Corso di Laurea Magistrale in

Ingegneria Biomedica

Esame di

Robotica Medica

Lezione del 19/10/23

Inversione della cinematica differenziale I

Esercitazione 3

Anno Accademico 2022/2023

Traccia

Facendo riferimento ad un manipolatore planare a 3 bracci si implementino i due algoritmi di inversione della cinematica differenziale tramite

- inversa dello jacobiano
- trasposta dello jacobiano

Si assegni un set-point di posizione/orientamento desiderati ($\dot{\boldsymbol{x}} = \mathbf{0}$ e $r = 3$ quindi) ed un passo di campionamento di $T = 1$ ms
Si retroazioni l'orientamento usando $\psi_d - \psi$ (ricavare ψ semplicemente come somma dei 3 giunti)
Una possibile implementazione è disponibile nel file `soluzione03.zip`

Figura 1: Traccia

La terza esercitazione è la prima a dare avvio ad una serie di esercitazioni relative al controllo dell'inversione cinematica differenziale. Per eseguirle si parte dal caso più semplice (manipolatore planare a tre bracci) fino ad arrivare a trattare casi più complessi (Jaco a 7 DOF).
Questa esercitazione, prendendo come riferimento appunto il manipolatore planare a tre bracci, richiede di implementare due algoritmi di inversione della cinematica differenziale (uno attraverso l'inverso dello Jacobiano mentre l'altro attraverso la trasposta). Il tempo di campionamento assegnato corrisponde a $T = 1\ ms$. Si deve partire assegnando un set point di posizione ed orientamento desiderati (dove per set point, non si intende ancora una traiettoria ma un punto costante e fisso nello spazio): con $\dot{x} = 0$ si intende che la velocità desiderata dell'end effector è nulla mentre con $r = 3$ si intende dire che la dimensione dello spazio in cui bisogna espletare tale compito coincide con quella dello spazio operativo, essendo n = 3. Bisogna retroazionare l'orientamento usando la formula $\psi_d - \psi$ per il calcolo dell'orrore di orientamento. Questo significa che per un robot planare l'orientamento si può esprimere tramite un solo angolo, e quindi l'errore di orientamento si esprime sottraendo all'angolo desiderato ψ_d quello corrente ψ. Inoltre, l'orientamento corrente ψ dell'end effector lo si ottiene tramite la somma dei tre angoli di giunto $(\theta_1 + \theta_2 + \theta_3)$. Ovviamente, quest'ultima cosa è valida solo per il manipolatore in esame, che è planare, mentre per quelli tridimensionali il discorso è ben diverso.

- Assumendo t_f (`tf`) come il tempo finale della simulazione e T come il passo di campionamento, definire un vettore dei tempi t e il numero totale dei campioni della simulazione N come

```
t = 1:T:tf;
N = length(t);
```

- Ogni variabile $(\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t), \boldsymbol{x}(t), \boldsymbol{x}_d(t), \ldots)$ può essere inizializzata come

```
q = zeros(n,N)
```

  e `q(:,i)` rappresenterà $\boldsymbol{q}(t_i) = \boldsymbol{q}(i \cdot T)$
- Implementare una semplice integrazione del primo ordine del tipo

```
q(:,i+1) = q(:,i) + T*dq(:,i)
```

- Dopo la simulazione, plottare tutte le variabili: `plot(t,q)` ...

```
% initialize variables
for i=1:N
% generate desired e.e. value
...
% compute current e.e. value
...
% compute controller's output
...
% integration
...
end
% plot
```

Figura 2: (a) Suggerimenti ; (b) Pseudo-codice

Background teorico

Il *problema cinematico diretto* permette, conoscendo le variabili di giunto $\mathbf{q}$, di arrivare a definire la posa (posizione ed orientamento) dell'end effector $\mathbf{x_e}$. Il *problema cinematico inverso*, al contrario, consiste nel partire dalla posizione e dall'orientamento dell'end effector $\mathbf{x_e}$ per arrivare a definire le variabili di giunto $\mathbf{q}$ da cui essi vengono ottenuti.

$$\begin{array}{ccccc} \mathbf{q} & \rightarrow & \mathbf{x_e} & & \textit{Cinematica diretta} \\ \mathbf{x_e} & \rightarrow & \mathbf{q} & & \textit{Cinematica inversa} \end{array}$$

