Elaborato

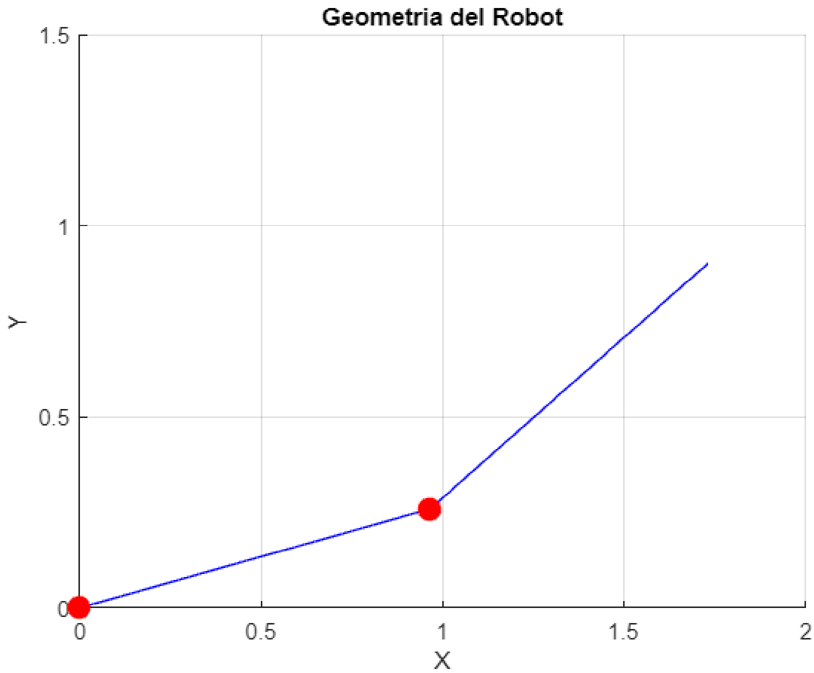
Lazzaro Francesco 220810

Traccia

Si vuole progettare il modello dinamico di un Robot industriale così composto:

- 2 bracci L_1 ed L_2 entrambi di lunghezza $1\ m$
- 2 giunti rotoidali, q_1 che collega L_1 alla base del robot e q_2 che collega L_1ed L_2
- L_1 ed L_2 hanno una massa pari ad $1\ \mathrm{Kg}$

```
% Lunghezza dei segmenti
lunghezza = 1;

% Angoli dei segmenti (in radianti)
angolo_l1 = deg2rad(15); % Angolo del primo segmento (rispetto all'asse x)
angolo_l2 = deg2rad(40); % Angolo del secondo segmento (rispetto all'asse x)

% Calcolo delle coordinate dei punti terminali dei segmenti
x_origine = 0;
y_origine = 0;

x_l1 = lunghezza * cos(angolo_l1);
y_l1 = lunghezza * sin(angolo_l1);

x_l2 = x_l1 + lunghezza * cos(angolo_l2);
y_l2 = y_l1 + lunghezza * sin(angolo_l2);

% Disegno dei segmenti
figure;
hold on;
plot([x_origine, x_l1], [y_origine, y_l1], 'b'); % Disegna il primo segmento
plot([x_l1, x_l2], [y_l1, y_l2], 'b'); % Disegna il secondo segmento

% Segna i punti di contatto
plot(x_origine, y_origine, 'ro', 'MarkerSize', 10, 'MarkerFaceColor', 'r'); % Origine
plot(x_l1, y_l1, 'ro', 'MarkerSize', 10, 'MarkerFaceColor', 'r'); % Estremità del primo
segmento

% Impostazioni dell'asse
axis([0 2 0 1.5]); % Imposta i limiti dell'asse x e y positivi
xlabel('X');
ylabel('Y');
title('Geometria del Robot');

% Mostra la griglia
grid on;
```

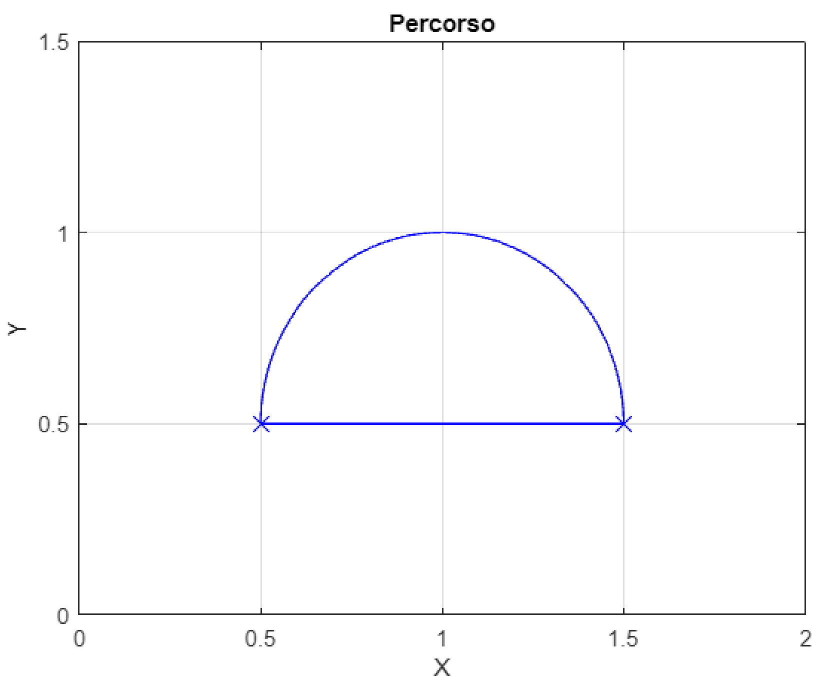


Il robot deve seguire un percorso così composto:

- Dal punto $P_1 = (0.5, 0.5)$ al tempo $t = 0$ si sposta nel punto $P_2 = (1.5, 0.5)$ al tempo $t = 10s$ seguendo un percorso a minima distanza
- Rimane fermo per $5s$
- Dal punto P_2 al tempo $t = 15s$ si sposta nel punto P_1 al tempo $t = 30s$ seguendo un percorso semicircolare con centro in $c = (1, 0.5)$ e raggio $r = 0.5$

```
% Definisci il centro e il raggio del semicerchio
figure;
centro = [1, 0.5];
raggio = 0.5;

% Calcola gli angoli iniziali e finali per disegnare il semicerchio
theta_iniziale = 0;
theta_finale = pi;

% Genera un vettore di angoli per disegnare il semicerchio
theta = linspace(theta_iniziale, theta_finale, 100);

% Calcola le coordinate x e y dei punti lungo il semicerchio
x = centro(1) + raggio * cos(theta);
y = centro(2) + raggio * sin(theta);

% Definisci i punti P1 e P2
p1 = [0.5, 0.5];
p2 = [1.5, 0.5];

% Disegna il semicerchio e i punti P1 e P2
plot(x, y, 'b');
```

```
hold on;
plot(p1(1), p1(2), 'bx', 'MarkerSize', 10);
plot(p2(1), p2(2), 'bx', 'MarkerSize', 10);

% Disegna il segmento che collega P1 e P2
plot([p1(1), p2(1)], [p1(2), p2(2)], 'b', 'LineWidth', 1);

axis([0 2 0 1.5]);
grid on;
xlabel('X');
ylabel('Y');
title('Percorso');
```



L'obiettivo è generare un modello che minimizzi l'errore di posizione rispetto al percorso

Percorso a minima distanza 1)

Legge oraria e Generatore di riferimento

Parametrizzando l'equazione che descrive la traiettoria si ottiene $P(\lambda) = P_1 + \lambda(P_2 - P_1) = \begin{bmatrix} x(\lambda) \\ y(\lambda) \end{bmatrix}$ con $0 \leq \lambda \leq 1$

$\lambda = \lambda(t)$ e inoltre si può riscrivere l'equazione come $P(\lambda(\sigma)) = P_1 + \lambda(\sigma)(P_2 - P_1)$ con $\sigma = \dfrac{t - T_1}{T_2 - T_1}$

Si rende necessario per ottenere un andamento realistico della posizione e della velocità usare una funzione λdel tipo $\lambda(\sigma) = -2\sigma^3 + 3\sigma^2$.

Sviluppando le equazioni si ottengono:

$$P(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} P_{1\,x} + \left(-2\left(\dfrac{t-T_1}{T_2-T_1}\right)^3 + 3\left(\dfrac{t-T_1}{T_2-T_1}\right)^2\right)(P_{2\,x} - P_{1\,x}) \\ P_{1\,y} \end{bmatrix}$$ da cui

$$\dot{P(t)} = \begin{bmatrix} \dot{x(t)} \\ \dot{y(t)} \end{bmatrix} = \begin{bmatrix} \dfrac{1}{T_2 - T_1}\left(-6\left(\dfrac{t-T_1}{T_2-T_1}\right)^2 + 6\left(\dfrac{t-T_1}{T_2-T_1}\right)\right)(P_{2\,x} - P_{1\,x}) \\ 0 \end{bmatrix}$$

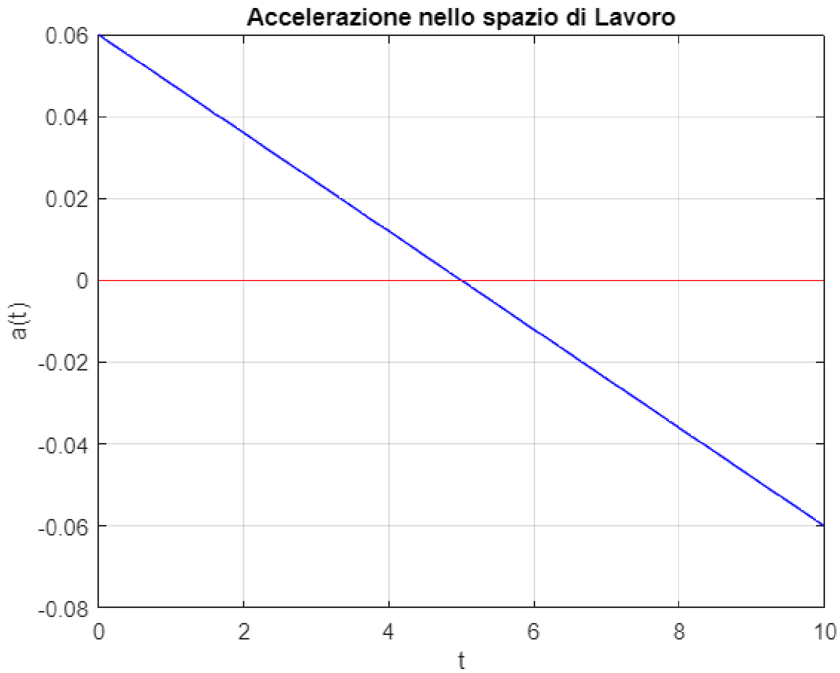
(per le successive rappresentazioni: colore blu primo braccio, colore rosso secondo braccio)

```
% Definizione dei punti P1 e P2
P1 = [0.5, 0.5];
P2 = [1.5, 0.5];

% Definizione dei tempi T1 e T2
T1 = 0;
T2 = 10;

% Definizione delle lunghezze L1 e L2
L1 = 1;
L2 = 1;

% Generazione di valori di t da 0 a 10
t = linspace(T1, T2, 100); % Esempio con 100 valori

%Generazione lambda e sigma
sigma = (t - T1) / (T2 - T1);

lambda=-2*sigma.^3+3*sigma.^2;

%% Calcoli delle funzioni spazio e velocità per ogni valore di t
% Inizializzazione dei vettori per i risultati
x = zeros(size(t)); y = zeros(size(t));
vx = zeros(size(t)); vy = zeros(size(t));
```

```
ax = zeros(size(t)); ay = zeros(size(t));
c2 = zeros(size(t)); s2 = zeros(size(t));
c1 = zeros(size(t)); s1 = zeros(size(t));
q1 = zeros(size(t)); q2 = zeros(size(t));
dq1 = zeros(size(t)); dq2 = zeros(size(t));
ddq1 = zeros(size(t)); ddq2 = zeros(size(t));

% Calcolo dei valori per ogni valore di t
for i = 1:numel(t)
    % Calcoli delle funzioni posizione
    x(i) = P1(1) + lambda(i) * (P2(1) - P1(1));
    y(i) = P1(2);

    % Calcoli delle funzioni velocità
    vx(i) = ((-6*sigma(i)^2) +6*sigma(i))*(1 / (T2 - T1)) * (P2(1) - P1(1));
    vy(i) = 0;

    % Calcolo accelerazione
    ax(i) = ((-12*sigma(i)) +6)*(1 / (T2 - T1))^2 * (P2(1) - P1(1));
    ay(i) = 0;

    % Cinematica inversa
    %Calcolo di q2
    c2(i) = (x(i)^2 + y(i)^2 - L1^2 - L2^2) / (2 * L1 * L2);
    s2(i) = sqrt(1 - c2(i)^2);
    q2(i) = atan2(s2(i), c2(i));

    % Calcolo di q1
    A=[L1+L2*c2(i) -L2*s2(i);
    L2*s2(i) (L1+L2*c2(i))];
    Q1=(inv(A))*[x(i);y(i)];
    c1(i)=Q1(1);
    s1(i)=Q1(2);
    q1(i)=atan2(s1(i),c1(i));
end

figure; plot(t,x,'b',t,y,'r');
grid on;
xlabel('t');
ylabel('P(t)');
title('Posizione Spazio di Lavoro');
```


```
figure; plot(t,vx,'b',t,vy,'r');
grid on;
xlabel('t');
ylabel('v(t)');
title('Velocità nello spazio di Lavoro');
```


```
figure; plot(t,ax,'b',t,ay,'r');
grid on;
xlabel('t');
ylabel('a(t)');
title('Accelerazione nello spazio di Lavoro');
```



