Corso di Laurea Magistrale in

Ingegneria Biomedica

Esame di

Robotica Medica

Lezione del 24/10/23

Inversione della cinematica differenziale II

Esercitazione 4

Anno Accademico 2022/2023

Traccia

Facendo riferimento all'esercitazione precedente, si retroazioni l'orientamento facendo ricorso ai quaternioni
Si suggerisce di scrivere dapprima le funzioni che consentono la trasformazione fra le diverse rappresentazioni dell'orientamento (Angoli di Eulero, quaternione, matrice di rotazione)
Una possibile implementazione è disponibile nel file `soluzione04.zip`

Figura 1: Traccia

La quarta esercitazione affronta anch'essa un problema di implementazione di un algoritmo per l'inversione cinematica, il quale però si basa su *quaternioni*. Oltre allo script "main" viene quindi richiesto di implementare nello specifico due funzioni MATLAB: una che consenta la trasformazione tra diverse rappresentazioni dell'orientamento (in particolare la funzione "*rot2quat*" per passare da matrice di rotazione a quaternioni); l'altra ("*quatError*") relativa all'errore quaternione. Il caso in esame rimane sempre quello di un manipolatore planare a 3 bracci.

Background teorico

L'*inversione cinematica* è il processo di determinare le posizioni delle giunture di un robot in modo che l'estremità del braccio robotico raggiunga una posizione e orientamento desiderati nello spazio.
Nel contesto della cinematica inversa, è essenziale definire e misurare l'errore di posizione e l'errore di orientamento tra la posizione desiderata e quella calcolata effettivamente dall'organo terminale del robot. L'*errore di posizione* è la differenza tra la posizione desiderata e quella calcolata, mentre l'*errore di orientamento* è la discrepanza tra l'orientamento desiderato e quello calcolato, che varia a seconda della rappresentazione dell'orientamento scelta, come nel caso specifico dei quaternioni unitari. Dunque, la rappresentazione dell'errore di orientamento dipende dalla particolare rappresentazione dell'orientamento dell'organo terminale del robot che viene scelta. Queste rappresentazioni possono essere gli angoli di Eulero, la descrizione asse e angolo o i quaternioni unitari.
L'uso di quaternioni unitari è una scelta comune in quanto garantisce che il quaternione abbia una norma (lunghezza) di uno, rendendo la rappresentazione stabile matematicamente.
Per procedere con l'inversione cinematica basata sui quaternioni unitari, è necessario definire un errore di orientamento. Questo errore di orientamento è calcolato come la differenza tra i quaternioni unitari associati all'orientamento desiderato, $Q_d = \eta_d, \boldsymbol{\epsilon}_d$ e all'orientamento calcolato, $Q_e = \eta_e, \boldsymbol{\epsilon}_e$ dell'organo terminale del robot. L'errore di orientamento può essere definito in termini di una matrice di rotazione $\mathbf{R}_d\,\mathbf{R}_e^T$ (dove $\mathbf{R}_d$ è la matrice di rotazione associata a Q_d).
L'errore di orientamento, rappresentato come $\Delta Q = \Delta\eta, \Delta\boldsymbol{\epsilon}$, è calcolato come:

$$\Delta Q \;=\; Q_d * Q_e^{-1}$$

Per semplificare il calcolo dell'errore di orientamento, esso viene definito come:

$$\mathbf{e}_o \;=\; \Delta\boldsymbol{\epsilon} \;=\; \eta_e(\mathbf{q})\boldsymbol{\epsilon}_d \;-\; \eta_d\boldsymbol{\epsilon}_e(\mathbf{q}) \;-\; \mathbf{S}(\boldsymbol{\epsilon}_d)\boldsymbol{\epsilon}_e(\mathbf{q})$$

L'uso dell'operatore anti-simmetrico $\mathbf{S}$ contribuisce a calcolare l'errore di orientamento in modo efficiente. Tuttavia, vale la pena notare che, in generale, non è possibile calcolare direttamente η_e e $\boldsymbol{\epsilon}_e$ dalle variabili di giunto;

