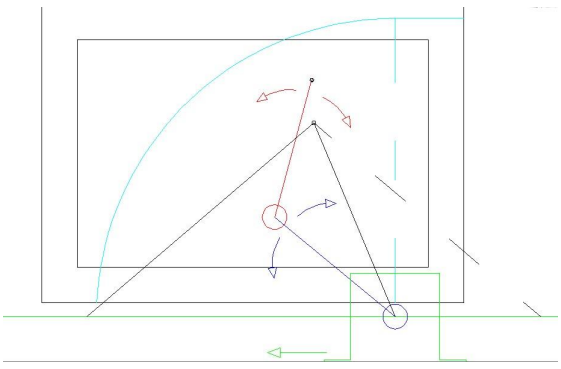

Figura 3.10: punto si trova nel quarto di cerchio o nel rettangolo destro rappresentazione del massimo risultante rispetto a quello che dovrebbe essere

.

```
   % cerchio di destra
   cer_des =  sqrt((XY(1)-r.XLIM(2))^2+(XY(2)-(r.B(2)/2+r.G))^2);
if((cer_des <= L_stesi)&& (XY(1) <=r.XLIM(2)))
    Arc_des = 1;
else
     Arc_des = 0;
end

%4) nel rettangolo di destra
 if(XY(1) >=r.XLIM(2))
      rett_des = 1;
 else
      rett_des = 0;
 end
```

3.1.4 Calcolo della distanza iniziale e valore iniziale d'iterazione

La posizione iniziale x_{in}che consideriamo dipende dalla posizione del carrello dal punto finale:

1. se il carrello è alla destra del punto cioè

$$d_{xmax} = x_{max} - X_r \leq 0 \quad \Longrightarrow \quad x_{in} = x_{max}$$

2. se il carrello è alla sinistra del punto cioè

$$d_{xmin} = x_{min} - X_r \geq 0 \quad \Rightarrow \quad x_{in} = x_{min}$$

   .

3. invece se carrello si trova tra i valori limiti

$$x_{min} < X_r < x_{max} \quad \Rightarrow \quad x_{in} = X_r$$

   .

Dalla posizione in cui il manipolatore si trova rispetto a quello da raggiungere abbiamo la possibilità di avvicinarsi o allontanarsi al punto asseconda del valore

d_p rispetto a d_{patt} .
Durante quest'analisi ID rappresenta il fattore di incremento o decremento se $ID = 0$ si sta incrementando e $ID = 1$se si sta decrementando.
Il valore della variabile 'a' che rappresenta la distanza da iterare partendo dal punto iniziale ci permette di valutare la distanza da percorrere dal carrello andando verso sinistra o destra una volta trovata la posizione iniziale xin .La variabile 'a' viene calcolato in funzione degli insiemi analizzati prima in cui si trova il punto selezionato.

Posizione carrello a sinistra della posizione minima calcolata

$$d\_xmin = x\_min - x\_r > 0$$

Risulta ovvio che in questo caso la distanza del punto rispetto alla terna(1) è maggiore di quella dell'end-effector

$$d_p > d\_patt$$

quindi il robot deve muoversi verso il punto cioè incrementare (ID = 0) il valore della sua posizione attuale X_r Il valore iniziale d'iterazione e la posizione iniziale del robot vengono calcolati verificando l'appartenenza del punto all'insieme rettangolo destra . Questo perchè la posizione del carrello potrebbe superare il limite destro presente sulla guida se la distanza viene incrementata di troppo. 3.11

Spiegazione delle tappe del codice.

1. punto non appartiene al rettangolo di destra. possiamo muoverci senza rischio di superare il limite imposto ponendo il valore d'iterazione $a = x_{min} - XY(1)$ così da muovere al massimo il carrello alla posizione XY(1) del punto e come distanza iniziale $x_{in} = x_{min}$ scelta in base alle impostazioni di prima.