Cinematica inversa

Nella cinematica diretta le equazioni che legano $P\ e\ Q$ sono

$$P_x = L_1\,c_1 + L_2\,c_{12}$$

$$P_y = L_1\,s_1 + L_2\,s_{12}$$

Quadrando e sommando le equazioni si ottiene

$$P_x^2 + P_y^2 - \left(L_1^2 + L_2^2\right) = 2L_1L_2(c_1c_{12} + s_1s_{12}) = 2L_1L_2c_2$$

Da cui si ricavano c_2 ed s_2

$$c_2 = \frac{P_x^2 + P_y^2 - L_1^2 - L_2^2}{2L_1L_2} \qquad s_2 = \pm\sqrt{1 - (c_2)^2}$$

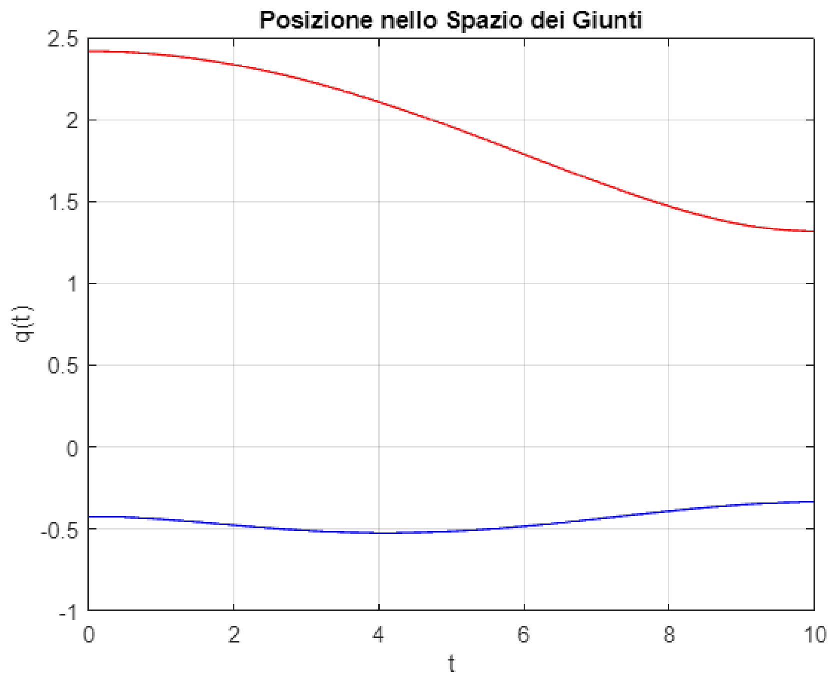
Pertanto si può stabilire il valore di q_2 come $q_2 = \mathrm{ATAN2}(s_2, c_2)$

q_1 si può ricavare da q_2 per ispezione geometrica, oppure risolvendo invece il sistema di equazioni della cinematica diretta per c_1 ed s_1

$$Q_1 = A^{-1}P \quad \rightarrow q_1 = \mathrm{ATAN2}(s_1, c_1)$$

oppure $q_1 = \mathrm{ATAN2}(P_y, P_x) - \mathrm{ATAN2}(L_2s_2, L_1 + L_2c_2)$

```
figure; plot(t,q1,'b',t,q2,'r');
grid on;
xlabel('t');
ylabel('q(t)');
title('Posizione nello Spazio dei Giunti');
```



Cinematica differenziale inversa

Dalla cinematica differenziale diretta l'equazione che descrive il legame tra $\dot{P} = v$ e $\dot{Q}$ è

$$\dot{P} = J(Q)\dot{Q} \quad \text{dove } J = \begin{bmatrix} -L_1 s_1 - L_2 s_{12} & -L_2 s_{12} \\ L_1 c_1 + L_2 c_{12} & L_2 c_{12} \end{bmatrix}$$

Lo Jacobiano J è una matrice che descrive come le velocità dei giunti influenzano le velocità di un punto di interesse nello spazio cartesiano.

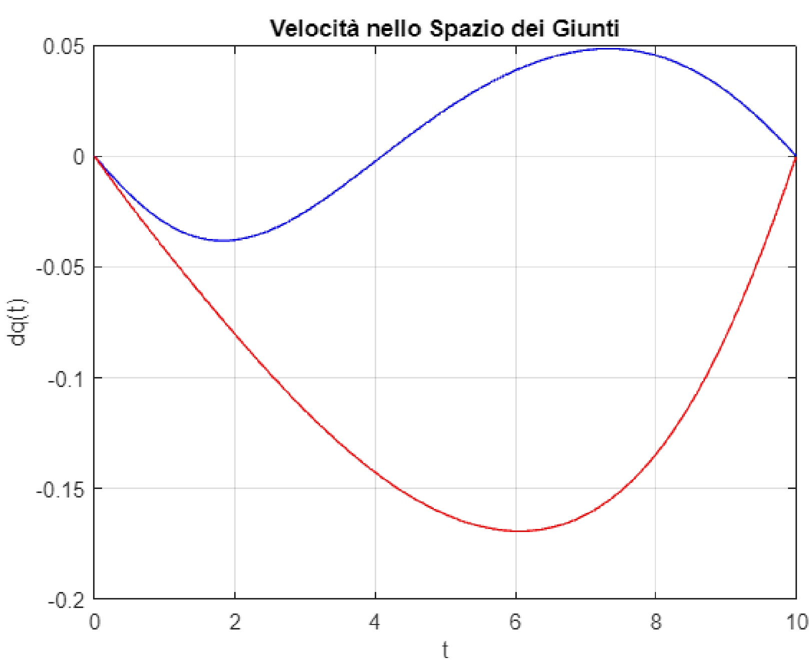
J è la matrice delle derivate parziali che collega la velocità delle coordinate articolari $\dot{q}$ alla velocità delle coordinate cartesiane v.

Se J ha determinante diverso da zero (invertibile) si può quindi scrivere

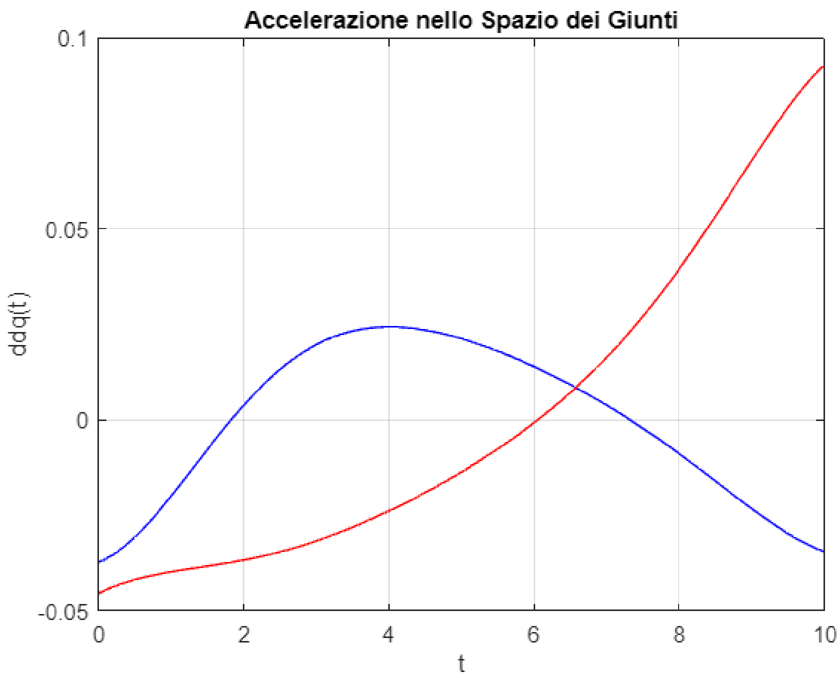
$$\dot{Q} = J^{-1}(Q)\dot{P}$$

```
% Cinematica differenziale inversa

for i = 1:numel(t)
    % Calcolo della matrice J
    J = [-L1*sin(q1(i)) - L2*sin(q1(i)+q2(i)), -L2*sin(q1(i)+q2(i));
         L1*cos(q1(i)) + L2*cos(q1(i)+q2(i)), L2*cos(q1(i)+q2(i))];

    % Inversione della matrice J
    J_inv = inv(J);
    v=[vx(i); vy(i)];

    % Moltiplicazione della matrice invertita per il vettore di velocità
    dQ = (J_inv )* v;
```

```
    % Estrazione dei valori di dq1 e dq2
    dq1(i) = dQ(1, :);
    dq2(i) = dQ(2, :);

    % Accelerazione
    Jp = [-L1*c1(i)*dq1(i)-L2*cos(q1(i)+q2(i))*(dq1(i)+dq2(i)), -
L2*cos(q1(i)+q2(i))*(dq1(i)+dq2(i));
        -L1*s1(i)*dq1(i)-L2*sin(q1(i)+q2(i))*(dq1(i)+dq2(i)), -
L2*sin(q1(i)+q2(i))*(dq1(i)+dq2(i))];

    J = [-L1*s1(i)-L2*sin(q1(i)+q2(i)), -L2*sin(q1(i)+q2(i));
        L1*c1(i)+L2*cos(q1(i)+q2(i)), L2*cos(q1(i)+q2(i))];
    b = [ax(i);ay(i)] - Jp*[dq1(i);dq2(i)];
    Qpp = (inv(J))*b;

     ddq1(i) = Qpp(1, :);
    ddq2(i) = Qpp(2, :);

end

figure; plot(t,dq1,'b',t,dq2,'r');
grid on;
xlabel('t');
ylabel('dq(t)');
title('Velocità nello Spazio dei Giunti');
```




```
figure; plot(t,ddq1,'b',t,ddq2,'r');
grid on;
xlabel('t');
ylabel('ddq(t)');
title('Accelerazione nello Spazio dei Giunti');
```



Percorso semicirconferenza 2)

Legge Oraria e Generatore di riferimento

Le coordinate del centro sono $P_{\text{centro}} = \dfrac{P_2 - P_1}{2}$

Parametrizzando l'equazione che descrive la traiettoria si ottiene $P(\theta) = \begin{bmatrix} x_c \\ y_c \end{bmatrix} + \begin{bmatrix} R\cos(\theta) \\ R\sin(\theta) \end{bmatrix} = \begin{bmatrix} x(\theta) \\ y(\theta) \end{bmatrix} \text{con } 0 \le \theta \le \pi$