Infine, l'errore di orientamento ($\mathbf{e}_o$) è utilizzato nell'algoritmo di inversione cinematica. La formula:

$$\dot{\mathbf{q}} \;=\; \mathbf{J}^{-1}(\mathbf{q}) \begin{bmatrix} \dot{\mathbf{p}}_d \;+\; \mathbf{K}_P\mathbf{e}_P \\ \dot{\boldsymbol{\omega}}_d \;+\; \mathbf{K}_O\mathbf{e}_O \end{bmatrix}$$

rappresenta come vengono calcolate le velocità dei giunti ($\dot{\mathbf{q}}$) utilizzando l'inversa dello Jacobiano ($\mathbf{J}^{-1}$), l'errore di posizione ($\mathbf{e}_P$), la velocità desiderata dell'organo terminale ($\dot{\mathbf{p}}_d$), e le costanti di guadagno ($\mathbf{K}_P$ ed $\mathbf{K}_O$) e l'errore di orientamento ($\mathbf{e}_O$). Questo calcolo sfrutta lo *Jacobiano geometrico*.

L'equazione $\boldsymbol{\omega}_d - \boldsymbol{\omega}_e + \mathbf{K}_O\mathbf{e}_O = 0$ mostra come il termine di errore di orientamento $\mathbf{e}_O$influisce sull'errore di velocità angolare ($\boldsymbol{\omega}_d \;-\; \boldsymbol{\omega}_e$), contribuendo alla regolazione dell'orientamento del robot.

Nella cinematica inversa di un robot, la stabilità del sistema è cruciale per garantire che l'organo terminale raggiunga con precisione e in modo stabile la posizione e l'orientamento desiderati.

L'equazione dell'errore di orientamento è non lineare in $\mathbf{e}_O$ poiché contiene l'errore di velocità angolare ($\boldsymbol{\omega}_d \;-\; \boldsymbol{\omega}_e$) al posto della derivata temporale dell'errore di orientamento ($\dot{\mathbf{e}}_O$). Per comprendere la relazione tra la derivata temporale del quaternione Q_e e la velocità angolare $\boldsymbol{\omega}_e$ vengono introdotte le equazioni note come "propagazione del quaternione":

$$\dot{\eta_e} \;=\; -\frac{1}{2}\,\boldsymbol{\epsilon}_e^T\,\boldsymbol{\omega}_e$$

$$\dot{\boldsymbol{\epsilon}_e} \;=\; \frac{1}{2}(\eta_e\mathbf{I}_3 \;-\; \mathbf{S}(\boldsymbol{\epsilon}_e))\,\boldsymbol{\omega}_e$$

Queste relazioni mostrano come variano nel tempo la parte reale e immaginaria del quaternione Q_e rispetto alla velocità angolare $\boldsymbol{\omega}_e$.

Infine, viene introdotta la funzione candidata di Lyapunov, V, che dipende dalla discrepanza tra i quaternioni associati all'orientamento desiderato (Q_d) e all'orientamento calcolato (Q_e). Questa funzione è fondamentale per lo studio della stabilità del sistema. Derivando questa funzione rispetto al tempo e considerando l'equazione $\boldsymbol{\omega}_d - \boldsymbol{\omega}_e + \mathbf{K}_O\mathbf{e}_O = 0$ che descrive l'errore nell'angolo di velocità tra l'orientamento desiderato e calcolato, si ottiene l'equazione:

$$\dot{V} \;=\; -\,\mathbf{e}_O^T\,\mathbf{K}_O\,\mathbf{e}_O$$

Questa equazione è definita come negativa, il che implica che $\mathbf{e}_O$ (l'errore di orientamento) converge a zero nel tempo. Questa è una proprietà desiderabile in termini di controllo e stabilità: significa che l'errore di orientamento si riduce sempre di più, garantendo che il robot si avvicini sempre più all'orientamento desiderato.

Progettazione dello script

L'esercitazione prevede essenzialmente la scrittura di due funzioni MATLAB, cioè *rot2quat* e *quatEror*. Di seguito viene illustrata la schematizzazione concettuale delle conversioni teoriche che sono in grado di effettuare.