2. se il punto appartiene al rettangolo di destra il valore massimo viene modificato e è uguale al limite destro $x_{max} = r.Lim(2)$. abbiamo due casi da verificare:

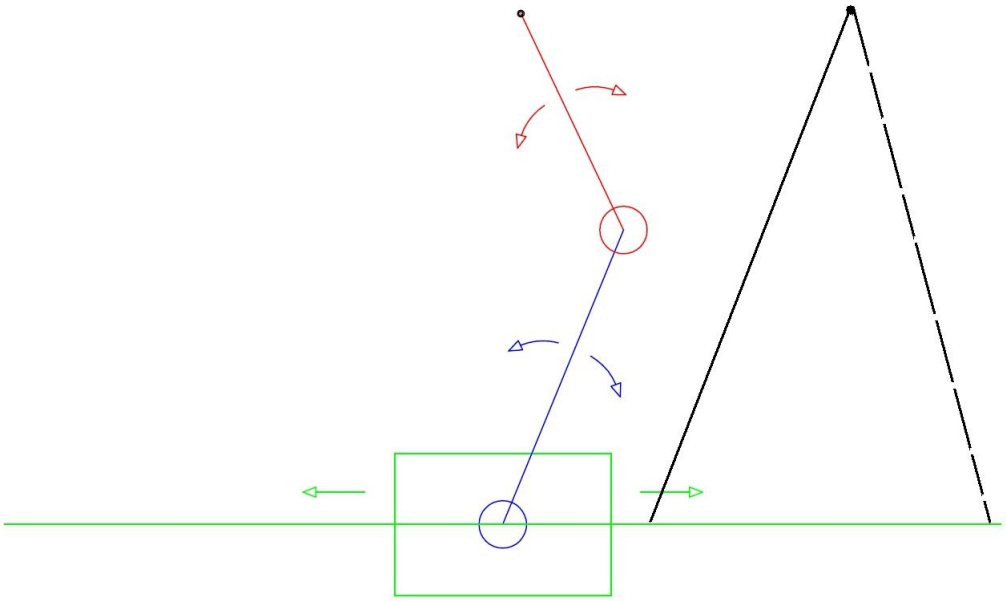

Figura 3.11: Posizione carrello a sinistra della posizione minima .

- se $x_{min} \geq x_{max}$ viene assegnato al valore iniziale d'iterazione $a = 0$ in modo che il carrello non si muove dalla posizione iniziale trovata ponendo la distanza iniziale a $x_{in} = x_{max}$ limite per non sorpassare il blocco di destra.
- altrimenti la posizione iniziale risulta normalmente essere come imposto per il funzionamento normale. $x_{in} = x_{min}$ però la posizione massima che può raggiungere il carrello è x_{max} dato che $a = x_{max} - x_{min}$.

Viene riportato qua sotto il codice Matlab che implementa la parte appena argomentata

```
-------- posizione del carrello rispetto al punto selezionato
 carrello è tutto alla sinistra del valore minimo
 indice di incremento (0) o decremento (1)
if dxmin > 0
    ID=0;
    if (rett_des == 0) % punto non è sul rettangolo di destra
        a = abs(XY(1)-xmin);
        xin = xmin;
    else % punto sul rett destra
        if (xmin>=xmax)
```

```
            xin = xmax;
            a = 0 ; % carello non si muove
        else
            xin = xmin;
            a = xmax-xmin; % carello si muove al massimo fino a xmax
        end
    end
```

Posizione carrello maggiore della posizione massima calcolato

$$d\_xmax = x_{max} - x\_r < 0$$

Come nel caso precedente risulta anche che :

$$d_p > d\_patt$$

Però il valore di posizione del carrello verrà decrementato(ID=1) per portare il robot sul punto che si trova alla sinistra della posizione del carrello . Come già osservato prima dato che gli insiemi trovati prima sono simmetriche il raggionamento fatto per il carrello alla sinistra del punto è lo stesso che viene usato in quest'analisi con l'unico differenza che per il calcolo dei valori valori x_{in} e a vengono sostituiti a posto di xmin il valore xmax e vice-versa .come lo mostra il seguente codice :

```
 % carello è tutto alla destra del valore massimo
elseif dxmax < 0
    ID = 1 ;
    if (rett_sin == 0) % punto non è sul rettangolo di sinistra
        a = abs(XY(1)-xmax);
        xin = xmax;
    else % punto sul rett sin
        if (xmin>=xmax)
            xin = xmin;
            a = 0 ; % carello non si muove
        else
            xin = xmax;
            a = xmax-xmin; % carello si muove al massimo fino a xmin
        end
    end
```