Mentre l'equazione cinematica diretta è caratterizzata da equazioni non lineari $\mathbf{x_e} = \mathbf{k}(\mathbf{q})$, l'equazione cinematica differenziale $\mathbf{v_e} = \mathbf{J}(\mathbf{q})\,\dot{\mathbf{q}}$ rappresenta una trasformazione lineare tra spazio dei giunti e spazio operativo.

$$\dot{\mathbf{q}} \quad \rightarrow \quad \mathbf{v_e} \qquad \textit{Cinematica differenziale diretta}$$

Il legame lineare che caratterizza la cinematica differenziale apre alla possibilità di sfruttare l'equazione cinematica differenziale per affrontare i problemi dell'inversione cinematica. Nel caso in cui $n = r$ lo Jacobiano risulta essere una matrice quadrata e non singolare, dunque invertibile, il che permette di ottenere le velocità ai giunti mediante semplice inversione dello Jacobiano: assegnando all'e.e. una traiettoria di moto specificando $\mathbf{v_e}(t)$ e le condizioni iniziali su posizione e orientamento, l'obiettivo è determinare una possibile traiettoria ai giunti ($\mathbf{q}$, $\dot{\mathbf{q}}$) che riproduca la traiettoria data, attraverso l'equazione $\dot{\mathbf{q}} = \mathbf{J^{-1}}(\mathbf{q})\,\mathbf{v_e}$

$$\begin{array}{ccccc} \mathbf{v_e} & & & & \\ + & & \rightarrow & \dot{\mathbf{q}} & \text{Inversione della cinematica differenziale} \\ \text{cond iniziali} & & & & \end{array}$$

Secondo la *regola di integrazione di Eulero*, nota la postura iniziale del manipolatore $\mathbf{q}(0)$, le posizioni $\mathbf{q}$ possono essere ottenute integrando le velocità $\dot{\mathbf{q}}$ nel dominio del tempo, ottenute tramite inversione della cinematica differenziale. In forma numerica, fissato un intervallo di integrazione Δt, note posizione e velocità dei giunti all'istante di tempo t_k, si calcola la posizione dei giunti al tempo $t_{k+1} = t_k + \Delta t$:

$$\mathbf{q}(t_{k+1}) = \mathbf{q}(t_k) \; + \; \dot{\mathbf{q}}(t_k)\,\Delta t$$
$$\mathbf{q}(t_{k+1}) = \mathbf{q}(t_k) \; + \; \mathbf{J^{-1}}(\mathbf{q}(t_k))\,\mathbf{v_e} \;\; \Delta t$$

L'integrazione numerica comporta fenomeni di deriva dalla soluzione: le $\dot{\mathbf{q}}$ calcolate numericamente non coincidono con quelle che soddisfano $\dot{\mathbf{q}} = \mathbf{J^{-1}}(\mathbf{q})\,\mathbf{v_e}$ nel tempo continuo. Bisogna allora tener conto dell'errore commesso nello spazio operativo tra la posa desiderata e quella corrente:

$$\mathbf{e} = \mathbf{x_d} - \mathbf{x_e}$$

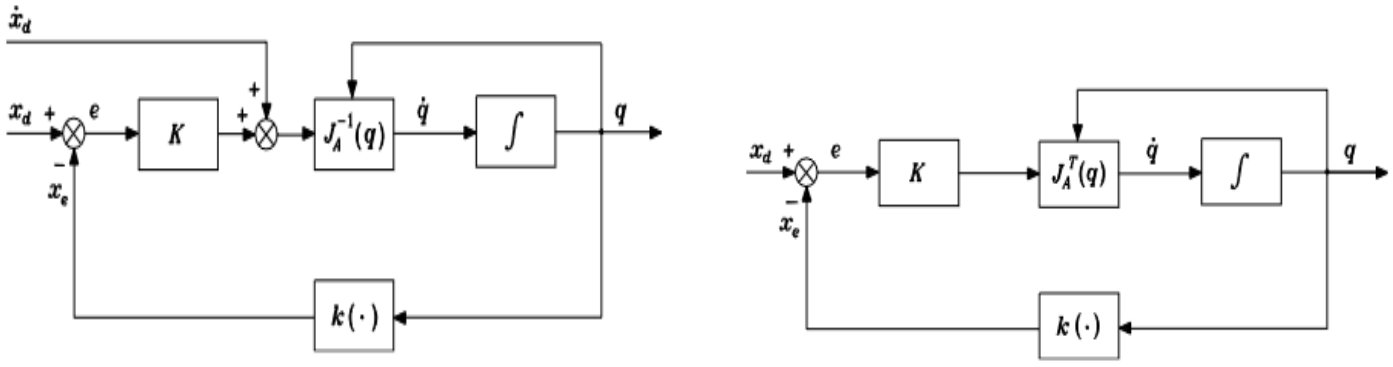
Inoltre, è fondamentale garantire un'iterazione accurata nel processo di ricerca della soluzione. La necessità di un *ciclo chiuso* all'interno del processo di ricerca della soluzione deriva dall'inesattezza dei calcoli o delle approssimazioni coinvolte nella cinematica inversa. A causa di errori numerici o imprecisioni nei dati di input, il risultato ottenuto in ogni iterazione potrebbe divergere dalla soluzione desiderata. Il ciclo chiuso è un meccanismo che consente di correggere iterativamente tali discrepanze, avvicinando gradualmente la soluzione al risultato atteso. Questo assicura che la soluzione sia stabile e converga al risultato corretto nel contesto della cinematica inversa.

Nell'ambito della cinematica inversa, due principali approcci vengono impiegati per ottenere la soluzione desiderata. Il primo metodo si basa sull'utilizzo dell'*inversa del Jacobiano analitico*, una matrice che collega le variazioni delle variabili cinematiche alle variazioni delle variabili congiunte. Questo approccio prevede il calcolo iterativo delle variazioni delle variabili cinematiche, utilizzando l'inversa del Jacobiano per tradurre queste variazioni nelle variabili congiunte in variazioni nello spazio operativo. Questo metodo è efficace quando il Jacobiano è ben definito e può essere calcolato in modo efficiente.

$$\dot{\mathbf{q}} = \mathbf{J_A^{-1}}(\mathbf{q})\,(\dot{\mathbf{x}}_\mathbf{d} \;+\; \mathbf{Ke})$$

Il secondo approccio si basa sull'utilizzo della trasposta del Jacobiano, che esprime la relazione inversa tra le variazioni delle variabili cinematiche e quelle delle variabili congiunte. Questo metodo consente il calcolo diretto delle variazioni delle variabili congiunte, che vengono poi utilizzate per aggiornare le variabili cinematiche. Questo approccio è particolarmente utile quando il calcolo dell'inversa del Jacobiano è complesso o problematico a causa della non-linearità delle equazioni cinematiche. La scelta tra questi due metodi dipende dalla natura del problema specifico e dalla necessità di ottenere una soluzione precisa e stabile. Entrambi mirano a gestire le sfide legate all'errore numerico e all'iterazione, garantendo che la soluzione finale sia coerente con gli obiettivi prefissati.

$$\dot{\mathbf{q}} = \mathbf{J_A^T}(\mathbf{q})\,\mathbf{Ke}$$



Algoritmo mediante inversa dello Jacobiano *Algoritmo mediante trasposta dello Jacobiano*

Figura 3: Algoritmi di inversione cinematica

Progettazione dello script

Lo script *main* deve essere progettato andando innanzitutto a definire le variabili in gioco. In un secondo momento va implementato il ciclo di controllo che permette di simulare ad ogni passo di campionamento l'aggiornamento delle variabili di giunto. Infine è consigliato plottare l'andamento dei vari tipi di errore e diverse configurazioni assunte dal robot durante l'analisi.