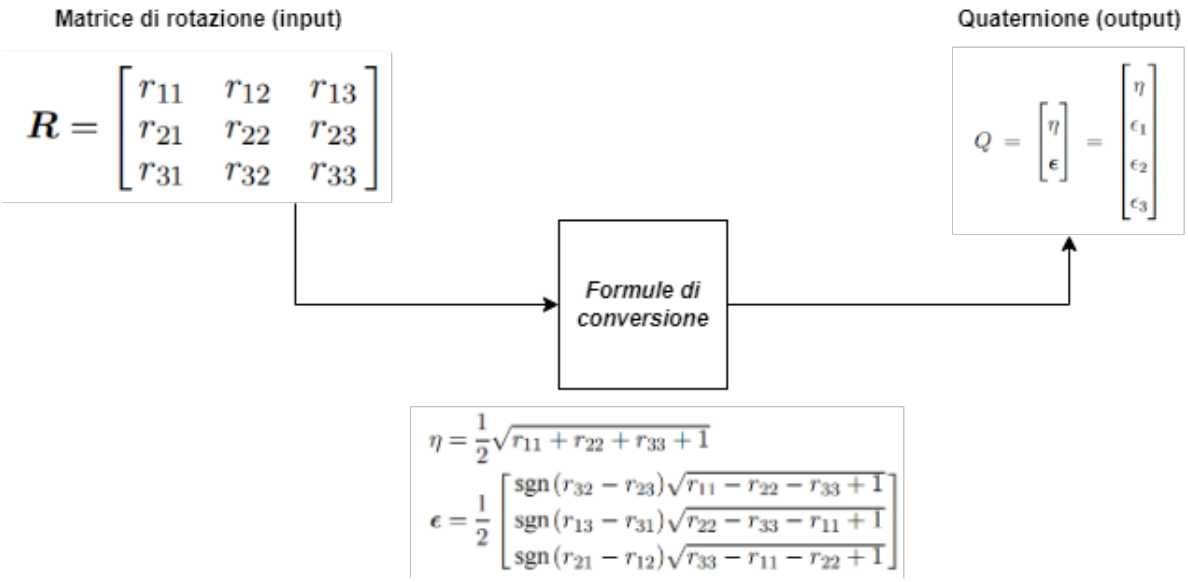

Figura 2: Conversione dalla matrice di rotazione al quaternione

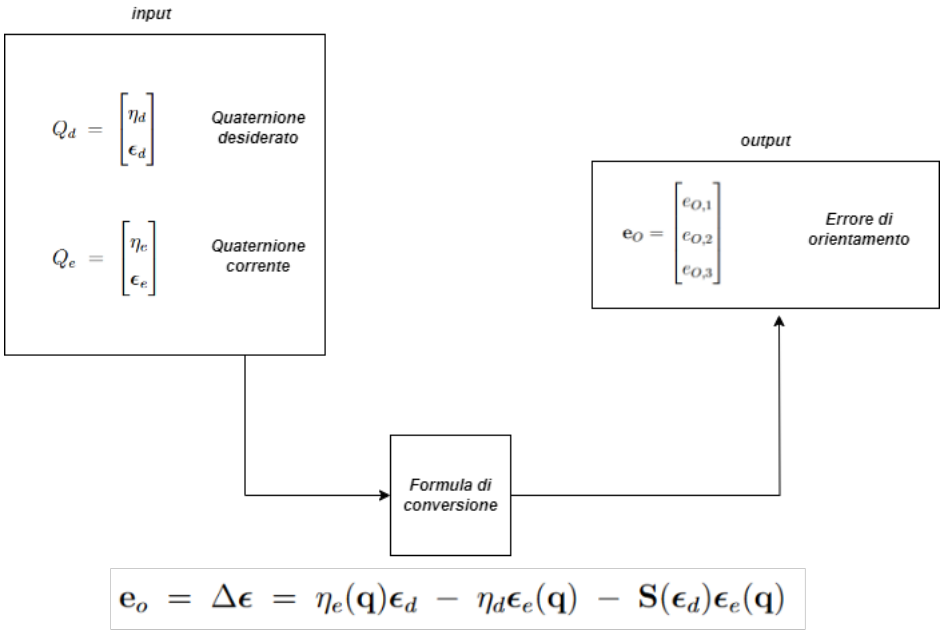

Figura 3: Calcolo dell'errore di orientamento a partire dalla conoscenza del quaternione desiderato e del quaternione corrente