Posizione carrello uguale al valore massimo trovato

$$d\_xmax = xmax - x_r = 0$$

In questa posizione risulta evidente che per potere raggiungere il punto prima di tutto porre senz'altro $x_{in} = x_{max}$ dalle considerazioni fatte all'inizio del paragrafo e poi per evitare di arrivare sul punto con il braccio steso , dobbiamo evidentemente spostare il carrello verso esso quindi decrementare il valore della sua posizione $(ID = 1)$.
Il valore iniziale 'a' dipende anche in questo caso dalla posizione del punto sul foglio di lavoro

- se il punto appartiene all'insieme arco-rettangolo di destra .
  Ricordiamo che in questo caso abbiamo $x_{max} = r.Lim(2)$ quindi abbiamo due possibilità:
  la prima è quello di tenere fermo il manipolatore se la distanza $d_p > d_{patt}$ cioè $a = 0$. farsi che la posizione finale del manipolatore corrisponde al limite destro per non andare oltre al valore imposto.
  altrimenti la distanza $d_p < d_{patt}$ muoverlo verso sinistra allontanandosi con $a = abs(XY(1) - xmax)$;.

- se invece il punto appartiene al rettangolo di destra , il manipolatore sta fermo se il punto sta alla destra e la sua distanza $d_p > d_{patt}$;In questo caso poniamo $a = 0$ in modo che la posizione finale corrisponde a quella iniziale trovata se il valore massimo è minore di quello minimo $xmin > xmax$ altrimenti $a = xmax - xmin$.

3.12

```
  %posizione carello coincide con quello del valore massimo
elseif dxmax ==0
    xin = xmax;
    ID = 1;
    if((Arc_des == 1)||(rett_des ==1)) %punto nel acro dest
        if(dp > dpatt)
        a = 0;  %non si muove se dp< dpatt
```