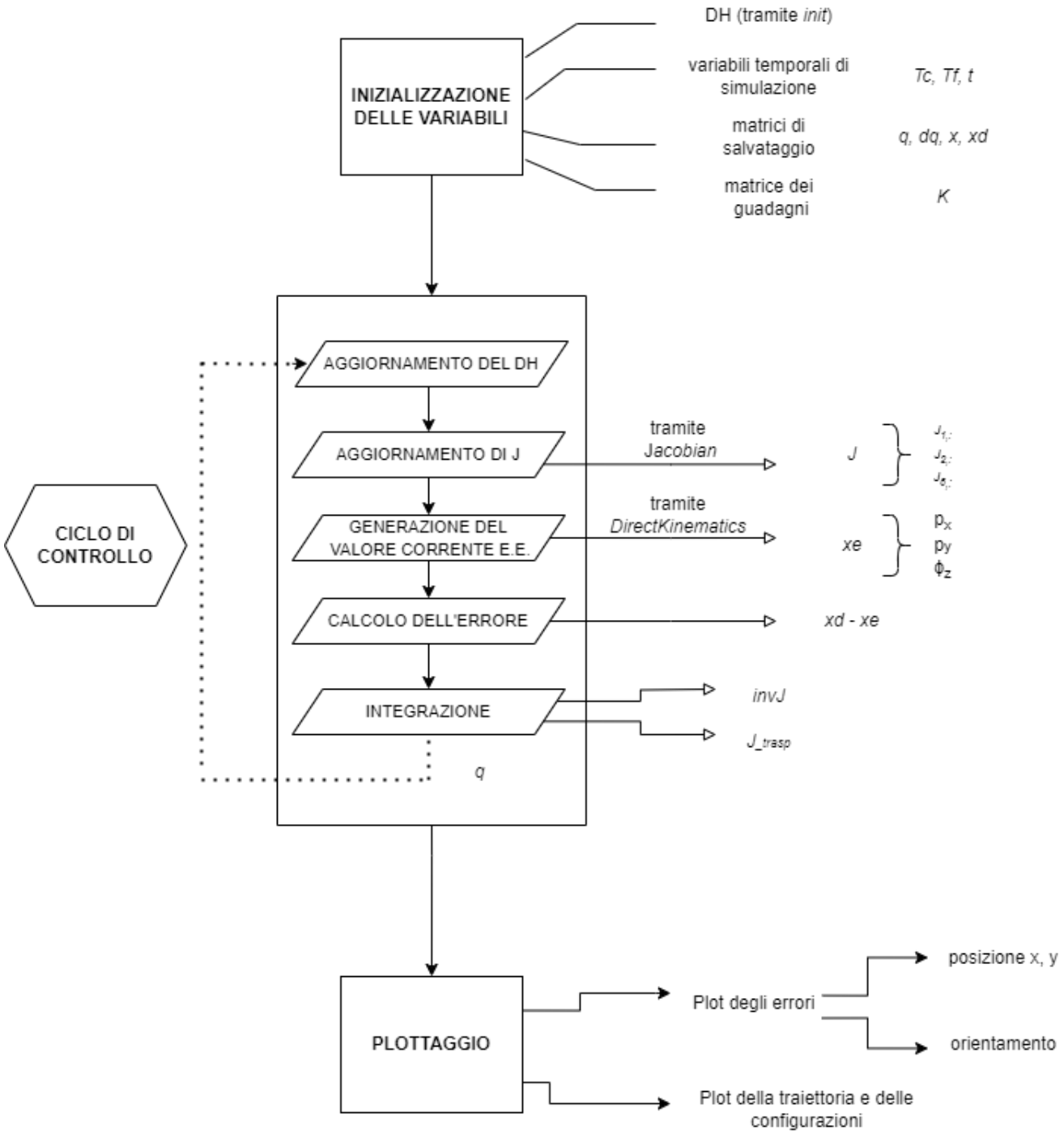


Figura 4: Workflow dello script di inversione della cinematica differenziale per un manipolatore a 3 bracci

Implementazione

Lo script implementato è disponibile al link GitHub a piè pagina [1].
Per poter effettuare il "Run" del codice è opportuno disporre delle funzioni MatLab contenute nella cartella "DrawRobot" e delle funzioni precedentemente implementate come "Homogeneous", "DirectKinematics" e "Jacobian", nonché ovviamente della funzione "init".

```
% Richiamo la configurazione iniziale del manipolatore
init

% Definisco le variabili temporali di simulazione
Tf = 3;
Tc = 1e-3;
t = 0 : Tc : Tf;
nPoints = length(t);

% Definisco il numero di bracci
nLink = length(DH(:,1));

% Definisco le matrici di salvataggio
q = zeros(nLink, nPoints);
dq = zeros(nLink, nPoints);
xe = zeros(nLink, nPoints);
errors = zeros(nLink, nPoints);
DHS = zeros(nLink, 4, nPoints);

% Definisco il set point desiderato
xd = [1.8 0.6 pi/8]';

% Definisco la matrice dei guadagni
matrixK = diag([25 25 25]);

% Salvo la prima tabella DH
firstDH = DH;
```