Implementazione

Lo script "*main*" della quarta esercitazione (che può essere visualizzato al seguente link a piè di pagina [1]) ricalca in gran parte quello dell'esercitazione precedente, la terza.
I due script trattano del problema di inversione della cinematica differenziale per un manipolatore planare a 3 bracci, ma presentano alcune differenze significative nella rappresentazione dell'orientamento e nell'implementazione. In particolare, quello della terza esercitazione utilizza la rappresentazione dell'orientamento come somma dei tre angoli ai giunti, mentre questo in esame utilizza i quaternioni unitari per calcolare l'errore di orientamento. In questa esercitazione si utilizza una rappresentazione più compatta e robusta dell'orientamento attraverso quaternioni unitari. Inoltre, si sfruttano le funzioni *rot2quat* e *quatError* per calcolare l'errore di orientamento, migliorando la precisione e l'efficienza del controllo dell'orientamento del manipolatore. Questo rende il secondo script più flessibile e adatto a scenari in cui è importante controllare sia la posizione che l'orientamento del manipolatore.

```
function quat = rot2quat(rotMatrix)

    r11 = rotMatrix(1,1);
    r22 = rotMatrix(2,2);
    r33 = rotMatrix(3,3);

    r13 = rotMatrix(1,3);
    r31 = rotMatrix(3,1);

    r23 = rotMatrix(2,3);
    r32 = rotMatrix(3,2);

    r12 = rotMatrix(1,2);
    r21 = rotMatrix(2,1);

    eta = 0.5*sqrt(r11 + r22 + r33 + 1);

    eps1 = 0.5*sign(r32 - r23)*sqrt(r11 - r22 - r33 + 1);
    eps2 = 0.5*sign(r13 - r31)*sqrt(r22 - r33 - r11 + 1);
    eps3 = 0.5*sign(r21 - r12)*sqrt(r33 - r11 - r22 + 1);

    quat = [eta eps1 eps2 eps3]';

end
```

```
function eO = quatError(quatE, quatD)

    etaE = quatE(1);
    etaD = quatD(1);

    epsE = quatE(2:end);
    epsD = quatD(2:end);

    eO = etaE*epsD - etaD*epsE - cross(epsD, epsE);

    eta_tilde = quatE(1)*quatD(1) + quatE(2:end)'*quatD(2:end);

    if eta_tilde < 0

        eO = -eO;

    end

end
```

Figura 4: (a) Codice della funzione MATLAB "*rot2quat*" ; (b) Codice della funzione MATLAB "*quatError*"

Prima differenza: Rappresentazione dell'orientamento desiderato

- Nel *main3*, l'orientamento desiderato x_d è rappresentato come un vettore di tre elementi che corrispondono agli angoli ai giunti, specificamente $x_d = [1.8\ 0.6\ \pi/8]$. Questi angoli definiscono l'orientamento desiderato del manipolatore;

- Nel *main4*, l'orientamento desiderato è rappresentato da un vettore x_d di tre elementi (lo stesso di prima), ma si introduce anche un quaternione unitario QD, rappresentato come $QD = [1,0,0.01,0.01]$. Questo quaternione rappresenta in modo più compatto e robusto l'orientamento desiderato del manipolatore.

[1] https://github.com/SimCir01/robotica-medica/blob/main/Esercitazioni/Esercitazione_4/Script/main.m

```
% Definisco il set point desiderato
xd = [1.8 0.6 pi/8]';
QD = [1 0 0.01 0.01]';

% Definisco la matrice dei guadagni
matrixK_P = diag([25 25 25]);
matrixK_O = diag([25 25 25]);
```

Figura 5: Rappresentazione dell'orientamento desiderato

Seconda differenza: Calcolo dell'errore di orientamento

- Nel *main3*, l'errore di orientamento non è calcolato direttamente tramite quaternioni. L'errore di orientamento è rappresentato come un vettore di tre elementi, che rappresentano gli errori sugli angoli ai giunti. L'errore è calcolato come la differenza tra l'orientamento calcolato xe (ottenuto dalla matrice di trasformazione) e l'orientamento desiderato xd;
- Nel *main4*, l'errore di orientamento è calcolato direttamente usando la funzione *quatError*. Questa funzione calcola l'errore tra due quaternioni unitari, QE (ottenuto dalla matrice di rotazione) e QD. L'errore di orientamento è rappresentato come un vettore di quattro elementi, che contengono le differenze tra gli elementi dei quaternioni. Questo approccio è più robusto e evita problemi legati alle singolarità.