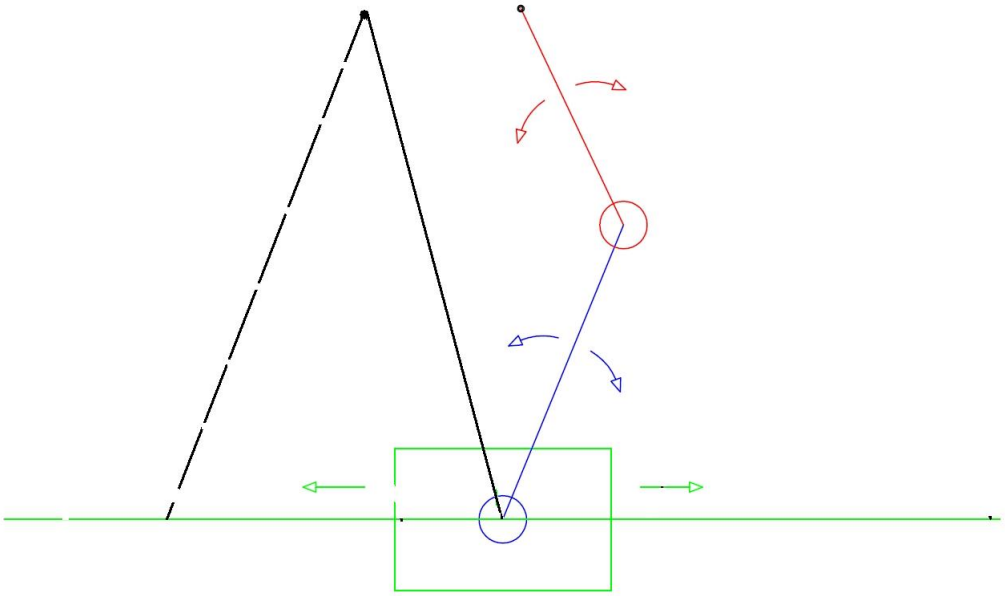

Figura 3.12: Posizione carrello uguale al valore massimo trovato.

```
    else % va verso il punto se dp > dpatt
         a =abs(XY(1)-xmax);
    end
elseif(rett_sin ==1) % nel rett dest.
    if(xmin>=xmax)
        a = 0;  %non si muove
    else
        a = xmax-xmin; % carello si muove al massimo fino a xmin
    end
else
    a = Xr-XY(1);
end
```

Posizione carrello uguale al valore minimo trovato

Abbiamo svolto questo caso in maniera analogica al precedente dato che i due insiemi sono simetriche vengono solo variati le variabili xin e 'a'. come descritto dal seguete codice

```
%posizione carello coincide con quello del valore minimo
elseif dxmin == 0
    xin = xmin;
```

```
ID = 0;
if((Arc_sin == 1)||(rett_sin ==1)) %punto nel sin sin
    if(dp < dpatt)
    a = 0;  %non si muove se dp< dpatt
    else % va verso il punto se dp > dpatt
         a =abs(XY(1)-xmin);
    end

elseif(rett_des ==1) % nel rett dest.
    if(xmin>=xmax)
        a = 0;  %non si muove
    else
        a = xmax-xmin; % carello si muove al massimo fino a xmax
    end
else
    a = XY(1)-Xr;
end
```

Posizione carrello tra valore minimo e massimo trovato

La posizione iniziale del manipolatore viene assegnate a quella attuale $x_{in} = X_r$ In questa parte viene prima di tutto confrontati le distanze del d_pe d_{patt} per decidere se incrementare o no la posizione del carrello vedi 3.13.

- Se la distanza del punto d_p è maggiore a quella attuale del manipolatore d_{patt}

  $$d_p > d_{patt}$$

  Caso in cui il manipolatore deve avvicinare al punto per farsi che il braccio si apre il meno possibile il punto : risulta allora che se il carrello si trova alla sinistra del punto dobbiamo spostarsi verso la destra cioè incrementare la distanza $ID = 0$ altrimenti $ID = 1$ viene decrementata

- distanza del punto d_p è minore di quella attuale del manipolatore d_{patt}

  $$d_p < d_{patt}$$

manipolatore si deve allontanare dal punto per potere raggiungerlo senza dovere chiudere troppo il braccio. Asseconda della posizione del carrello rispetto al punto invece abbiamo $D = 1$ se carrello a destra del punto altrimenti $ID = 1$

Visto che incrementare o decrementare di troppo la distanza del carrello può portarci a sfiorare i limiti imposti; il valore di 'a' dipende a questo punto dall'insieme in cui il punto selezionato si trova.

1. Se si sta incrementando c'è il rischio di portarci oltre il limite destra per evitare che ciò accadesse il valore 'a' da iterare è trovato in funzione delle posizioni del punto
   Se il punto non si trova sul l'insieme del rettangolo destro possiamo semplicemente porre $a = XY(1) - Xr$ senza rischio.
   Altrimenti si pone $a = dxmax$

2. se si sta decrementando si rischia di portarsi oltre il valore limite sinistro per questo motivo analogamente a quanto fatto prima abbiamo
   Se il punto non si trova sul l'insieme del rettangolo sinistro possiamo anche in questo caso porre $a = Xr - XY(1)$.
   Altrimenti si pone $a = dxmin$