Figura 5: Parte iniziale del codice relativa alla inizializzazione delle variabili

La riga "*init*" richiama la configurazione iniziale del manipolatore ed include le specifiche DH (Denavit-Hartenberg) del manipolatore. Successivamente è opportuno definire le variabili temporali della simulazione: si definisce un intervallo di tempo totale Tf e un passo di campionamento Tc. Successivamente, si crea un vettore temporale t che copre il periodo di simulazione da 0 a Tf con incrementi di Tc. Vengono inizializzate alcune matrici (q, dq, xe, $errors$, DHS) per memorizzare i dati durante la simulazione. Viene definito un vettore xd che rappresenta il set point desiderato. Questo vettore contiene le coordinate x, y e l'angolo phi. Si definisce una matrice diagonale $matrixK$ con tre valori diagonali (25, 25, 25). Questa matrice verrà utilizzata nel calcolo della cinematica inversa. La configurazione iniziale DH del manipolatore viene salvata nella variabile $firstDH$.

[1] https://github.com/reFraw/robotica-medica/blob/main/Esercitazioni/Esercitazione_3/Script/main.m

```
for i = 1 : nPoints

    % Definisco il vettore corrente delle variabili di giunto
    q(:,i) = DH(:,4);

    % Calcolo la cinematica diretta
    T = DirectKinematics(DH);
    T0 = T(:,:,nLink);

    % Determino posizione e orientamento corrente dell'e.e come [x, y,
    % phi]' |
    % Aggiungo il valore corrente alla matrice delle pose
    xe_i = [T(1:2,4,nLink); sum(DH(:,4))];
    xe(:,i) = xe_i;

    % Calcolo l'errore e lo salvo nella matrice degli errori
    ei = xd - xe_i;
    errors(:,i) = ei;

    % Calcolo il Jacobiano geometrico ed estraggo le righe funzionali
    J = Jacobian(DH);
    J = J([1:2 6], :);
```

```
    if inverseKinematicsType == 'inverse'

        % Calcolo l'inversa dello Jacobiano
        invJ = inv(J);

        % Calcolo q differenziale e lo salvo nella matrice
        dqi = invJ*matrixK*ei;
        dq(:,i) = dqi;

    else

        % Calcolo la trasposta dello Jacobiano
        J_trasp = transpose(J);

        % Calcolo q differenziale e lo salvo nella matrice
        dqi = J_trasp*matrixK*ei;
        dq(:,i) = dqi;
    end

    % Calcolo le nuove variabili di giunto
    qi = q(:,i) + Tc*dqi;

    % Salvo e aggiorno la tabella di Denavit-Hartenberg
    DHS(:,:,i) = DH;
    DH(:,4) = qi;

end
```

Figura 6: Parte di codice relativa al ciclo di controllo

Si definisce una variabile *inverseKinematicsType* che può essere "inverse" o "transpose". Questo influenzerà il metodo utilizzato per la cinematica inversa. Viene inizializzato un *ciclo for* che scorre attraverso i punti temporali definiti in *t*. All'interno del ciclo, i seguenti calcoli vengono eseguiti per ogni iterazione: le variabili di giunto correnti (q) sono ottenute dalla matrice DH; Viene calcolata la cinematica diretta per ottenere la matrice di trasformazione T del manipolatore e la sua posizione corrente *T0*. La posizione e l'orientamento corrente dell'end-effector sono estratti da *T0* e salvati in xe_i; L'errore tra la posizione corrente e quella desiderata (*ei*) è calcolato e memorizzato in *errors*; Viene calcolato il Jacobiano geometrico J e vengono selezionate le righe rilevanti; In base al tipo di cinematica inversa selezionato, viene calcolato *dqi*, il cambio nelle variabili di giunto; Le nuove variabili di giunto (*qi*) vengono calcolate e memorizzate; La tabella DH viene aggiornata con i nuovi valori di giunto.

```
finalJointConfiguration = rad2deg(q(:,end));
```

```
%% Plot degli errori
figure
title("Error")

subplot(1,2,1)
plot(t, errors(1,:), "LineWidth", 2, DisplayName="X error")
hold on
plot(t, errors(2,:), "LineWidth", 2, DisplayName="Y error")
legend()
grid on

subplot(1,2,2)
plot(t, errors(3,:), "LineWidth", 2, DisplayName="Theta error")
legend()
grid on
```

```
%% Plot della traiettoria e delle configurazioni
startPoint = xe(1:2,1);
endPoint = xd(1:2);

figure
DrawRobot(firstDH, 1);
hold on
DrawRobot(DH)
p1 = plot(xe(1,:), xe(2,:), LineWidth=3, DisplayName="Traiettoria");
p2 = plot(startPoint(1), startPoint(2), Marker="o", MarkerSize=15, MarkerFaceColor="g", DisplayName="Start po
p3 = plot(endPoint(1), endPoint(2), Marker="o", MarkerSize=15, MarkerFaceColor="r", DisplayName="Set point");
realplot = [p1 p2 p3];
legend(realplot, "Traiettoria", "Start point", "Set point")
grid on
```

```
%% Variabili di giunto definitive

clc
disp("Le variabali di giunto da considerare sono:");
fprintf("\n");
disp(finalJointConfiguration);
```