$\theta = \theta(t)$ si può riscrivere l'equazione come $P(\theta(\sigma)) = P_{\text{centro}} + R \begin{bmatrix} \cos(\theta(\sigma)) \\ \sin(\theta(\sigma)) \end{bmatrix} \text{ con } \sigma = \dfrac{t - T_3}{T_4 - T_3}$

$\theta(\sigma) = \theta_1 + \lambda(\sigma)(\theta_2 - \theta_1) = \lambda(\sigma)\pi$

Si rende necessario per ottenere un andamento realistico della posizione e della velocità usare una funzione λdel tipo $\lambda(\sigma) = -2\sigma^3 + 3\sigma^2$

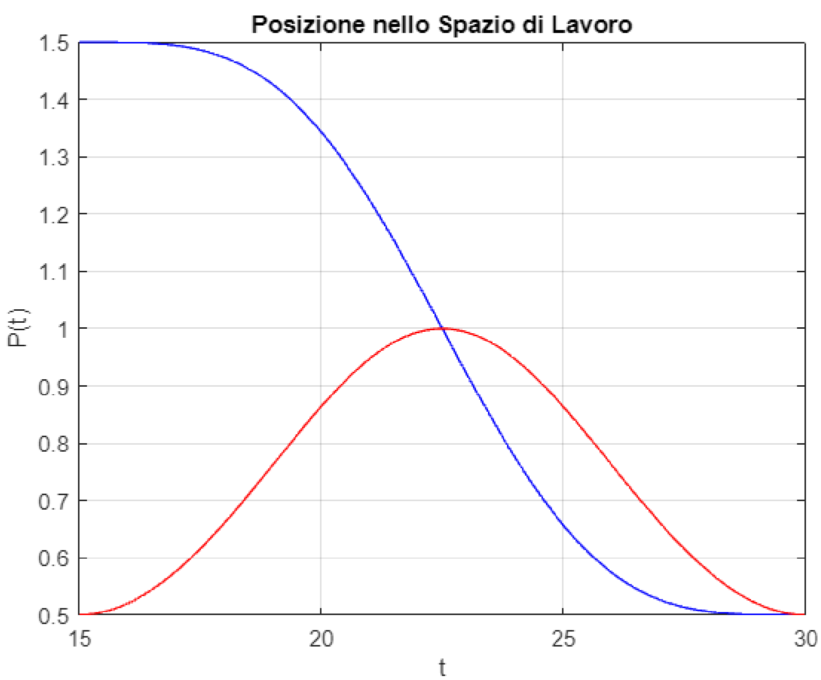
Sviluppando le equazioni si ottengono:

$$P(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} x_c + R\cos\left(\pi\left(-2\left(\dfrac{t-T_3}{T_4-T_3}\right)^3 + 3\left(\dfrac{t-T_3}{T_4-T_3}\right)^2\right)\right) \\ y_c + R\sin\left(\pi\left(-2\left(\dfrac{t-T_3}{T_4-T_3}\right)^3 + 3\left(\dfrac{t-T_3}{T_4-T_3}\right)^2\right)\right) \end{bmatrix}$$ da cui

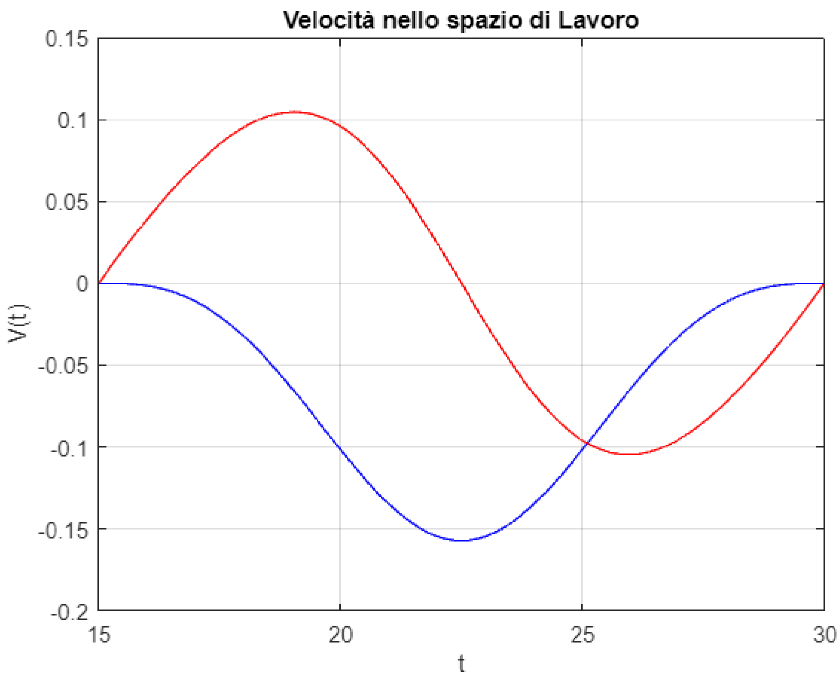
$$\dot{P(t)} = \begin{bmatrix} \dot{x(t)} \\ \dot{y(t)} \end{bmatrix} = \begin{bmatrix} -\dfrac{R\pi}{T_4 - T_3} * \left(-6\left(\dfrac{t-T_1}{T_2-T_1}\right)^2 + 6\left(\dfrac{t-T_1}{T_2-T_1}\right)\right) * \sin\left(\pi\left(-2\left(\dfrac{t-T_3}{T_4-T_3}\right)^3 + 3\left(\dfrac{t-T_3}{T_4-T_3}\right)^2\right)\right) \\ \dfrac{R\pi}{T_4 - T_3} * \left(-6\left(\dfrac{t-T_1}{T_2-T_1}\right)^2 + 6\left(\dfrac{t-T_1}{T_2-T_1}\right)\right) * \cos\left(\pi\left(-2\left(\dfrac{t-T_3}{T_4-T_3}\right)^3 + 3\left(\dfrac{t-T_3}{T_4-T_3}\right)^2\right)\right) \end{bmatrix}$$

```
% Definizione del centro
Pc = [1, 0.5];

% Definizione del raggio
R = 0.5;

% Definizione dei tempi T3 e T4
T3 = 15;
T4 = 30;

% Generazione di valori di t da 15 a30
s_t = linspace(T3, T4, 150); % Esempio con 150 valori

%Generazione lambda e sigma
s_sigma= (s_t - T3) / (T4 - T3);

s_lambda=pi*(-2*s_sigma.^3+3*s_sigma.^2);

%% Calcoli delle funzioni spazio e velocità per ogni valore di t
% Inizializzazione dei vettori per i risultati
s_x = zeros(size(s_t)); s_y = zeros(size(s_t));
s_vx = zeros(size(s_t)); s_vy = zeros(size(s_t));
s_ax = zeros(size(s_t)); s_ay = zeros(size(s_t));
s_c2 = zeros(size(s_t)); s_s2 = zeros(size(s_t));
s_c1 = zeros(size(s_t)); s_s1 = zeros(size(s_t));
s_q2 = zeros(size(s_t)); s_q1 = zeros(size(s_t));
s_dq2 = zeros(size(s_t)); s_dq1 = zeros(size(s_t));
s_ddq2 = zeros(size(s_t)); s_ddq1 = zeros(size(s_t));

% Calcolo dei valori per ogni valore di t
for i = 1:numel(s_t)
    % Calcoli delle funzioni spazio
    s_x(i) = Pc(1) + R*cos(s_lambda(i));
    s_y(i) = Pc(2) + R*sin(s_lambda(i));

    s_P=[s_x(i);s_y(i)];

    % Calcoli delle funzioni velocità
    s_vx(i) = (-R*pi/ (T4 - T3)) * sin(s_lambda(i)) * ((-6*s_sigma(i)^2) +6*s_sigma(i));
    s_vy(i) = (R*pi/ (T4 - T3)) * cos(s_lambda(i)) * ((-6*s_sigma(i)^2) +6*s_sigma(i));

    % Calcolo accelerazione
    s_ax(i) = -R* ((pi/ (T4 - T3))^3) * sin(s_lambda(i)) * cos(s_lambda(i)) * (((-
6*s_sigma(i)^2) +6*s_sigma(i))^2) * (-12*s_sigma(i)+6);
    s_ay(i) = -R* ((pi/ (T4 - T3))^3) * sin(s_lambda(i)) * cos(s_lambda(i)) * (((-
6*s_sigma(i)^2) +6*s_sigma(i))^2) * (-12*s_sigma(i)+6);

    % Cinematica inversa
    % Calcolo di q2
    s_c2(i) = (s_x(i)^2 + s_y(i)^2 - L1^2 - L2^2) / (2 * L1 * L2);
    s_s2(i) = sqrt(1 - s_c2(i)^2);
```

```
    s_q2(i) = atan2(s_s2(i), s_c2(i));

    % Calcolo di q1
    A=[L1+L2*s_c2(i) -L2*s_s2(i);
    L2*s_s2(i) (L1+L2*s_c2(i))];
    s_Q1=(inv(A))*s_P;
    s_c1(i)=s_Q1(1);
    s_s1(i)=s_Q1(2);
    s_q1(i)=atan2(s_s1(i),s_c1(i));
end

figure; plot(s_t,s_x,'b',s_t,s_y,'r');
grid on;
xlabel('t');
ylabel('P(t)');
title('Posizione nello Spazio di Lavoro');
```




```
figure; plot(s_t,s_vx,'b',s_t,s_vy,'r');
grid on;
xlabel('t');
ylabel('V(t)');
title('Velocità nello spazio di Lavoro');
```




```
figure; plot(s_t,s_ax,'b',s_t,s_ay,'r');
grid on;
xlabel('t');
ylabel('a(t)');
title('Accelerazione nello spazio di Lavoro');
```

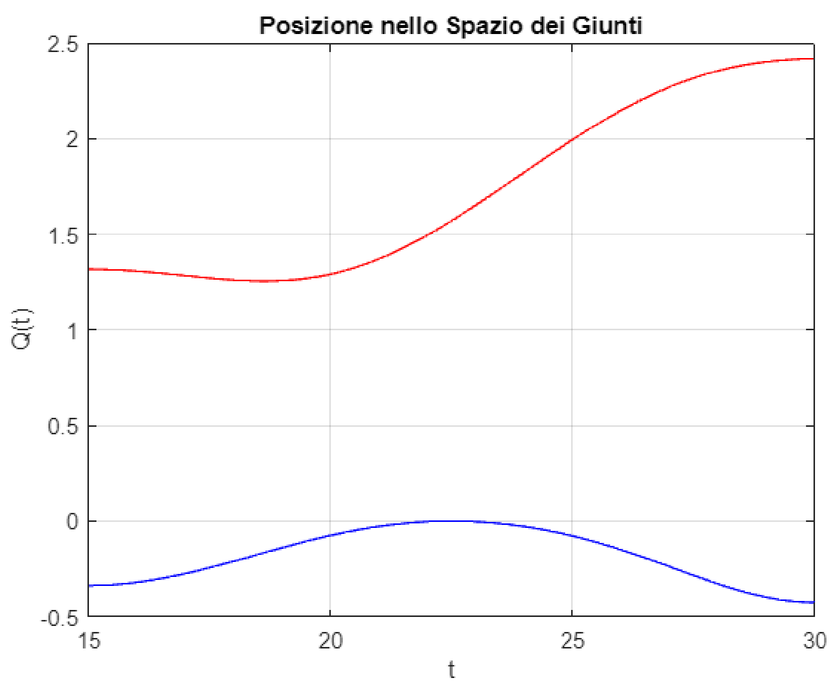
Cinematica inversa

Il procedimento è analogo a quello illustrato nella sezione 1)

$$c_2 = \frac{P_x^2 + P_y^2 - L_1^2 - L_2^2}{2L_1L_2} \qquad s_2 = \pm\sqrt{1-(c_2)^2} \quad \rightarrow\ q_2 = \mathrm{ATAN2}(s_2, c_2)$$

$$Q_1 = A^{-1}P \quad \rightarrow q_1 = \mathrm{ATAN2}(s_1, c_1)$$

```
figure; plot(s_t,s_q1,'b',s_t,s_q2,'r');
grid on;
xlabel('t');
ylabel('Q(t)');
title('Posizione nello Spazio dei Giunti');
```