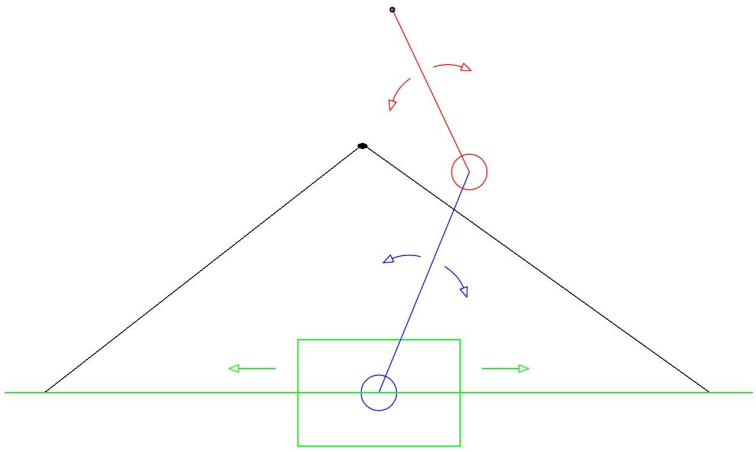

Figura 3.13: Posizione carrello tra valore minimo e massimo trovato.

```
 carrello è tra valore massimo e minimo
else
    xin = Xr;
if (dp>dpatt)
            ID = (XY(1)<Xr);% 1 se sinistra 0 se no
        else
            ID = (XY(1)>Xr);% 1 se destra 0 se no
        end
        if ID==0
         if((Xr<XY(1))&&(rett_des==0))%a destra non nei rettangoli
                a = XY(1)-Xr;
          else
            a = abs(dxmax);
          end
        else
         if((Xr>XY(1))&&(rett_sin ==0))%sinistra non nei rettangoli
                a = Xr-XY(1);
         else
            a = abs(dxmin);
            end
        end
end
```

3.1.5 La funzione iterativa : Ott_config

questa funzione ha come ingresso il robot la sua configurazione iniziale e il punto da raggiungere e ritorna come uscita la configurazione finale del manipolatore il tempo ottimo di movimentazione .Dati che vengono poi essere usati per pianificare in modo ottimo il movimento del manipolatore da una configurazione iniziale a una finale (vedi figura 3.14) . Le tappe sono:

1. calcolo della distanza minima iniziale x_{in} con le considerazioni fatte all'inizio di questo capitolo.

2. Calcola la distanza iniziale pari alla media tra quella minima e il il valore a trovato sempre con le considerazioni fatte all'inizio$x_{in} = x_{in} \pm \frac{a}{2}$

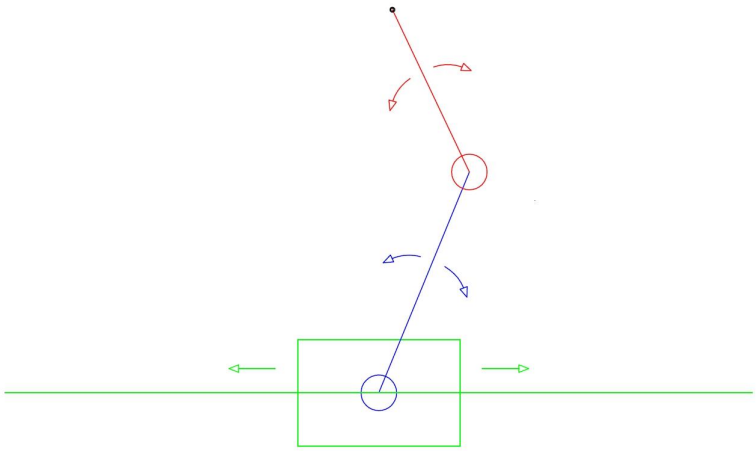

Figura 3.14: incremento o decremento pos carrello per arrivare alla posizione ottima.

3. si sceglie la posizione del manipolatore come la posizione iniziale calcolata $r.Tx = x_{in}$

   - si calcola la soluzione della cinematica inversa.
     A questo punto viene fatta la verifica solo sulla posizione del secondo membro dato che il primo non supera mai in questa applicazione il limite dovuti alla presenza degli switch meccanici sul carrello che limitano il moto del primo membro ;considerando la distanza del manipolatore dal foglio utile di scrittura .
     In effetti se la distanza tra il punto e la posizione del carrello è maggiore della distanza minima necessaria per non arrivare al lo switch meccanico sul membro 2 per qualsiasi posizione del membro 1,Si apre di più il braccio aumentando o diminuendo il valore di posizione del carrello trovato; se poi aumentando questo valore ci troviamo a dovere sorpassare il limite imposto sul carrello allora in quel caso si cambia la configurazione del gomito del manipolatore.
   - si calcola i tempi di movimento per ciascun membro $T_1 \quad T_2$ e quello del carrello T_x

4. verifica se $T_x > max(T_1, T_2)$ allora Incremento x_in rispetto a quello iniziale

altrimenti ($ID = 1$) decremento.