Figura 7: Parte di codice sul plottaggio degli errori e sulla visualizzazione delle configurazioni assunte dal manipolatore

In questa parte del codice, vengono generate diverse figure per visualizzare i risultati della simulazione. Alla fine del ciclo, le variabili di giunto finali vengono convertite in gradi e stampate a schermo. Viene creata una figura con due grafici: il primo grafico mostra gli errori lungo gli assi X e Y rispetto al tempo mentre il secondo grafico mostra l'errore angolare rispetto al tempo.Quindi, questi grafici mostrano come gli errori cambiano durante la simulazione. Inoltre, vengono create due figure:la prima figura mostra la traiettoria percorsa dall'end-effector del robot nel suo spazio operativo. Include un punto verde per il punto di partenza e un punto rosso per il set point desiderato. La seconda figura visualizza il robot nelle configurazioni iniziali e finali delle articolazioni. Mostra come il robot si è mosso nel corso della simulazione. Infine, vi è la parte finale che stampa semplicemente le configurazioni finali delle variabili di giunto in gradi sulla console. Le variabili di giunto rappresentano l'angolazione delle articolazioni del robot alla fine della simulazione.

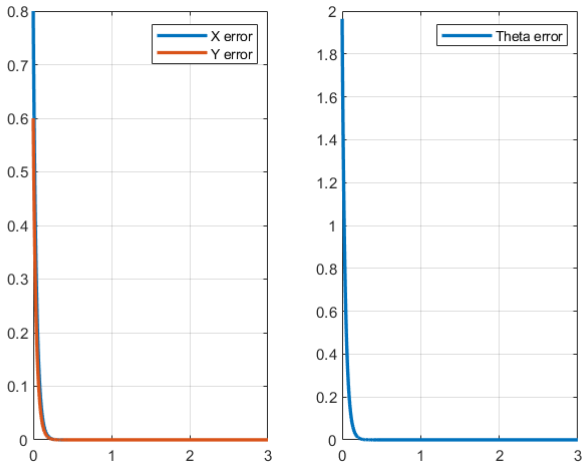


Figura 8: Plot dell'andamento degli errori di posizione e di orientamento

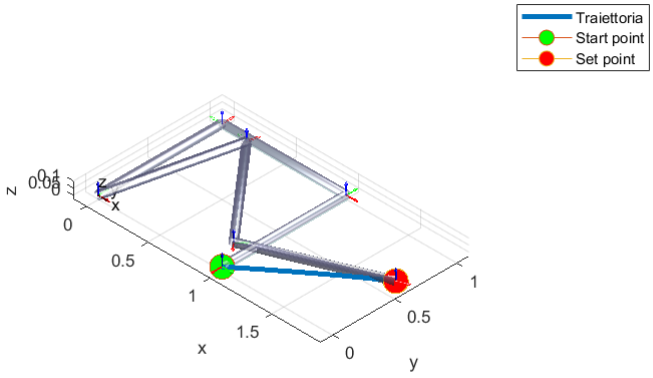


Figura 9: Visualizzazione delle configurazioni iniziale e finale e della traiettoria percorsa dallo *start point* al *set point*