PROGETTO AUTOMAZIONE E ROBOTICA

Antonio D’Ortona

Matricola 133857

PUNTO 1)

Dato il modello del robot Puma 560 è richiesto di calcolare le 6 matrici di rototraslazione necessarie per passare dal sistema di riferimento i-1 associato ad un link al sistema di riferimento i. Al fine di arrivare al risultato desiderato è stata definita una funzione in Matlab “matrice_rototraslazione” utile al calcolo di una generica matrice A.

```
function [A] = matrice_rototraslazione(alpha, a, theta, d)
A = [cos(theta), -sin(theta)*cos(alpha), sin(theta)*sin(alpha), a*cos(theta);
     sin(theta), cos(theta)*cos(alpha), -cos(theta)*sin(alpha), a*sin(theta);
     0, sin(alpha), cos(alpha), d;
     0, 0, 0, 1;];
```

Definisco i parametri DH:

```
alpha = [pi/2, 0, -pi/2, pi/2, -pi/2, 0];

a = [0, 0.4323, 0, 0, 0, 0];

d = [0, 0, 0.1501, 0.4331, 0, 0];

theta = [0, 0, 0, 0, 0, 0];
```

Si calcolano le 6 matrici di rototraslazione con la funzione mostrata in predecedenza, passando i parametri opportuni.

```
A_1 = matrice_rototraslazione(alpha(1), a(1), theta(1), d(1))

A_2 = matrice_rototraslazione(alpha(2), a(2), theta(2), d(2))

A_3 = matrice_rototraslazione(alpha(3), a(3), theta(3), d(3))

A_4 = matrice_rototraslazione(alpha(4), a(4), theta(4), d(4))

A_5 = matrice_rototraslazione(alpha(5), a(5), theta(5), d(5))

A_6 = matrice_rototraslazione(alpha(6), a(6), theta(6), d(6))
```

Le matrici calcolate sono:

```
A_1 =

    1.0000         0         0         0
         0    0.0000   -1.0000         0
         0    1.0000    0.0000         0
         0         0         0    1.0000


A_2 =

    1.0000         0         0    0.4323
         0    1.0000         0         0
         0         0    1.0000         0
         0         0         0    1.0000


A_3 =

    1.0000         0         0         0
         0    0.0000    1.0000         0
         0   -1.0000    0.0000    0.1501
         0         0         0    1.0000


A_4 =

    1.0000         0         0         0
         0    0.0000   -1.0000         0
         0    1.0000    0.0000    0.4331
         0         0         0    1.0000
```

```
A_5 =

    1.0000         0         0         0

         0    0.0000    1.0000         0

         0   -1.0000    0.0000         0

         0         0         0    1.0000


A_6 =

     1     0     0     0

     0     1     0     0

     0     0     1     0

     0     0     0     1
```

Una volta calcolate le matrici, per verificarne la correttezza, vado a moltiplicarle per ottenere la matrice T complessiva, in modo da poter effettuare il confronto utilizzando la funzione “fkine” fornita dal toolbox di Corke.

```
T = A_1 * A_2 * A_3 * A_4 * A_5 * A_6;

fkine(puma560, q)
```

dove q = theta, vettore nullo con dimensione 1x6. Il risultato della funzione fkine è:

```
ans =

         1         0         0    0.4323

         0         1         0   -0.1501

         0         0         1    0.4331

         0         0         0         1
```

La matrice T calcolata tramite la moltiplicazione delle matrici A è:

```
T =

    1.0000         0         0    0.4323

         0    1.0000         0   -0.1501

         0         0    1.0000    0.4331

         0         0         0    1.0000
```

PUNTO 2)

Viene richiesto di calcolare la matrice complessiva di rototraslazione T per diverse posizioni espresse nelle variabili di giunto. La matricola è 133857, quindi, considerando l'ultima cifra le diverse posizioni sono:

```
q1 = [0, 0, 0, 0, 0, 0];

q2 = [0, 0 -pi/7, 0, 0, 0];

q3 = [0, pi/7, -pi, 0, 0, 0];

q4 = [0, pi/4, -pi, 0, pi/7, 0];

q5 = [0, 0, -pi/2, pi/6, 0, -pi/7];

q6 = [0, -pi/7, 0, 0, -pi/4, 0];
```