Cinematica differenziale inversa

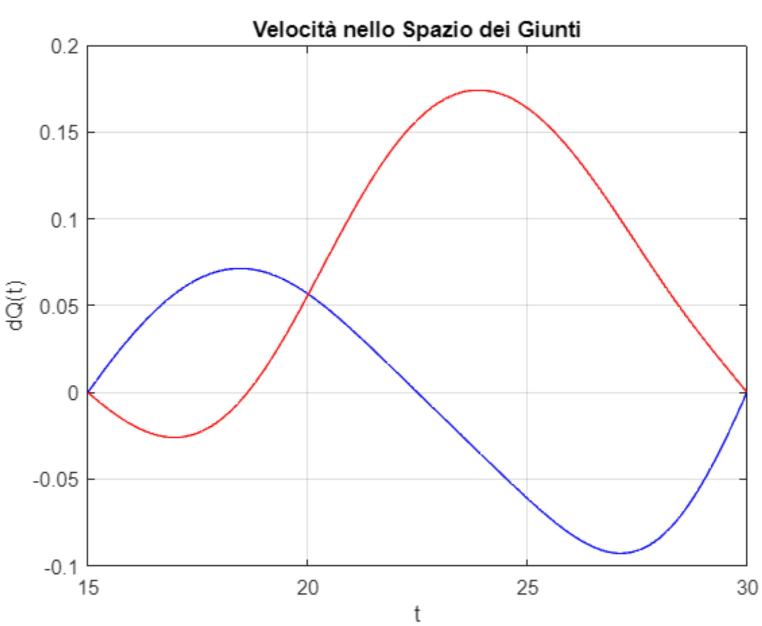
Il procedimento è analogo a quello illustrato nella sezione 1)

$\dot{P} = J(Q)\dot{Q}$ dove $J = \begin{bmatrix} -L_1s_1 - L_2s_{12} & -L_2s_{12} \\ L_1c_1 + L_2c_{12} & L_2c_{12} \end{bmatrix}$ il cui determinante se diverso da zero permette di scrivere

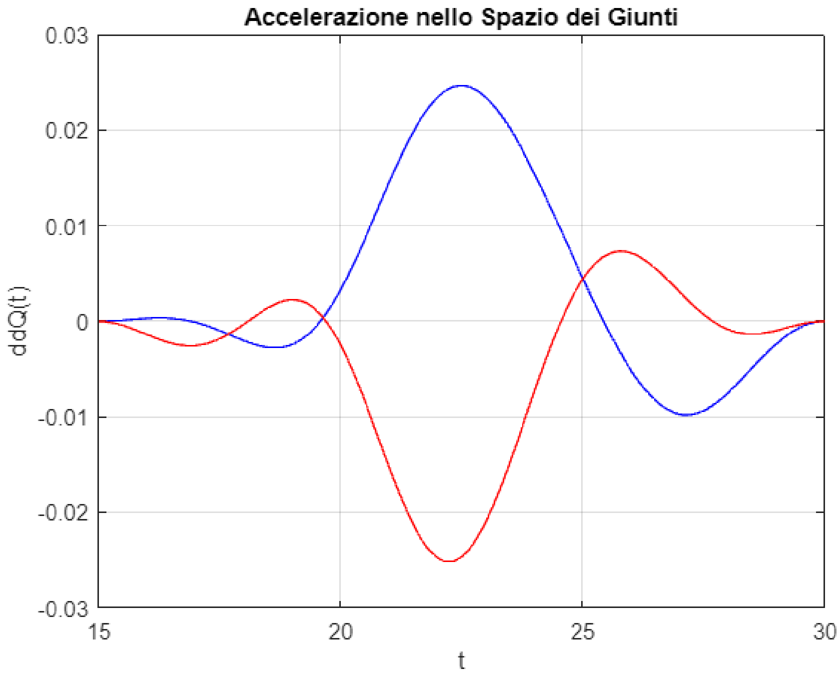
$$\dot{Q} = J^{-1}(Q)\dot{P}$$

```
for i = 1:numel(s_t)
% Calcolo della matrice J
    s_J = [-L1*sin(s_q1(i)) - L2*sin(s_q1(i)+s_q2(i)), -L2*sin(s_q1(i)+s_q2(i));
           L1*cos(s_q1(i)) + L2*cos(s_q1(i)+s_q2(i)), L2*cos(s_q1(i)+s_q2(i))];
```

```
    % Inversione della matrice J
    s_J_inv = inv(s_J);

    % Moltiplicazione della matrice invertita per il vettore di velocità
    s_dq = (s_J_inv) * [s_vx(i), s_vy(i)]';

    % Estrazione dei valori di dq1 e dq2
    s_dq1(i) = s_dq(1, :);
    s_dq2(i) = s_dq(2, :);

    % Accelerazione
    Jp = [-L1*s_c1(i)*s_dq1(i)-L2*cos(s_q1(i)+s_q2(i))*(s_dq1(i)+s_dq2(i)), -
L2*cos(s_q1(i)+s_q2(i))*(s_dq1(i)+s_dq2(i));
        -L1*s_s1(i)*s_dq1(i)-L2*sin(s_q1(i)+s_q2(i))*(s_dq1(i)+s_dq2(i)), -
L2*sin(s_q1(i)+s_q2(i))*(s_dq1(i)+s_dq2(i))];

    J = [-L1*s_s1(i)-L2*sin(s_q1(i)+s_q2(i)), -L2*sin(s_q1(i)+s_q2(i));
        L1*s_c1(i)+L2*cos(s_q1(i)+s_q2(i)), L2*cos(s_q1(i)+s_q2(i))];
    b = [s_ax(i);s_ay(i)] - Jp*[s_dq1(i);s_dq2(i)];
    Qpp = (inv(J))*b;

     s_ddq1(i) = Qpp(1, :);
    s_ddq2(i) = Qpp(2, :);

end

figure;
plot(s_t,s_dq1,'b', s_t,s_dq2, 'r');
grid on;
xlabel('t');
ylabel('dQ(t)');
title('Velocità nello Spazio dei Giunti');
```



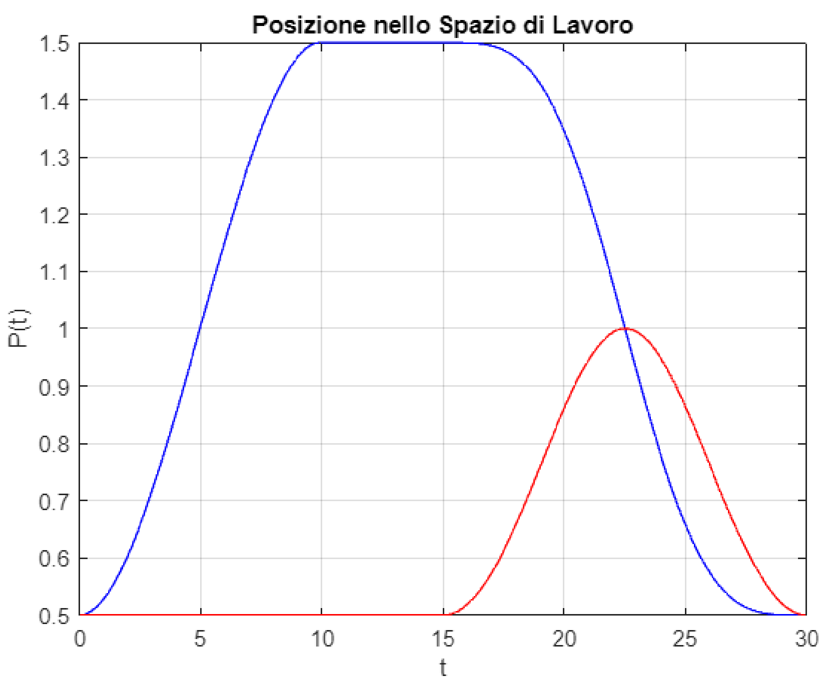
```
figure;
plot(s_t,s_ddq1,'b', s_t,s_ddq2, 'r');
grid on;
xlabel('t');
ylabel('ddQ(t)');
title('Accelerazione nello Spazio dei Giunti');
```



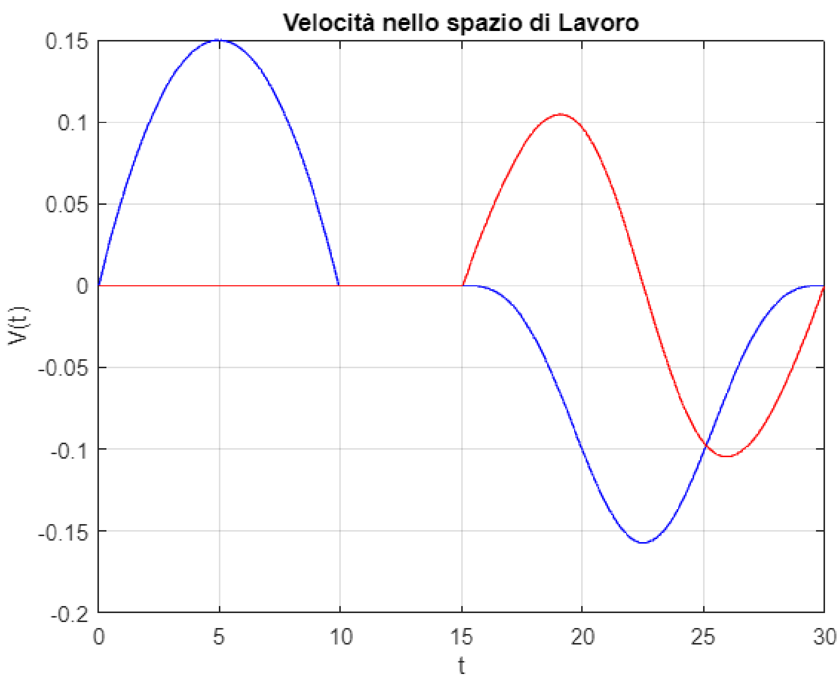
Vista d'insieme da T0 a T4

```
% Tempo totale
t_tot=linspace(T1, T4, 300);

x_tot = zeros(size(t_tot));
y_tot = zeros(size(t_tot));

vx_tot = zeros(size(t_tot));
vy_tot = zeros(size(t_tot));

ax_tot = zeros(size(t_tot));
ay_tot = zeros(size(t_tot));

q1_tot = zeros(size(t_tot));
q2_tot = zeros(size(t_tot));

dq1_tot = zeros(size(t_tot));
dq2_tot = zeros(size(t_tot));

ddq1_tot= zeros(size(t_tot));
ddq2_tot= zeros(size(t_tot));
```