5. poi ricalcoliamo tutto l'iterazione per 10 volte

Con questo metodo troviamo la posizione minima del carrello che ci permette ai membri di muoversi di poco rispetto alla posizione del punto desiderato.

Capitolo 4

Utilizzo dell'interfaccia utente e Comunicazione con Simulink

4.1 Implementazione interfaccia GUI

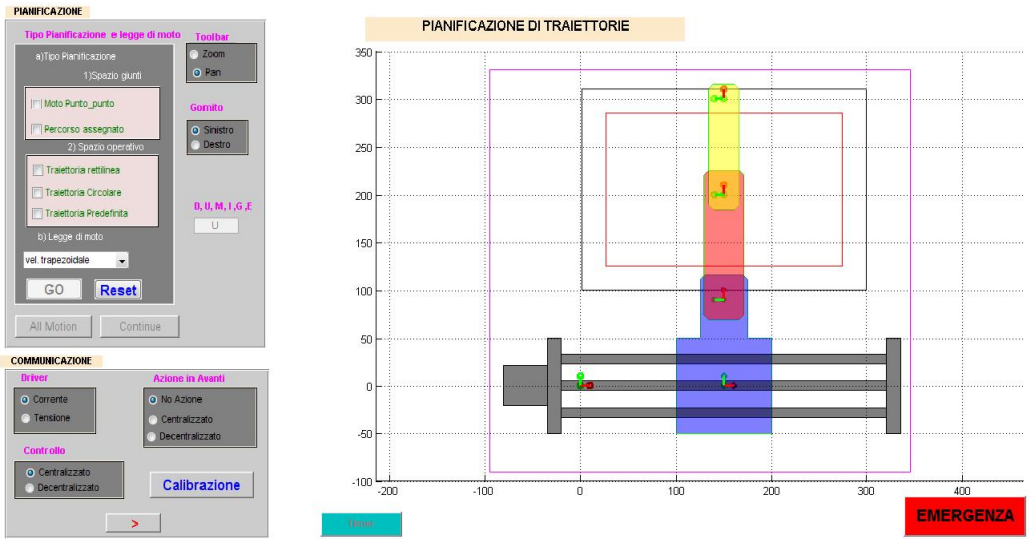

Figura 4.1: interfaccia GUI

L'interfaccia realizzata con la GUI di Matlab in figura 4.1) Permette Non solo all'utente di scegliere i criteri per la pianificazioni ma anche di visualizzare alla frequenza di 25 Hz tutte le posizioni del manipolatore inviati da Simulink ; e

contemporaneamente di inviare alla frequenza di 100 Hz le posizioni pianificate . Le tappe di funzionamento della Gui sono le seguenti :

- La scelta del tipo di driver ,controllo ,azione in avanti
- La scelta del tipo di pianificazione
- La scelta della legge di moto

4.2 Funzionamento e Modo Uso dell'interfaccia

Il funzionamento dell'interfaccia GUI in figura è comandato dal valore della variabile di stato 'stato' ricevuto da simulink:

1. se lo stato è allo zero cioè stato emergenza tutta la GUI è disattivata e non viene visualizzato il manipolatore
2. se lo $stato = 1$ si attiva la parte di 'comunicazione' e l'utente può scegliere il tipo di driver , azione in avanti o il controllo poi preme il tasto 'Calibrazione' a questo punto lo stato desiderato diventa 'stato desiderato = 3' per la calibrazione.
3. se lo $stato = 3$ si attiva il tasto $' >'$ che una volta premuto porta lo stato desiderato a 'stato desiderato = 4' per il plot del manipolatore in posizione iniziale
4. se lo $stato = 4$ viene disegnato il manipolatore in posizione a questo punto l'utente può scegliere il tipo di pianificazione e la legge di moto con cui vuole che il manipolatore si muova.