Per calcolare la matrice T viene utilizzata la seguente funzione "calcola_T":

```
function [T] = calcola_T(alpha, a ,q ,d)

A_1 = matrice_rototraslazione(alpha(1), a(1), q(1), d(1));
A_2 = matrice_rototraslazione(alpha(2), a(2), q(2), d(2));
A_3 = matrice_rototraslazione(alpha(3), a(3), q(3), d(3));
A_4 = matrice_rototraslazione(alpha(4), a(4), q(4), d(4));
A_5 = matrice_rototraslazione(alpha(5), a(5), q(5), d(5));
A_6 = matrice_rototraslazione(alpha(6), a(6), q(6), d(6));

T = A_1 * A_2 * A_3 * A_4 * A_5 * A_6;
```

in cui vengono calcolate le matrici come nel punto 1 (con la funzione 'matrice_rototraslazione"), poi vengono moltiplicate per ottenere la matrice complessiva, che viene restituita. Per ogni posizione si calcola la matrice T con la funzione "calcola_T", passando come parametro il q opportuno, e si confronta il risultato restituito dalla funzione fkine. I parametri passati alla funzione calcola T sono i parametri DH mostrati nel punto 1.

```
q1 = [0, 0, 0, 0, 0, 0];

T1 = calcola_T(alpha, a ,q1, d);

T1

fkine_1 = fkine(puma560, q1)
```

```
T1 =

    1.0000         0         0    0.4323

         0    1.0000         0   -0.1501

         0         0    1.0000    0.4331

         0         0         0    1.0000



 fkine_1 =

         1         0         0    0.4323

         0         1         0   -0.1501

         0         0         1    0.4331

         0         0         0         1
```

Lo stesso avviene per le altre posizioni, ottenendo (vengono omesse le chiamate alle funzioni):

```
q2 = [0, 0 -pi/7, 0, 0, 0];
```

```
T2 =

    0.9010    0.0000    0.4339    0.6202

   -0.0000    1.0000   -0.0000   -0.1501

   -0.4339   -0.0000    0.9010    0.3902

         0         0         0    1.0000

 fkine_2 =

    0.9010         0    0.4339    0.6202

         0         1         0   -0.1501

   -0.4339         0    0.9010    0.3902
```

```
         0         0         0         1
```

```
q3 = [0, pi/7, -pi, 0, 0, 0];
```

```
T3 =

   -0.9010    0.0000    0.4339    0.5774

   -0.0000    1.0000   -0.0000   -0.1501

   -0.4339   -0.0000   -0.9010   -0.2026

         0         0         0    1.0000

fkine_3 =

   -0.9010         0    0.4339    0.5774

         0         1         0   -0.1501

   -0.4339         0   -0.9010   -0.2026

         0         0         0         1
```

```
q4 = [0, pi/4, -pi, 0, pi/7, 0];
```

```
T4 =

   -0.3303    0.0000    0.9439    0.6119

   -0.0000    1.0000   -0.0000   -0.1501

   -0.9439   -0.0000   -0.3303   -0.0006

         0         0         0    1.0000


fkine_4 =

   -0.3303         0    0.9439    0.6119

         0         1         0   -0.1501

   -0.9439         0   -0.3303-0.0005657

         0         0         0         1
```

```
q5 = [0, 0, -pi/2, pi/6, 0, -pi/7];
```

```
T5 =

    0.0000    0.0000    1.0000    0.8654
    0.0747    0.9972   -0.0000   -0.1501
   -0.9972    0.0747    0.0000    0.0000
         0         0         0    1.0000


 fkine_5 =

         0         0         1    0.8654
    0.0747    0.9972         0   -0.1501
   -0.9972    0.0747         0         0
         0         0         0         1
```

```
q6 = [0, -pi/7, 0, 0, -pi/4, 0];
```

```
T6 =

    0.3303    0.0000    0.9439    0.5774
   -0.0000    1.0000   -0.0000   -0.1501
   -0.9439   -0.0000    0.3303    0.2026
         0         0         0    1.0000


 fkine_6 =

    0.3303         0    0.9439    0.5774
         0         1         0   -0.1501
   -0.9439         0    0.3303    0.2026
         0         0         0         1
```