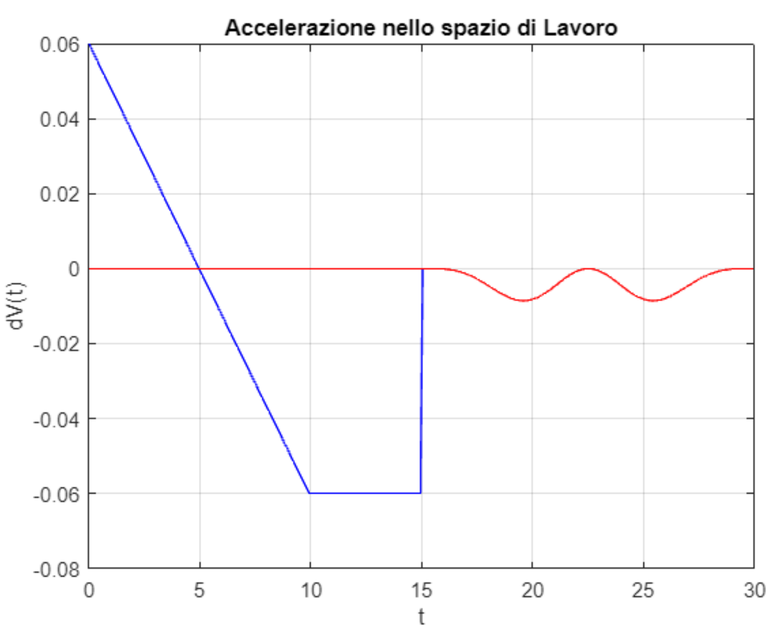
```
for i=1:100
    x_tot(i) = x(i);    y_tot(i) = y(i);
    vx_tot(i) = vx(i);    vy_tot(i) = vy(i);
    ax_tot(i) = ax(i);    ay_tot(i) = ay(i);
    q1_tot(i) = q1(i);    q2_tot(i) = q2(i);
    dq1_tot(i) = dq1(i);    dq2_tot(i) = dq2(i);
    ddq1_tot(i) = ddq1(i);    ddq2_tot(i) = ddq2(i);
end
for i=1:50
    x_tot(i+100) = x(100);    y_tot(i+100) = y(100);
    vx_tot(i+100) = vx(100);    vy_tot(i+100) = vy(100);
    ax_tot(i+100) = ax(100);    ay_tot(i+100) = ay(100);
    q1_tot(i+100) = q1(100);    q2_tot(i+100) = q2(100);
    dq1_tot(i+100) = dq1(100);    dq2_tot(i+100) = dq2(100);
    ddq1_tot(i+100) = ddq1(100);    ddq2_tot(i+100) = ddq2(100);
end
for i=1:150
    x_tot(i+150) = s_x(i);    y_tot(i+150) = s_y(i);
    vx_tot(i+150) = s_vx(i);    vy_tot(i+150) = s_vy(i);
    ax_tot(i+150) = s_ax(i);    ay_tot(i+150) = s_ay(i);
    q1_tot(i+150) = s_q1(i);    q2_tot(i+150) = s_q2(i);
    dq1_tot(i+150) = s_dq1(i);    dq2_tot(i+150) = s_dq2(i);
    ddq1_tot(i+150) = s_ddq1(i);    ddq2_tot(i+150) = s_ddq2(i);
end

figure; plot(t_tot,x_tot,'b',t_tot,y_tot,'r');
grid on;
xlabel('t');
ylabel('P(t)');
title('Posizione nello Spazio di Lavoro');
```



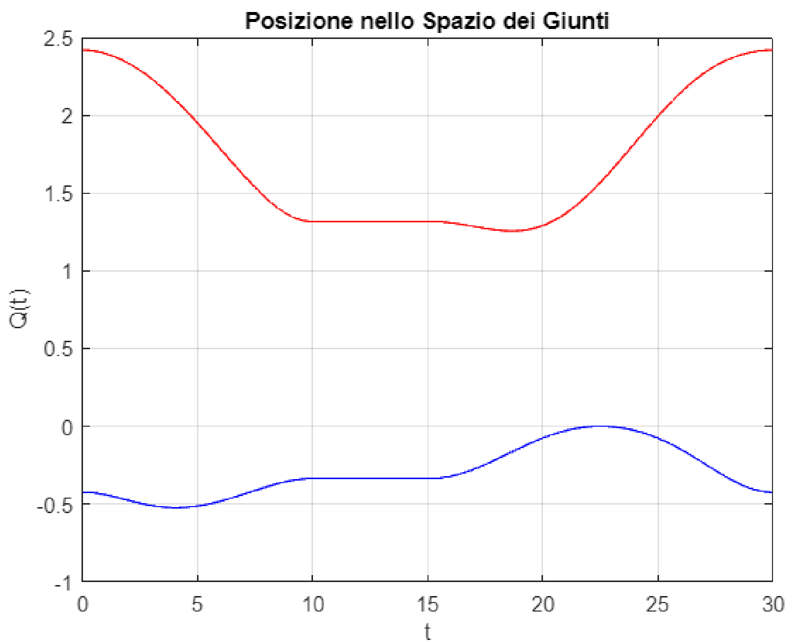
```
figure; plot(t_tot,vx_tot,'b',t_tot,vy_tot,'r');
grid on;
xlabel('t');
ylabel('V(t)');
title('Velocità nello spazio di Lavoro');
```



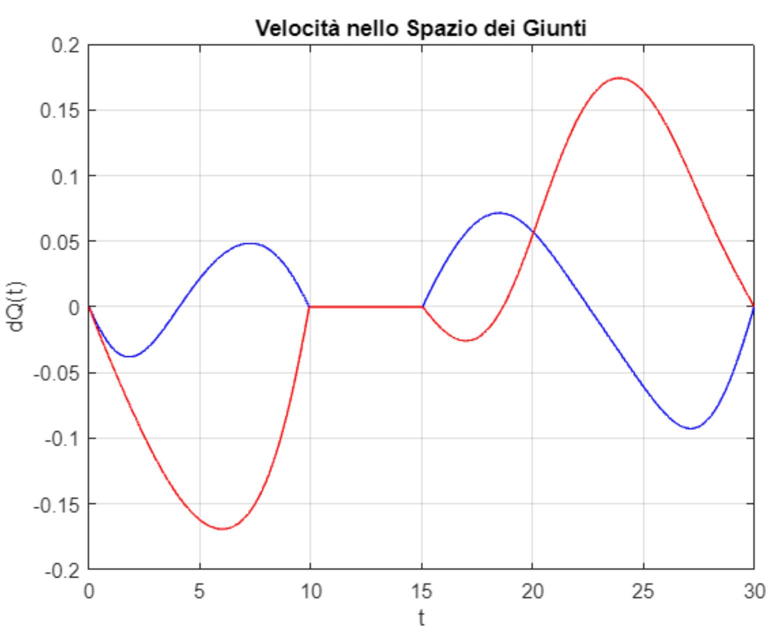

```
figure; plot(t_tot,ax_tot,'b',t_tot,ay_tot,'r');
grid on;
xlabel('t');
ylabel('dV(t)');
title('Accelerazione nello spazio di Lavoro');
```



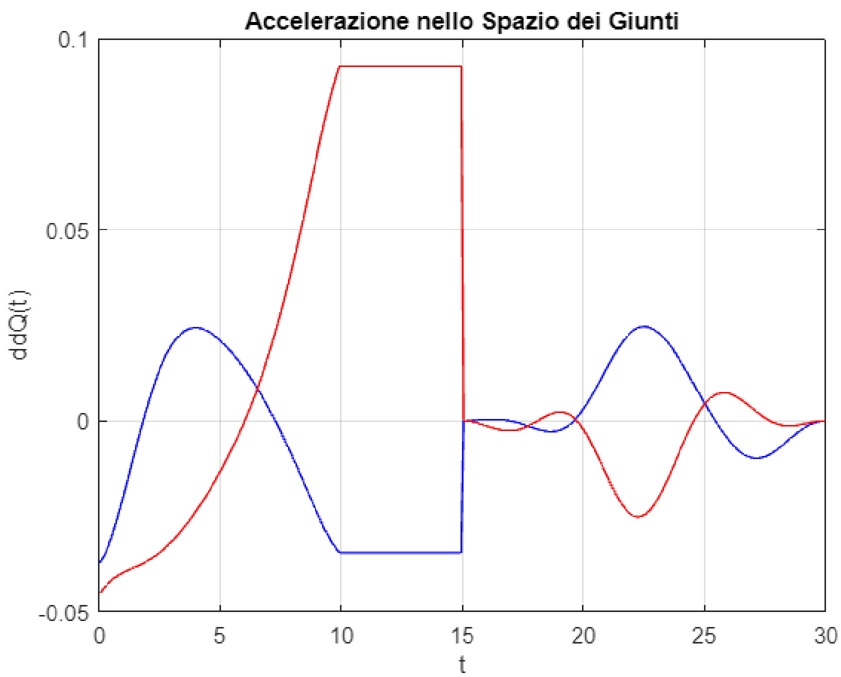
```
figure; plot(t_tot,q1_tot,'b',t_tot,q2_tot,'r');
grid on;
xlabel('t');
ylabel('Q(t)');
title('Posizione nello Spazio dei Giunti');
```




```
figure; plot(t_tot,dq1_tot,'b',t_tot,dq2_tot,'r');
grid on;
xlabel('t');
ylabel('dQ(t)');
title('Velocità nello Spazio dei Giunti');
```



```
figure; plot(t_tot,ddq1_tot,'b',t_tot,ddq2_tot,'r');
grid on;
xlabel('t');
ylabel('ddQ(t)');
title('Accelerazione nello Spazio dei Giunti');
```




```
P=[x_tot;y_tot];
V=[vx_tot;vy_tot];
Q=[q1_tot;q2_tot];
dQ=[dq1_tot,dq2_tot];
```

Dinamica

Approccio Lagrangiano

Considerando la geometria del robot risultano i seguenti vettori

$Q = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}$ vettore delle variabili di giunto.

$p_{c1} = \begin{bmatrix} l_1 c_1 \\ l_1 s_1 \end{bmatrix}$ $p_{c2} = \begin{bmatrix} a_1 c_1 + l_2 c_{12} \\ a_1 s_1 + l_2 s_{12} \end{bmatrix}$ baricentri dei due bracci

$v_{c1} = \begin{bmatrix} -\dot{q}_1 l_1 s_1 \\ \dot{q}_1 l_2 c_2 \end{bmatrix}$ $v_{c2} = \begin{bmatrix} -\dot{q}_1 a_1 s_1 - \left(\dot{q}_1 + \dot{q}_2\right) l_2 s_{12} \\ \dot{q}_1 a_1 c_1 + \left(\dot{q}_1 + \dot{q}_2\right) l_2 c_{12} \end{bmatrix}$ velocità rispetto ai baricentri

$g = \begin{bmatrix} 0 \\ -g \end{bmatrix}$ accelerazione di gravità

Dove l_1 ed l_2 sono le distanze dei due baricentri, ovvero $l_1 = \dfrac{a_1}{2}$ ec $l_2 = \dfrac{a_2}{2}$

Per calcolare l'energia cinetica bisogna introdurre i momenti di inerzia delle due aste rispetto ad assi passanti per i rispettivi baricentri

$I_1 = \dfrac{1}{12} m_1 a_1^2$ ed $I_2 = \dfrac{1}{12} m_2 a_2^2$

Da questi dati è possibile ricavare l'energia cinetica e l'energia potenziale

$$\mathrm{Ec}_1 = \frac{1}{2} m_1 \left(p_{c1}\right)^T p_{c1} + \frac{1}{2} I_1 \omega^2 = \frac{1}{2} m_1 l_1^2 (\dot{q}_1)^2 + \frac{1}{2} I_1 (\dot{q}_1)^2$$

$$\mathrm{Ec}_2 = \frac{1}{2} m_2 \left(p_{c2}\right)^T p_{c2} + \frac{1}{2} I_2 \omega^2 = \frac{1}{2} m_2 \left[(\dot{q}_1)^2 a_1^2 + \left(\dot{q}_1 + \dot{q}_2\right)^2 l_2^2 + 2\dot{q}_1 \left(\dot{q}_1 + \dot{q}_2\right) a_1 l_2 c_2 \right] + \frac{1}{2} I_{21} \left(\dot{q}_1 + \dot{q}_2\right)^2$$

$$\mathrm{Ec} = \mathrm{Ec}_1 + \mathrm{Ec}_2 = \frac{1}{2} m_1 l_1^2 (\dot{q}_1)^2 + \frac{1}{2} I_1 (\dot{q}_1)^2 + \frac{1}{2} m_2 \left[(\dot{q}_1)^2 a_1^2 + \left(\dot{q}_1 + \dot{q}_2\right)^2 l_2^2 + 2\dot{q}_1 \left(\dot{q}_1 + \dot{q}_2\right) a_1 l_2 c_2 \right] + \frac{1}{2} I_{21} \left(\dot{q}_1 + \dot{q}_2\right)^2$$

Per quanto riguarda invece l'energia potenziale gravitazionale

$$\mathrm{Ep} = - m_1 g^T p_{c1} - m_2 g^T p_{c2} = m_1\, g\, l_1\, s_1 + m_2\, g\, (a_1 s_1 + l_2 s_{12})$$

Si può dunque definire la funzione Lagrangiana

$$L = \mathrm{Ec} - \mathrm{Ep}$$

Da cui è possibile ricavare la coppia

$$\frac{d}{\mathrm{dt}}\frac{\partial L}{\partial \dot{Q}} - \frac{\partial L}{\partial Q} = T$$