Calibrazione Per dare invio all'inizio della calibrazione del manipolatore
Emergenza porta il controllo nello stato di emergenza
Fine Emergenza per segnalare la fine di emergenza porta lo stato desiderato ad 1.
Stop timer per arrestare la comunicazione tra Matlab e Simulink chiude tutte le connessioni.

Go Attivata quando viene scelta il tipo di pianifficazione calcola tutte le posizioni desiderati del manipolatore per raggiungere un o più punti.
Reset Cancella tutte le traiettorie disegnate e punti selezionati .
All Motion visualizza tutti i movimenti precedente effettuato dal manipolatore.
Continue per continuare la pianificazione calcola le posizione del manipolatore

4.3 pnet e Timer

La pnet è una libreria di I/o per la comunicazione attraverso porte UDP o TCP tra Matlab e altri software Peter *Rydersäter* della Mid sweden University. Nel nostro lavoro abbiamo usato La comunicazione tramite protocollo UDP servendosi di alcuni metodi della pnet .

```
pnet('closeall') ;% chuide tutte le conessione aperte
send_ip  ='10.157.4.137';%'192.168.64.85';
Local_port=3335; % porta di ricezione
Remote_port=3336; % porta di spedizione
receive_socket = pnet('udpsocket',Local_port);%socket ricezione
send_socket = pnet('udpsocket',Remote_port);
pnet(send_socket,'udpconnect',send_ip,Remote_port);% connessione
```

che vengono definite all'apertura della GUI per evitare di dovere aprire e chuidere le porte di connessione che sono un grande rischio.
Inizialmente per La creazione di ogni porte si crea un socket per la trasmissione dati ed impostazione della porta sorgente. In quest'applicazione abbiamo usato due tipi di socket per la comunicazioni uno per invio e l'altro per la ricezione definiti all'apertura della GUI.
Abbiamo definito 2 Timer : uno per invio dei dati e l'altra per La ricezione e il plot del movimento .

Invio

Viene mandato a Simulink un vettore con 13 elementi tra cui le posizioni di tutti i link le accelerazioni e le velocita lo stato desiderato il tipo di driver di azione in

avanti e del controllo e infine della posizione del della penna. tutti quanti salvati nel l'ordine

$Vect\_out = [posizioni, driver1, driver2, Azione, stato\_des, penna, velocita, accelerazione,]$

- Le posizioni velocità e accelerazioni vengono calcolati in funzione del tipo di pianificazione che si desidera.
- $driver_{1,2} = 0$ se primo o secondo driver in corrente $driver_{1,2} = 1$ se primo driver in tensione.
- $Azione = 0$ per il controllo senza azione in avanti $Azione = 1$ per azione in avanti decentralizzato $Azione = 2$ per Azione in avanti centralizzato ;
- lo stato desiderato sono $stato\_des = 0$ se viene premuto il pulsante **Emergenza** $stato\_des = 1$ all'avvio della GUI $stato\_des = 2$ se premuto il pulsante **Calibrazione** $stato\_des = 4$ se viene premuto il pulsante ¿
- $penna = 1$ se la penna deve stare su altrimenti $penna = 0$

Il timer per inviare i dati al Simulink viene definita secondo il seguente codice

```
Period=0.01;
MyTimer=timer('ExecutionMode','fixedRate','Period',Period);
MyTimer.TimerFcn={@Send ,handles.figure1};
```

Questo timer chiama la funzione Send ad una frequenza di 100Hz che esegue l'invio dei dati e usa i metodi della pnet seguente per la scrittura dei Pacchetti inviati in Simulink.

```
pnet(send_socket,'write',Vect_out,'double','native');
 pnet(send_socket,'writepacket');
```