PUNTO 3)

I punti iniziali e finali sono rispettivamente:

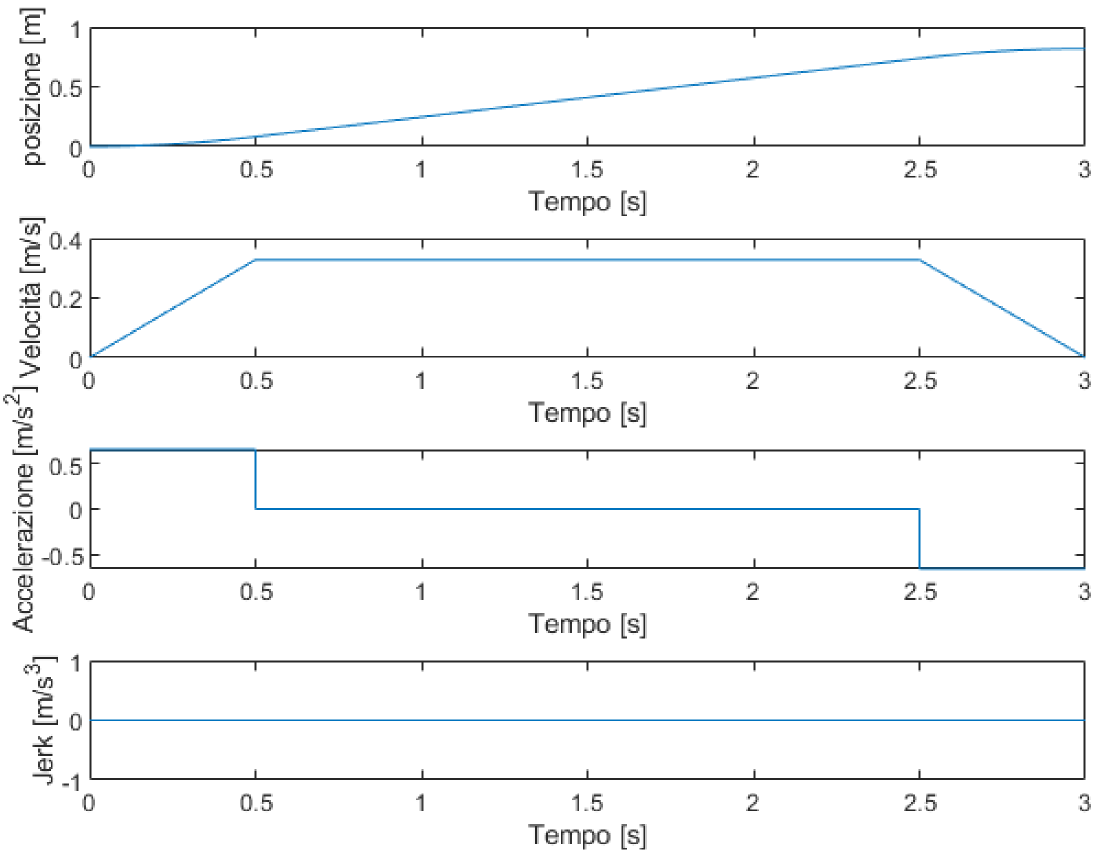
```
x_e_i = [0.05, -0.45, -0.05, 0, 0, 0]
x_e_f = [0.60, 0.15, 0.05, 0, 0, 0]
```

Calcolo dell'accelerazione nel moto trapezoidale:

$$\ddot{q}_v = \frac{q_f - q_i}{t_a(t_f - t_a)}$$

Utilizzo l'accelerazione appena calcolata per andare a pianificare la traiettoria nello spazio operativo con un profilo di moto trapezoidale.