Prima equazione

$$T_1 = \frac{d}{\mathrm{dt}}\frac{\partial L}{\partial \dot{q_1}} - \frac{\partial L}{\partial q_1} =$$
$$= \left(m_1 l_1^2 + I_1 + m_2 a_1^2 + m_2 l_2^2 + 2 m_2 a_1 l_2 c_2 + I_2\right)\ddot{q_1} + \left(m_2 l_2^2 + m_2 a_1 l_2 c_2 + I_2\right)\ddot{q_2} +$$
$$- 2 m_2 a_1 l_2 s_2\, \dot{q}_1\, \dot{q}_2 - m_2 a_1 l_2 s_2\, \dot{q}_2{}^2 + m_1 l_1\, g\, c_1 + m_2\, g\, a_1 c_1 + m_2\, g\, l_2\, c_{12}$$

Seconda equazione

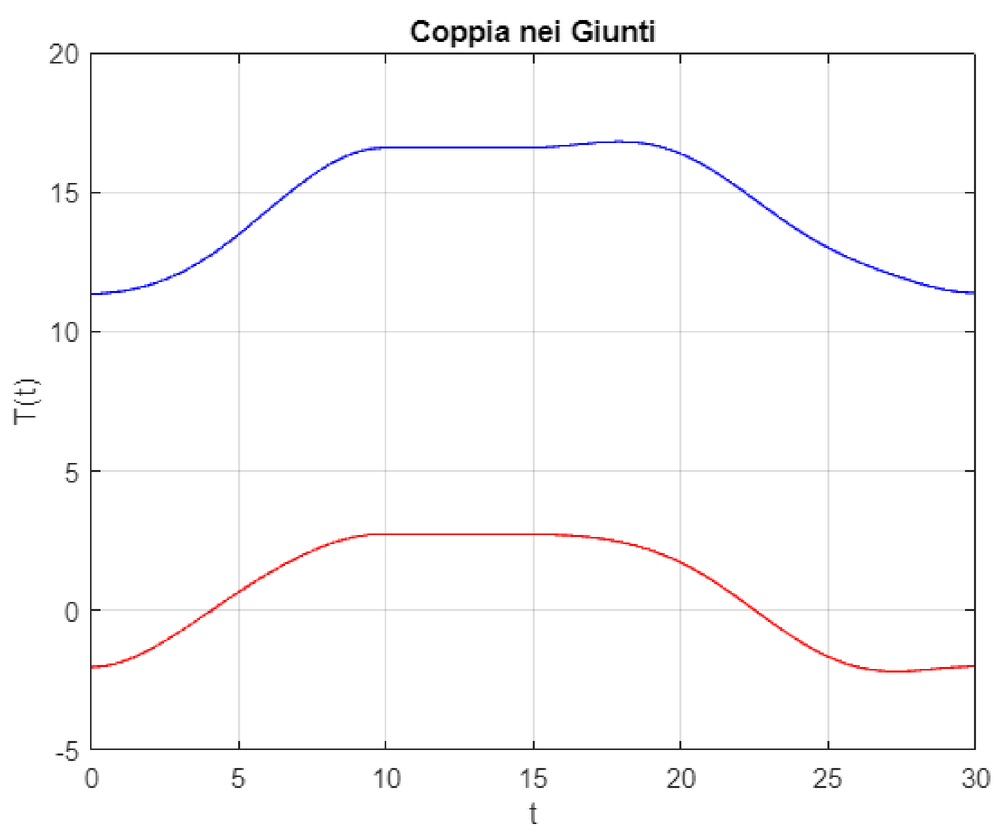
$$T_2 = \frac{d}{\mathrm{dt}}\frac{\partial L}{\partial \dot{q_2}} - \frac{\partial L}{\partial q_2} =$$
$$= \left(m_2 l_1^2 + m_2 a_1 l_2 c_2 + I_2\right)\ddot{q_1} + \left(m_2 l_2^2 + I_2\right)\ddot{q_2} +$$
$$+ m_2 a_1 l_2 s_2\, \dot{q}_1{}^2 + m_2\, g\, l_2\, c_{12}$$

Che in forma vettoriale si scrivono

$$\begin{bmatrix} m_1 l_1^2 + I_1 + m_2 a_1^2 + m_2 l_2^2 + 2 m_2 a_1 l_2 c_2 + I_2 & m_2 l_2^2 + m_2 a_1 l_2 c_2 + I_2 \\ m_2 l_1^2 + m_2 a_1 l_2 c_2 + I_2 & m_2 l_2^2 + I_2 \end{bmatrix}\begin{bmatrix}\ddot{q_1} \\ \ddot{q_2}\end{bmatrix} + \begin{bmatrix} -2 m_2 a_1 l_2 s_2\, \dot{q}_2 & -m_2 a_1 l_2 s_2\, \dot{q}_2 \\ m_2 a_1 l_2 s_2\, \dot{q}_1 & 0 \end{bmatrix}\begin{bmatrix}\dot{q}_1 \\ \dot{q}_2\end{bmatrix} + \begin{bmatrix}(m_1 l_1 + m_2 a_1) g\, c_1 + m_2\, g\, l_2\, c_{12} \\ m_2\, g\, l_2\, c_{12}\end{bmatrix} = \begin{bmatrix}T_1 \\ T_2\end{bmatrix}$$

```
% Massa dei bracci
m1=1; m2=1;

% Lunghezza dei bracci
a1=1; a2=1;

% Distanza baricentro
l1=a1/2; l2=a2/2;

% Accelerazione di gravità
g=9.81;

%Momenti di Inerzia
I1=(1/12)*m1*a1^2;
I2=(1/12)*m2*a2^2;

% Coppie

T1= zeros(size(t_tot));
T2= zeros(size(t_tot));
```

```
 for i = 1:numel(t_tot)

     % Matrice di Inerzia
     M=[(m1*l1^2 +I1+m2*a1^2+m2*l2^2 +2*m2*a1*l2*cos(q2_tot(i))+I2), (m2*l2^2 +
m2*a1*l2*cos(q2_tot(i))+I2);
           (m2*l1^2 +m2*a1*l2*cos(q2_tot(i))+I2), (m2*l2^2 +I2)];
     % Matrice delle Forze Apparenti
     H=[-2*m2*a1*l2*sin(q2_tot(i))*dq2_tot(i), -m2*a1*l2*sin(q2_tot(i))*dq2_tot(i);
           m2*a1*l2*sin(q2_tot(i))*dq1_tot(i), 0];

     % Termini Gravitazionali
     G=[(m1*l1+m2*a1)*g*cos(q1_tot(i))+m2*g*l2*cos(q1_tot(i)+q2_tot(i));
m2*g*l2*cos(q1_tot(i)+q2_tot(i))];

     % Calcolo delle Coppie
     T=M*[ddq1_tot(i);ddq2_tot(i)]+H*[dq1_tot(i);dq2_tot(i)]+G;

     T1(i) = T(1, :);
     T2(i) = T(2, :);
 end

 figure; plot(t_tot,T1,'b',t_tot,T2,'r');
 grid on;
 xlabel('t');
 ylabel('T(t)');
 title('Coppia nei Giunti');
```



L'equazione del modello dinamico si può dunque scrivere come

$$M(Q)\ddot{Q} + H\left(Q, \dot{Q}\right)\dot{Q} + G(Q) = T$$

che prendendo in considerazione l'attrito nei giunti diventa

$$M(Q)\ddot{Q} + H\left(Q, \dot{Q}\right)\dot{Q} + B\dot{Q} + G(Q) = T$$

In un modello a ciclo aperto la coppia generata attraverso queste equazioni viene poi adoperata nel modello reale del robot per generare i valori reali delle variabili di giunto.

In un modello in retroazione la coppia ai giunti viene invece generata attraverso l'equazione

$$T = K_p * e_p + K_v * e_v$$

dove e_p e e_v sono gli errori rispettivamente di posizione e velocità dei giunti, ovvero $(Q_r - Q)$ e $\left(\dot{Q}_r - \dot{Q}\right)$

mentre K_p e K_v sono delle matrici che rappresentano i guadagni, entrambe definite positive e diagonali.

Sostituendo l'equazione della coppia in retroazione con il modello reale del robot (che coincide per semplificazione con il modello dinamico ideale) si ottiene

$$M(Q)\ddot{Q} + H\left(Q, \dot{Q}\right)\dot{Q} + B\dot{Q} + G(Q) = K_p(Q_r - Q) + K_v\left(\dot{Q}_r - \dot{Q}\right)$$

$$M(Q)\ddot{Q} + \left[H\left(Q, \dot{Q}\right) + B + K_v\right]\dot{Q} + G(Q) + K_pQ = K_pQ_r + K_v\dot{Q}_r$$

Dinamica inversa

Avendo a disposizione il modello del robot e quindi il suo comportamento è possibile determinare l'accelerazione dei giunti poiché la matrice di inerzia $M(Q)$ è sempre invertibile in quanto definita positiva

Si può perciò scrivere

$$\ddot{Q} = -M^{-1}\left[H\left(Q, \dot{Q}\right) + B\right]\dot{Q} - M^{-1}G(Q) + M^{-1}T$$

Che dopo integrazione permette di ottenere i valori di Velocità e Posizione dei giunti $\dot{Q}\ e\ Q$, necessari per calcolare l'errore nel modello a ciclo chiuso

Leggi di controllo più precise

Ipotizzando leggi di controllo più precise è possibile ridurre gli effetti di interferenza delle forze apparenti, dell'attrito o della gravità

Ad esempio, ipotizzando una legge di controllo

$$T_1 = K_p(Q_r - Q) + K_v(\dot{Q}_r - \dot{Q}) + G_n(Q)$$

dove $G_n(Q)$ è un termine gravità nominale, sostituendo l'equazione della coppia nell'equazione del modello dinamico è possibile ridurre l'effetto della gravità, in quanto rimane solo l'effetto di un ΔG.

Con una legge del tipo

$$T_2 = T_1 + H_n(Q, \dot{Q})\dot{Q}$$

è invece possibile ridurre l'effetto delle forze apparenti.

(Questa è la legge di controllo usata nel modello semplificato)

Sfruttando questo metodo ricorsivamente, ove possibile in quanto la stima di tali parametri è molto complessa, è possibile arrivare anche ad annullare l'effetto della matrice di inerzia, approssimando così il modello ad un sistema lineare