Terza differenza: Calcolo dell'errore totale

Nel *main4*, l'errore totale è calcolato considerando sia l'errore di posizione eP che l'errore di orientamento eO. Questo consente di regolare separatamente i guadagni per l'errore di posizione e di orientamento.

```
% Determino posizione e orientamento corrente dell'e.e
% Aggiungo il valore corrente alla matrice delle pose
xe_i = T0(1:3,4);
rotM = T0(1:3,1:3);
QE = rot2quat(rotM);
xe(:,i) = xe_i;

% Calcolo l'errore e lo salvo nella matrice degli errori
eP = xd - xe_i;
eO = quatError(QE, QD);
errors(:,i) = [eP; eO];

% Calcolo il Jacobiano geometrico ed estraggo le righe funzionali
J = Jacobian(DH);

eTot = [matrixK_P*eP; matrixK_O*eO];
```

Figura 6: Calcolo dell'errore di orientamento

Quarta differenza: Plot degli errori

- Nel *main3*, ci sono due subplot relativi agli errori. Nel primo subplot vengono mostrati gli errori di posizione sull'asse X e Y. Nel secondo subplot viene mostrato l'errore di orientamento in termini di angolo θ. L'errore di orientamento in questo caso è rappresentato come unico angolo θ.;

- Nel *main4*, ci sono due subplot relativi agli errori di posizione e orientamento. Nel primo subplot vengono mostrati gli errori di posizione sull'asse X, Y e Z. Nel secondo subplot vengono mostrati gli errori di orientamento in termini di angoli ϕ,θ, e ψ. L'errore di orientamento è rappresentato come una terna di angoli, che fornisce una rappresentazione più dettagliata dell'orientamento desiderato rispetto a un singolo angolo θ.