Si vanno a generare i grafici per quanto riguarda la traiettoria nello spazio operativo generata con il moto trapezoidale:



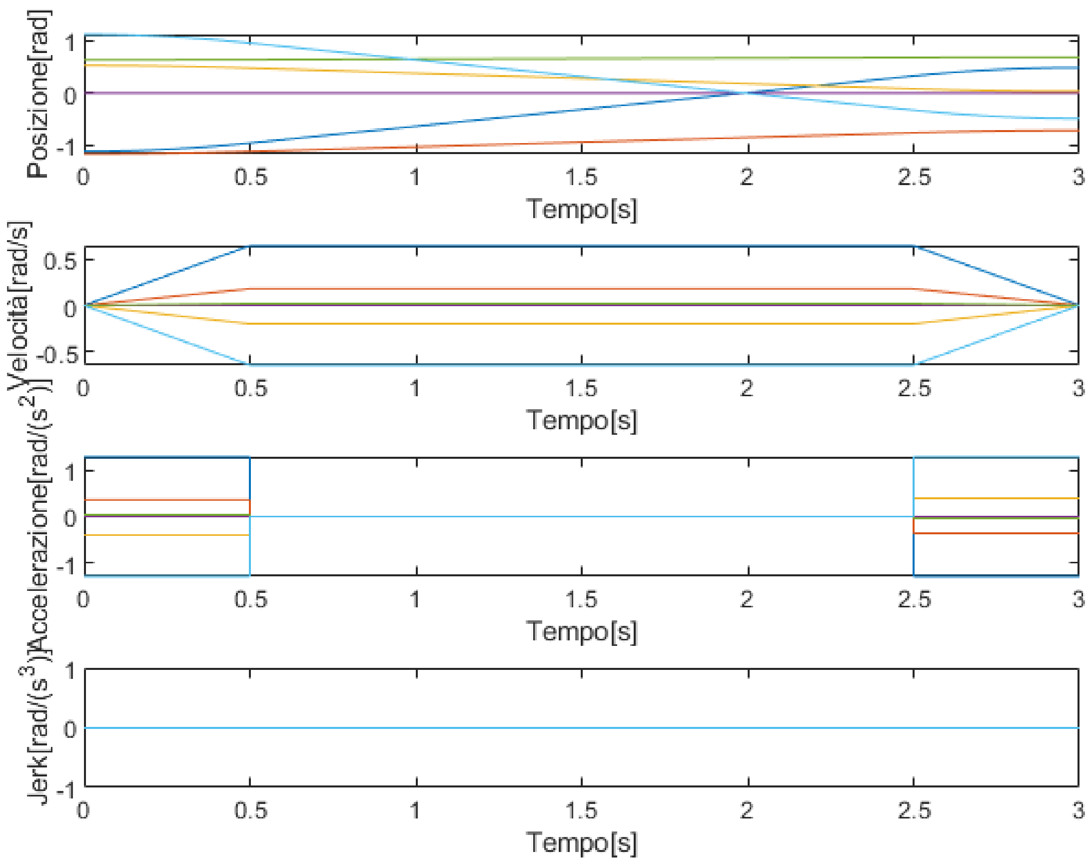
Il profilo di moto è trapezoidale nella velocità, infatti la funzione la legge di moto è stata generata andando a comporre due leggi di moto opportunamente “shiftate” nel tempo; è stato inoltre imposto il vincolo di continuità in ogni punto di raccordo. La durata è di 3 secondi ed è suddivisa come segue:

da 0s a 0.5s: moto uniformemente accelerato con accelerazione positiva

da 0.5s a 2.5s: moto rettilineo uniforme

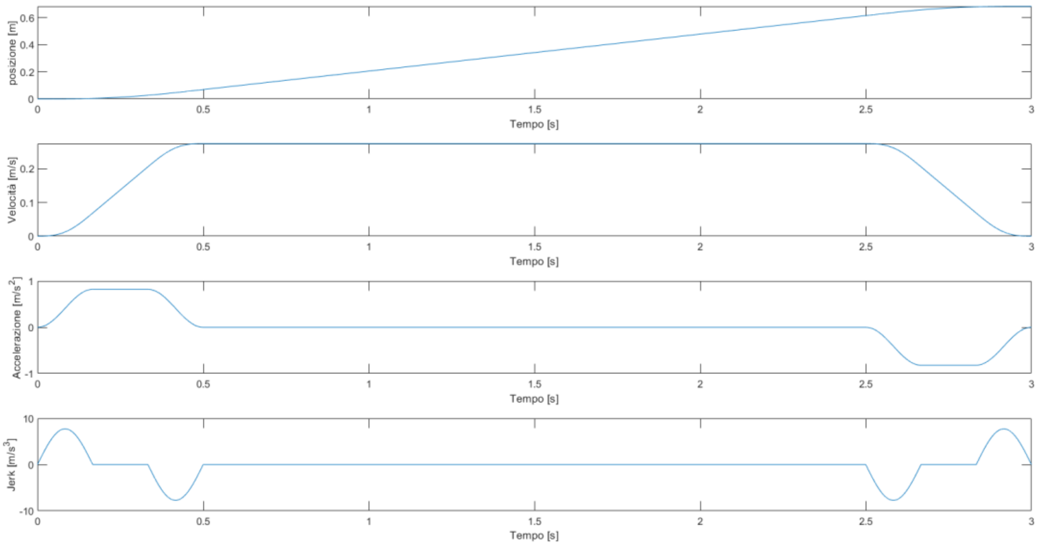
da 2.5s a 3s: moto uniformemente accelerato con la stessa accelerazione utilizzata nel primo tratto, ma con segno opposto.

In seguito si inseriscono la posa iniziale e finale in un vettore e si esegue la funzione della cinematica inversa con la funzione ikine. Si prendono la posizione iniziale e quella finale restituiti dalla funzione ikine e utilizzo la posizione iniziale e finale di ogni giunto nella funzione per il calcolo del moto trapezoidale.

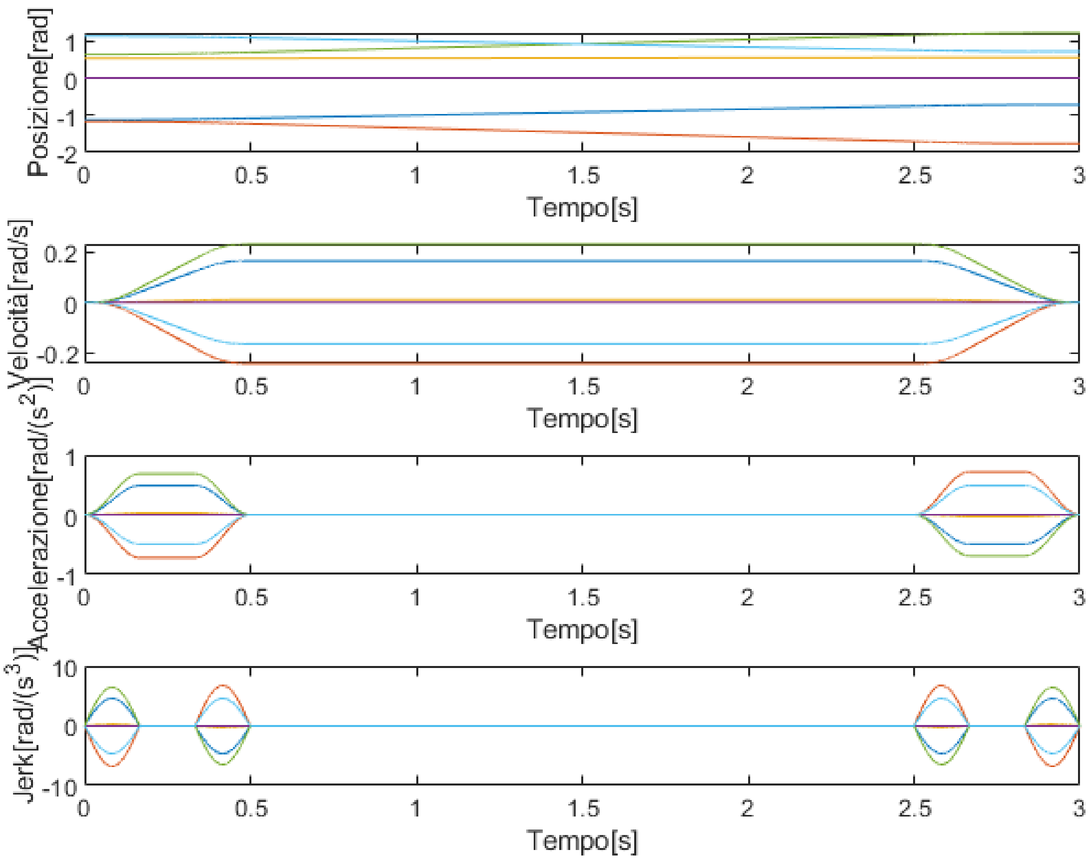


La stessa procedura è stata eseguita andando ad utilizzare il moto trapezoidale modificato costruito a lezione, di seguito i risultati.