Ricezione

Sono sei le variabile che riceviamo da Simulink e vengono salvati nel vettore *data_received* sono letti se e solo se la connessione risulta un successo cioè il numero di elementi corrisponde a quello da leggere. questa verifica viene fatta usando i comandi seguenti :

```
len_byte = 8*6; % ogni double -> 8 byte
len = pnet(handles.receive_socket,'readpacket',len_byte);
 if len > 0 % altrimenti non c'è connessione
 esegue
 end
```

Abbiamo

$$data\_received = [link0, link1, link2, stato, link1 pianificato, link2 pianificato];$$

ove

- link0,link1,link2 sono le posizioni reali del manipolatore
- $stato = 0$ il controllo è nello stato di emergenza in questa situazione la Gui è disattivata . $stato = 0$ il controllore è appena uscito dallo stato di emergenza è pronto alla calibrazione a questo punto viene abilitato il tasto di calibrazione .$stato = 2$ sta calibrando ,$stato = 3$ per la fine della calibrazione. $stato = 4$ se il manipolatore ha finito la calibrazione e è in posizione iniziale per muoversi.
- link1pianificato,link2pianificato sono le posizione dei link1 e 2 trovati pianificando le traiettorie del manipolatore questi risultano utile per fare il plot contemporaneamente il movimento dell'end effector pianificata a quello reale.

Il timer per la ricezione (definita come nel caso per l'invio) dei dati da Simulink e plot del movimento del manipolatore chiama la funzione receive ogni 25Hz.

Conclusioni

Lo scopo di questo lavoro di tesi è stato quello di pianificare il moto del manipolatore a tre gradi di libertà e creare un interfaccia utente con la GUI di Matlab che permettesse all'utente di definire i parametri per la pianificazione del manipolatore e di visualizzare il moto reale effettuato con le varie posizioni ricevuti attraverso la comunicazione con il software per il controllo di posizione fatto in Simulink il quale gira in real time in un computer diverso da quello in cui viene pianificato il movimento del manipolatore .Per la pianificazione del movimento del nostro manipolatore da un posizione iniziale a una finale è stato definito tenendo in considerazione le varie posizioni del carrello per poterlo ottimizzare ,per questo oltre ad avere definito i criteri di pianificazione abbiamo anche creato una funzione di ottimizzazione del moto del manipolatore che calcola il valore finale di posizione del carrello in funzione della configurazione iniziale e del punto finale da raggiungere per ottimizzare sia il tempo di movimentazione sia il movimento dei due membri. Per la comunicazione abbiamo usato due tipi di funzione timer tra cui uno per invio dati e l'atro per la ricezione e plot della movimentazione questa tappa è stata agevolata con la funzione le pnet per la definizione e l'uso dei socket. La Gui di Matlab per la creazione dell'interfaccia utente si è rivelato funzionare in modo soddisfacente rispettando le previsione del lavoro iniziale.

Bibliografia

[1] A.Rizzo, "Pianificazione di traiettorie," *Appunti*, p. 1, 2003.

[2] G. Legnani, *Robotica industriale*. Casa Editrice Ambrosiana, 2003.

[3] P. P. Rocco, "Controllo del moto e robotica industriale," *Appunti*, p. 1, 2003.

Ringraziamenti

Un ringraziamento ai membri della mia famiglia che hanno sempre creduto in me dandomi la possibilità di poter realizzare i miei sogni Malgrado le diffilcoltà economiche.
Ringrazio mia sorella Donna e suo marito Martial e i loro meravigliosi bambini Evrad Marielle e il prossimo per il loro aiuto e i loro consigli .
Un caro ringraziamento alla famiglia Tomasello per il calore e l'affetto che mi hanno sempre portato.
Un tenero e particolare ringraziamento al mio fidanzato Daniele per la sua disponibilità ad aiutarmi sempre e per il suo sostegno morale e per i suoi preziosi e intelligenti consigli su questo progetto.
Al professore Rosati per la sua disponibilità a risolvere ogni diffilcotà incontrata durante il lavoro.