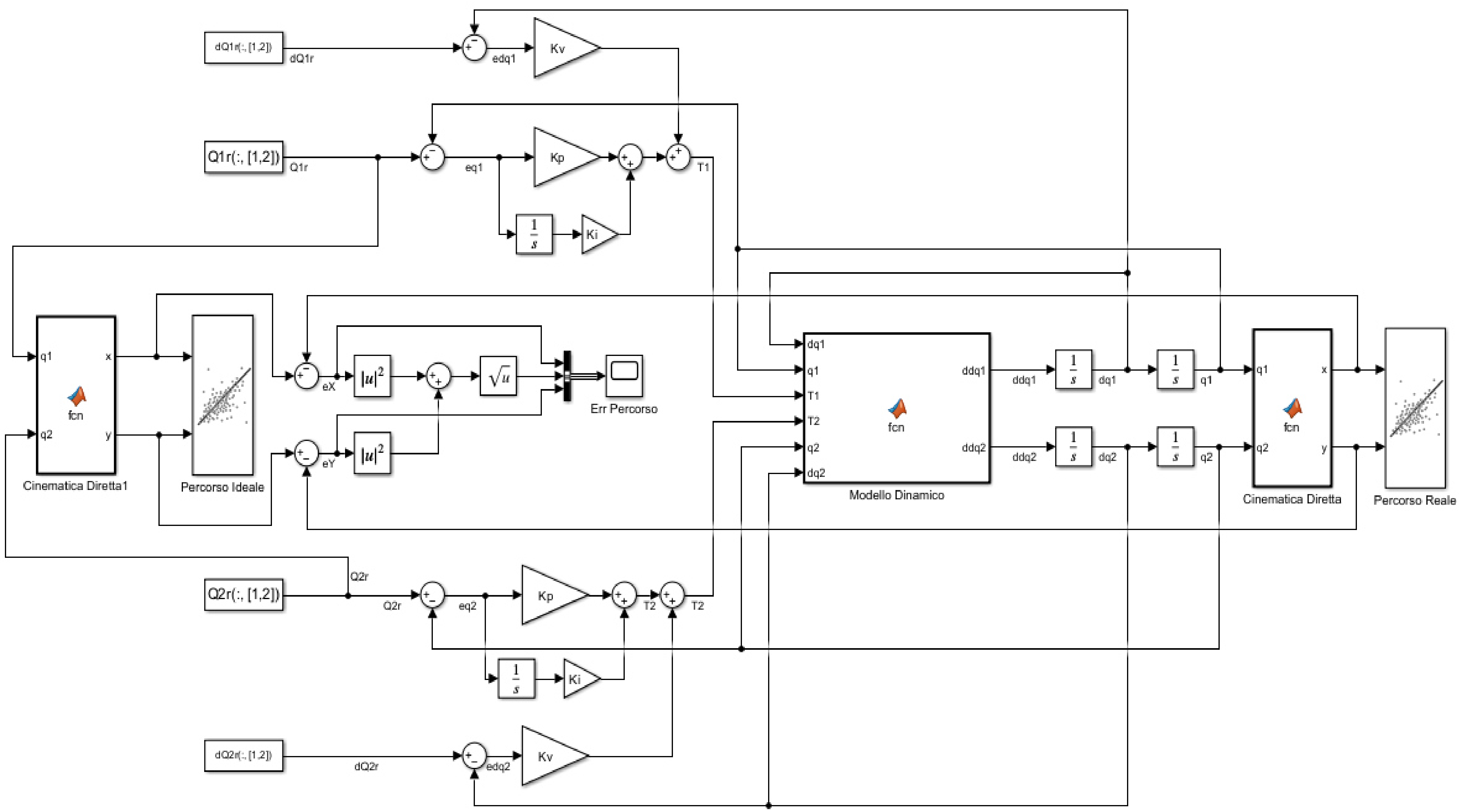
$$\ddot{q}_i(t) = \mathrm{kv}_i\, \dot{q}_i + \mathrm{kp}_i\, q_i = \mathrm{kv}_i\, \dot{\mathrm{qr}}_i + \mathrm{kp}_i\, \mathrm{qr}_i$$

a cui è possibile applicare la trasformata di Laplace, ottenendo così una FdT del tipo

$$Q_i(s) = \frac{\mathrm{kp}_i + \mathrm{kv}_i\, s}{s^2 + \mathrm{kv}_i\, s + \mathrm{kp}_i}\mathrm{qr}_i$$

Schema Simulink

```
Q1r=transpose([t_tot; q1_tot]);
Q2r=transpose([t_tot; q2_tot]);
dQ1r=transpose([t_tot;dq1_tot]);
dQ2r=transpose([t_tot;dq2_tot]);

Kp=300;
Kv=80;
Ki=2;

%open_system("DinamicaElaborato_final.slx");
```


dQ1r(:, [1,2])
dQ1r
edq1
Kv
Q1r(:, [1,2])
Q1r
eq1
Kp
Ki
T1
Cinematica Diretta1
Percorso Ideale
eX
eY
Err Percorso
Modello Dinamico
dq1
q1
T1
T2
q2
dq2
ddq1
ddq2
Cinematica Diretta
Percorso Reale
Q2r(:, [1,2])
Q2r
eq2
T2
dQ2r(:, [1,2])
dQ2r
edq2

Questo schema Simulink rappresenta il sistema di controllo per il robot a due bracci.

Nel file di simulazione sono presenti due schemi:

- Il primo adotta una legge di controllo meno precisa (Modello Dinamico)
- Il secondo adotta una legge di controllo più precisa che elimina l'effetto delle forze apparenti e riduce l'effetto della gravità (Modello Dinamico Semplificato)

Descrizione dei blocchi principali e del loro ruolo nel sistema:

Percorso Ideale e Percorso Reale:

- **Percorso Ideale**: Rappresenta la traiettoria desiderata del robot.
- **Percorso Reale**: Mostra la traiettoria effettiva seguita dal robot.

Cinematica Diretta:

- Calcola le posizioni x e y dei giunti q_1 e q_2 del robot.

```
function [x, y]= fcn(q1,q2)
L1=1;
L2=1;

c1=cos(q1);
c12=cos(q1+q2);
s1=sin(q1);
s12=sin(q1+q2);

x=L1*c1+L2*c12;
y=L1*s1+L2*s12;
```

Controllo di errore (Err Percorso):

- Confronta le posizioni attuali x e y con quelle desiderate.
- Calcola l'errore di posizione totale.

Modulo Dinamico (M.D.Semplificato):

- Simula la dinamica del robot considerando le velocità dq_1 e dq_2, e le posizioni q_1 e q_2.
- Calcola le nuove accelerazioni ddq_1 e ddq_2 basate sulle coppie T_1 e T_2.

DinamicaElaborato_final ▸ Modello Dinamico

```
function [ddq1,ddq2]= fcn(dq1,q1,T1,T2,q2,dq2)
g=9.81;
Q=[q1;q2];
dQ=[dq1;dq2];
T=[T1;T2];

a1=1;
a2=1;
m1=1;
m2=1;
l1=a1/2;
l2=a2/2;
I1=m1*a1^2/12;
I2=m2*a2^2/12;

c1=cos(q1);
c2=cos(q2);
s2=sin(q2);

% Matrice di Inerzia
    M=[(m1*l1^2 +I1+m2*a1^2+m2*l2^2 +2*m2*a1*l2*c2+I2), (m2*l2^2 + m2*a1*l2*c2+I2);
          (m2*l1^2 +m2*a1*l2*c2+I2), (m2*l2^2 +I2)];

% Matrice delle Forze Apparenti
    H=[-2*m2*a1*l2*s2*dq2, -m2*a1*l2*s2*dq2;
          m2*a1*l2*s2*dq1, 0];

    % Termini Gravitazionali
    G=[(m1*l1+m2*a1)*g*c1+m2*g*l2*cos(q1+q2); m2*g*l2*cos(q1+q2)];


ddQ = inv(M)*T - inv(M)*H*dQ - inv(M)*G;


ddq1=ddQ(1);
ddq2=ddQ(2);
```

DinamicaElaborato_final ▸ Modello Dinamico Semplificato

```
function [ddq1,ddq2]= fcn(dq1,q1,T1,T2,q2,dq2)
g=1.2;
Q=[q1;q2];
dQ=[dq1;dq2];
T=[T1;T2];

a1=1;
a2=1;
m1=1;
m2=1;
l1=a1/2;
l2=a2/2;
I1=m1*a1^2/12;
I2=m2*a2^2/12;

c1=cos(q1);
c2=cos(q2);
s2=sin(q2);

% Matrice di Inerzia
    M=[(m1*l1^2 +I1+m2*a1^2+m2*l2^2 +2*m2*a1*l2*c2+I2), (m2*l2^2 + m2*a1*l2*c2+I2);
          (m2*l1^2 +m2*a1*l2*c2+I2), (m2*l2^2 +I2)];

    % Termini Gravitazionali
    G=[(m1*l1+m2*a1)*g*c1+m2*g*l2*cos(q1+q2); m2*g*l2*cos(q1+q2)];


ddQ = inv(M)*T - inv(M)*G;


ddq1=ddQ(1);
ddq2=ddQ(2);
```

Controllori PID (Proporzionale-Integrale-Derivativo):

- Ogni asse ha un controllore PID per correggere l'errore di posizione.
- I blocchi con Kp, Ki, e Kv rappresentano rispettivamente i guadagni proporzionale, integrale e derivativo.

Ingresso di riferimento:

- Q_{1r} e Q_{2r} rappresentano i segnali di riferimento per le posizioni dei giunti.
- dQ_{1r} e dQ_{2r} rappresentano le velocità di riferimento.

Feedback di posizione e velocità:

- I segnali di velocità e posizione reali vengono retroazionati ai controllori PID per confrontarli con i valori di riferimento e correggere eventuali errori.

In sintesi, questo schema simula un sistema di controllo per un robot a due giunti, confrontando la posizione attuale con quella desiderata, calcolando l'errore, e applicando le correzioni necessarie tramite controllori PID per far seguire al robot il percorso ideale.

Report Simulazione

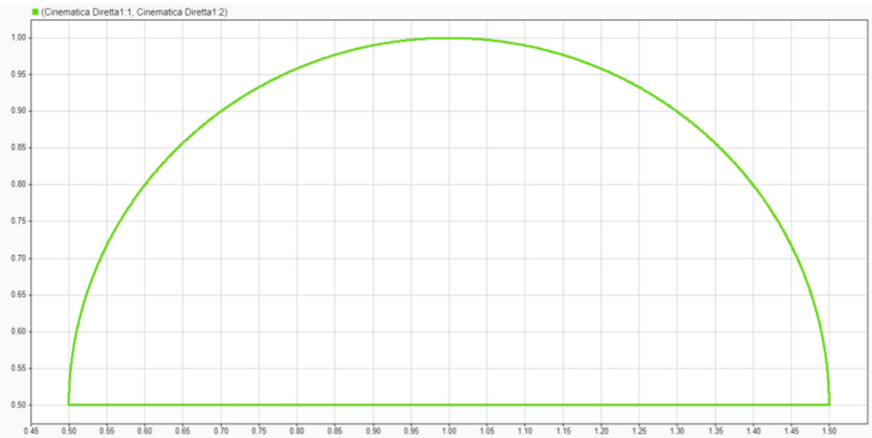
Attraverso questi parametri PID si ottengono i seguenti grafici.

Kp=300;

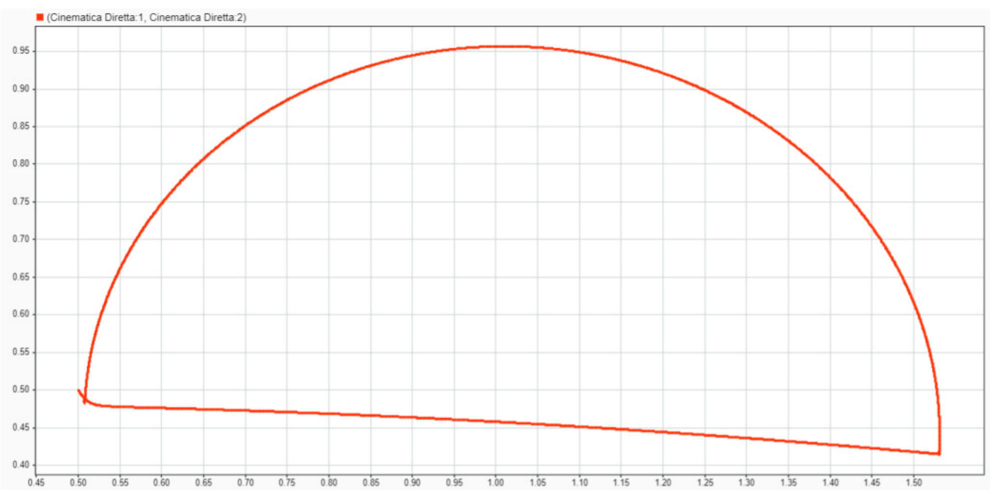
Kv=80;

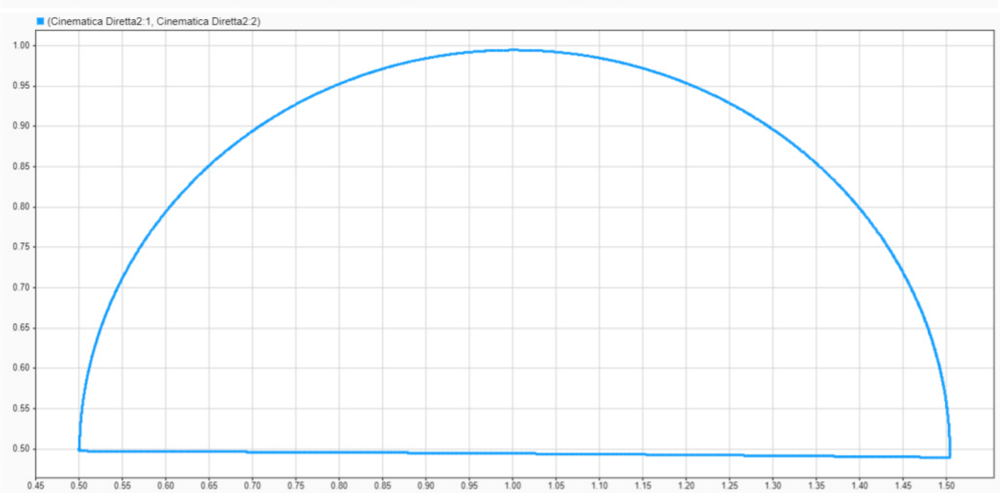
Ki=2;