Spazio operativo:



Spazio dei giunti:



PUNTO 4)

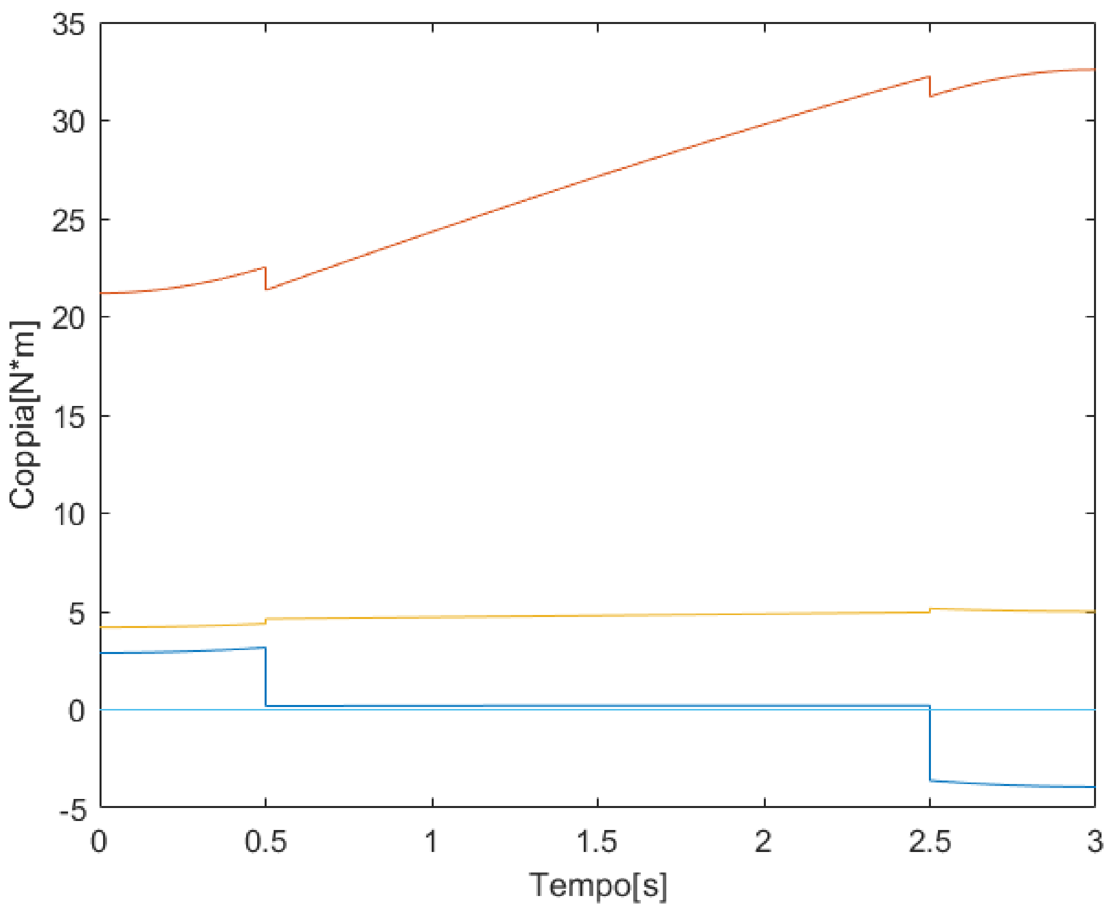
Dalla formula del modello dinamico del manipolatore si riesce a ricavarsi tau, ovvero le coppie applicate ai giunti. Le funzioni del toolbox di corke utilizzate sono:

- Puma560.inertia(): restituisce la matrice di inerzia di dimensione nxn
- Puma560.coriolis(): restituisce la matrice di Coriolis/effetti della forza centrifuga
- Puma560.friction(): restituisce un vettore contenente le forze di attrito ai giunti
- Puma560.gravoload(): restituisce il vettore contente l'effetto delle forza gravitazionali.