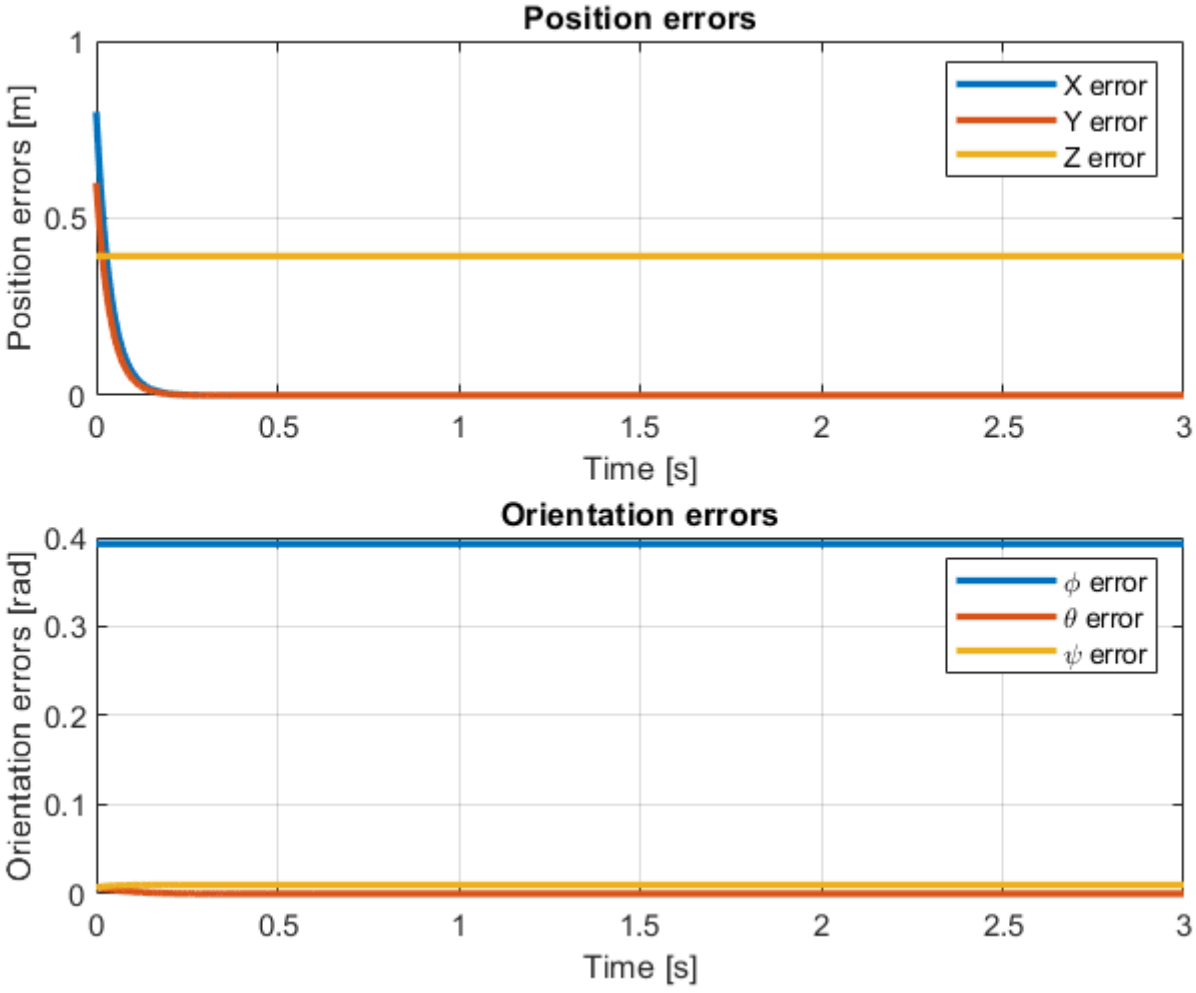

Figura 7: Plot degli errori di posizione ed orientamento

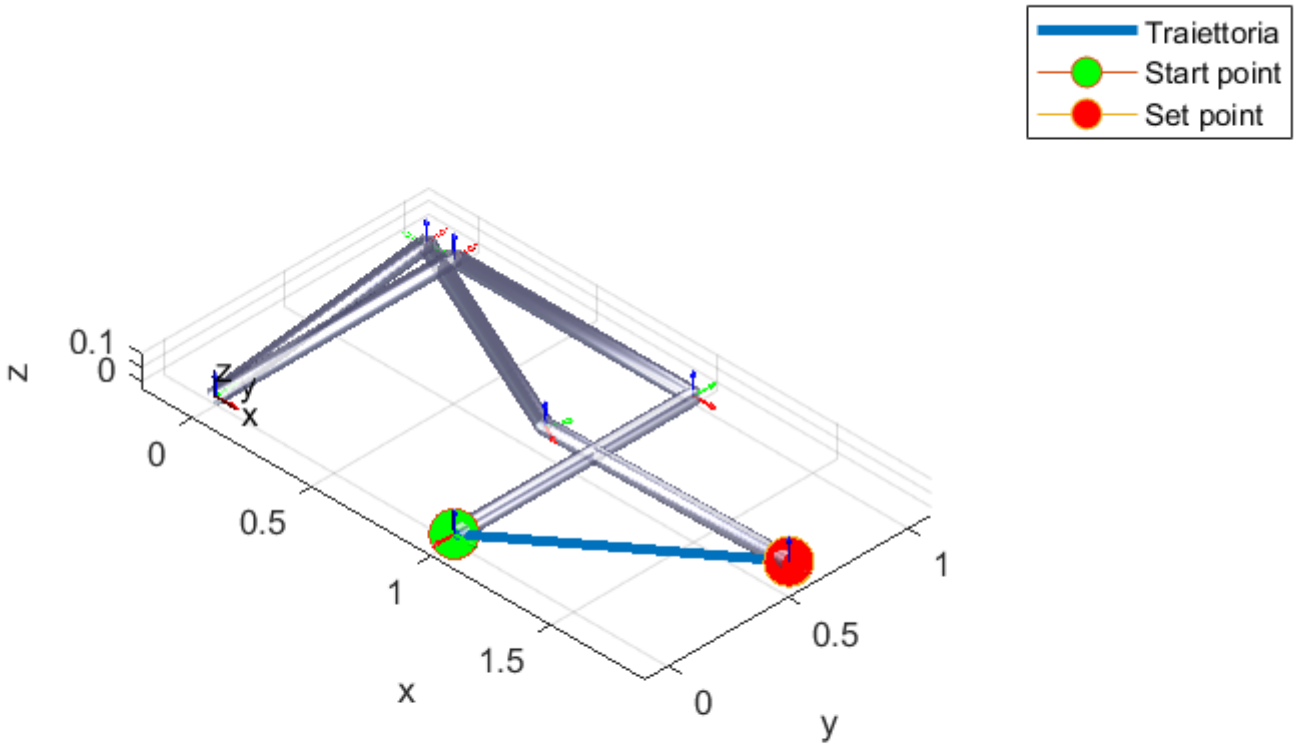

Figura 8: Visualizzazione tramite la funzione MATLAB *DrawRobot* delle configurazioni iniziale e finale del manipolatore e della traiettoria per arrivare dallo *start point* al *set point*