Quindi si calcolano le coppie come

```
for i=1:length(q_traj_p)
    tau(:,i) = (puma560_model.inertia(q_traj_p(:,i)') *
q_traj_a(:,i))' ...
    + (puma560_model.coriolis(q_traj_p(:,i)', q_traj_v(:,i)')
* q_traj_v(:,i))' ...
    + puma560_model.friction(q_traj_v(:,i)') .*
q_traj_v(:,i)' + puma560_model.gravload(q_traj_p(:,i)');
end
```

ottenendo:



PUNTO 5)

Viene richiesto di progettare un controllore PD + compensazione di gravità. La traiettoria da inseguire è la stessa del punto 3, mentre il punto di partenza è

```
q_start = [0.10, -0.45, -0.15, 0, 0, 0];
```

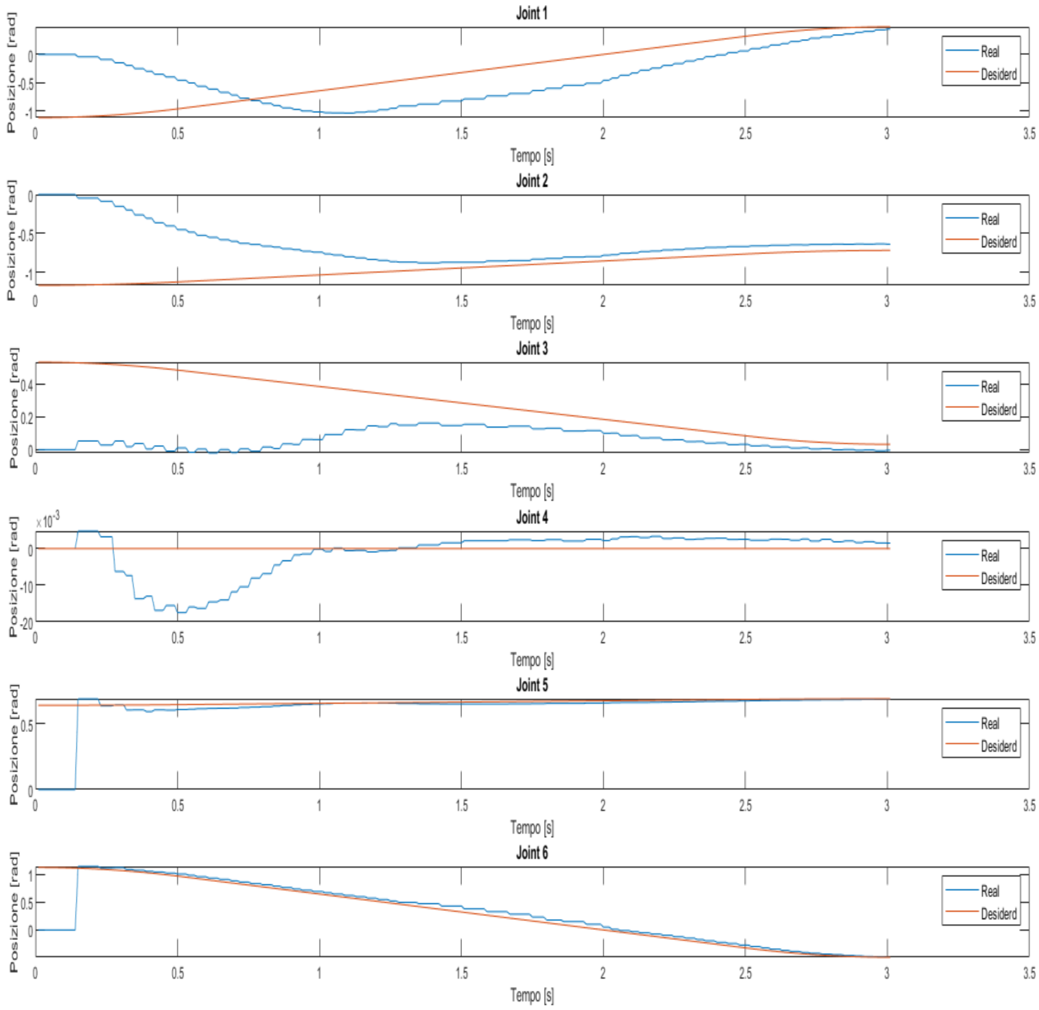
Dopo aver eseguito la cinemantica inversa, si definisce l’azione di controllo come:

```
controllore = Kp .* ep - Kd .* dq  + 
gravload(puma560_model,q);
```

Impostando Kp e KD come mostrato di seguito,

```
Kp = [250, 150, 150, 8, 10, 0.5];
Kd = [10, 15, 15, 0.2, 0.1, 0.005];
```

si ottengono i seguenti risultati per ogni giunto:



PUNTO 6)

Viene richiesto di progettare un controllore a dinamica inversa per risolvere lo stesso problema del punto 5.

La coppia tau è somma di due termini:

- La matrice di inerzia M moltiplicata per tau_dyn, dove tau_dyn è:

```
tau_dyn = Kp .* ep + Kd .* ed + ddq_d
```

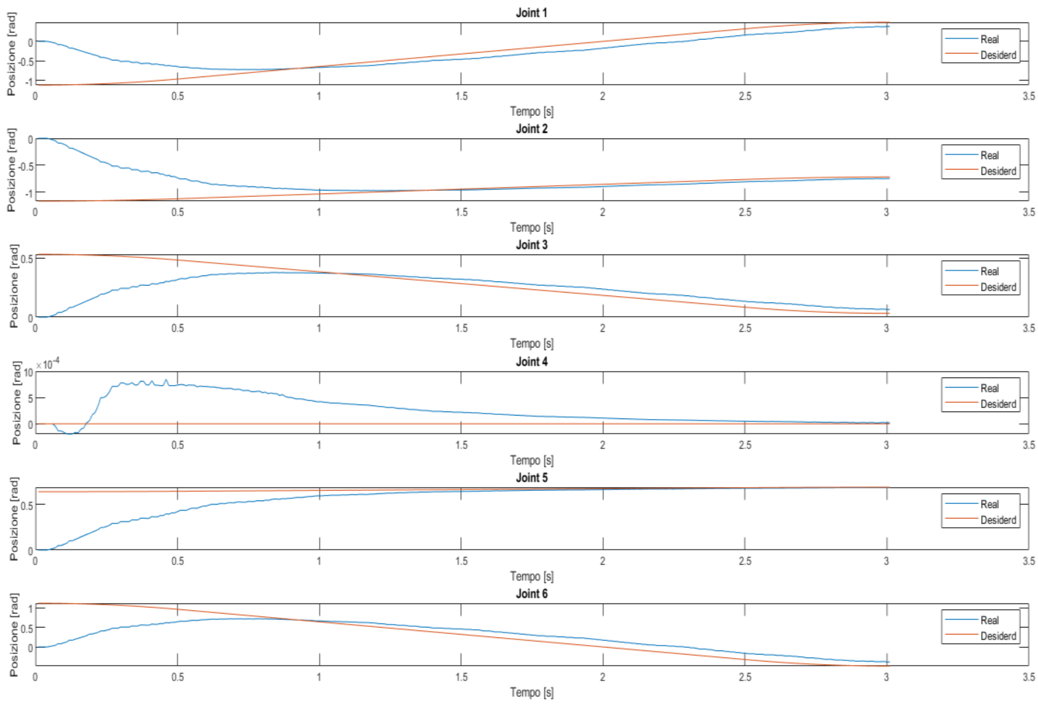
somma del termine proporzionale per l'errore di posizione, termine derivativo per l'errore di velocità e dell'accelerazione.

- Il termine n, ovvero la somma degli effetti di Coriolis e della forza centrifuga moltiplicati per la velocità dei giunti, le coppie di attrito moltiplicate per la velocità dei giunti e il termine gravitazionale.

Impostando Kp e Kd come:

```
Kp = [100, 100, 100, 100, 100, 100];
Kd = [20, 20, 20, 30, 20, 20];
```

Si ottengono i seguenti risultati:



Nota: Effettuando diverse prove si è notato che con gli stessi valori ci sono casi in cui appaiono delle oscillazioni nel transitorio.