Percorso Ideale

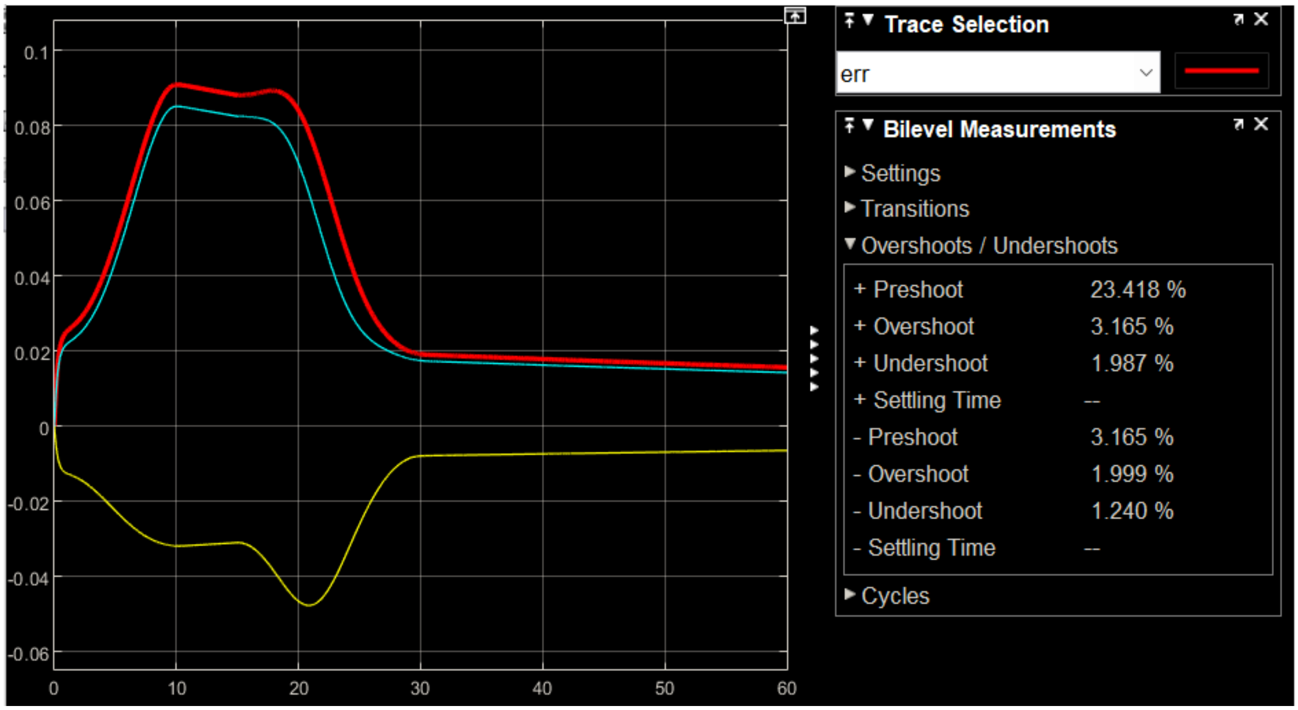


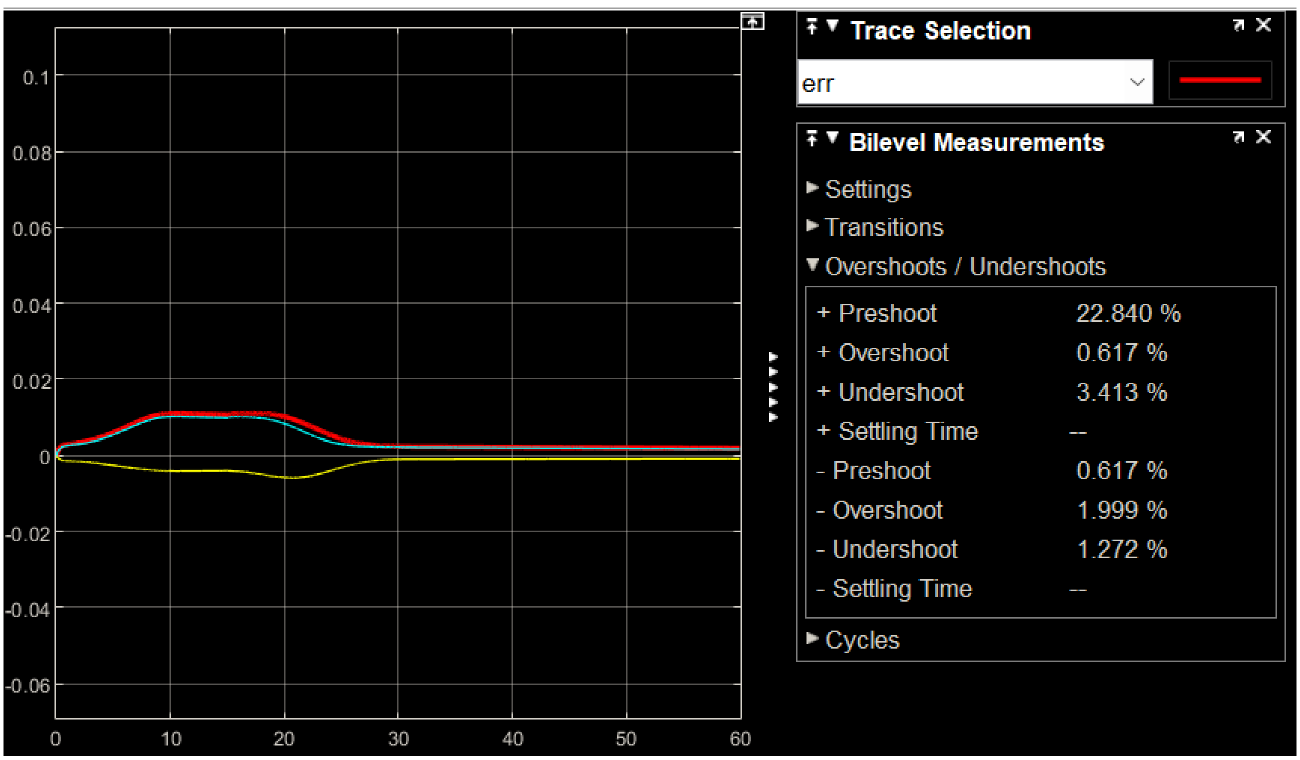
Percorso Reale Modello Dinamico (M.D.Semplificato)





Errore di Posizione





Come evidenziato dai grafici con questi parametri PID:

- sarebbe necessaria un'ulteriore taratura per il modello dinamico standard, in quanto l'effetto della gravità è molto più evidente, e ciò va a sfalsare di molto il percorso dell'attuatore rispetto al percorso ideale.
- sono accettabili per il modello dinamico semplificato, in quanto l'errore di posizione è molto meno significativo.

Varianti

Rototraslazione del Percorso

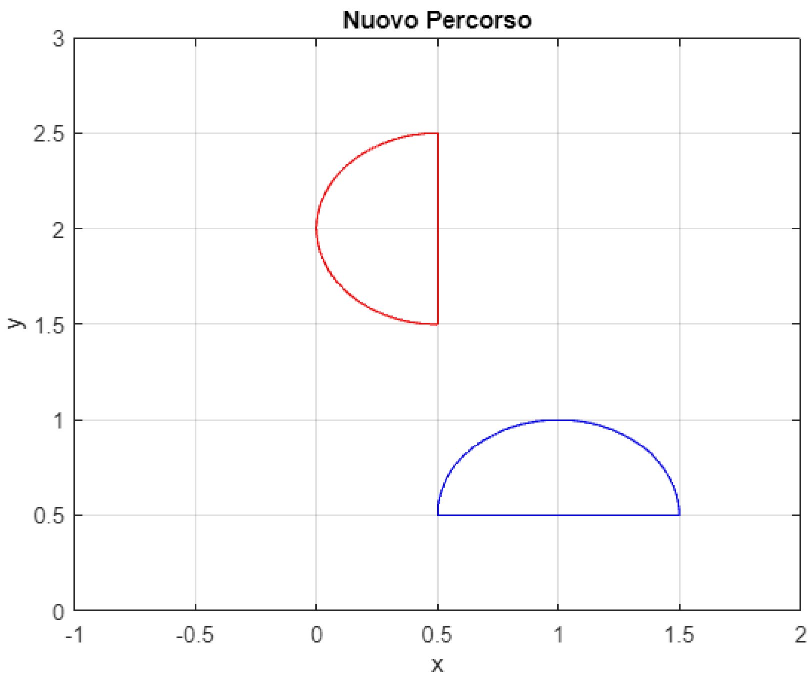
Per generare un nuovo percorso che sia uguale al precedente ma ruotato di $90°$ intorno all'asse z e traslato rispetto al vettore $\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$ è necessario utilizzare una matrice di rototraslazione così composta:

$$\begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & 1 \\ \sin(\theta) & \cos(\theta) & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Moltiplicando il vettore posizione normalizzato per la matrice si ottiene il vettore posizione del nuovo percorso.

$$\begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & 1 \\ \sin(\theta) & \cos(\theta) & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \\ 1 \end{bmatrix}$$

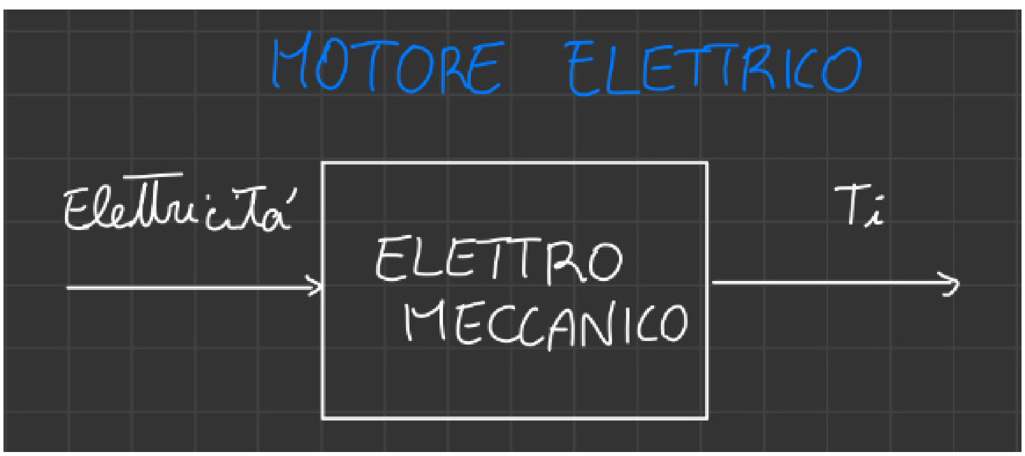
```
x_NewP = zeros(size(t_tot));
y_NewP = zeros(size(t_tot));

% Definire l'angolo di rotazione (in radianti)
Theta = pi/2;

% Calcolare la matrice di rototraslazione RT direttamente
RT = [cos(Theta), -sin(Theta), 0, 1;
      sin(Theta), cos(Theta), 0, 1;
      0, 0, 1, 0;
      0, 0, 0, 1];

for i=1:300
    POld=[x_tot(i);y_tot(i);0;1];
    PNew = RT * POld;
    x_NewP(i) = PNew(1);
    y_NewP(i) = PNew(2);
end
figure;
plot(x_NewP,y_NewP,'r');
hold on;
plot(x_tot,y_tot, 'b');
axis([-1 2 0 3]);
grid on;
xlabel('x');
ylabel('y');
title('Nuovo Percorso');
```

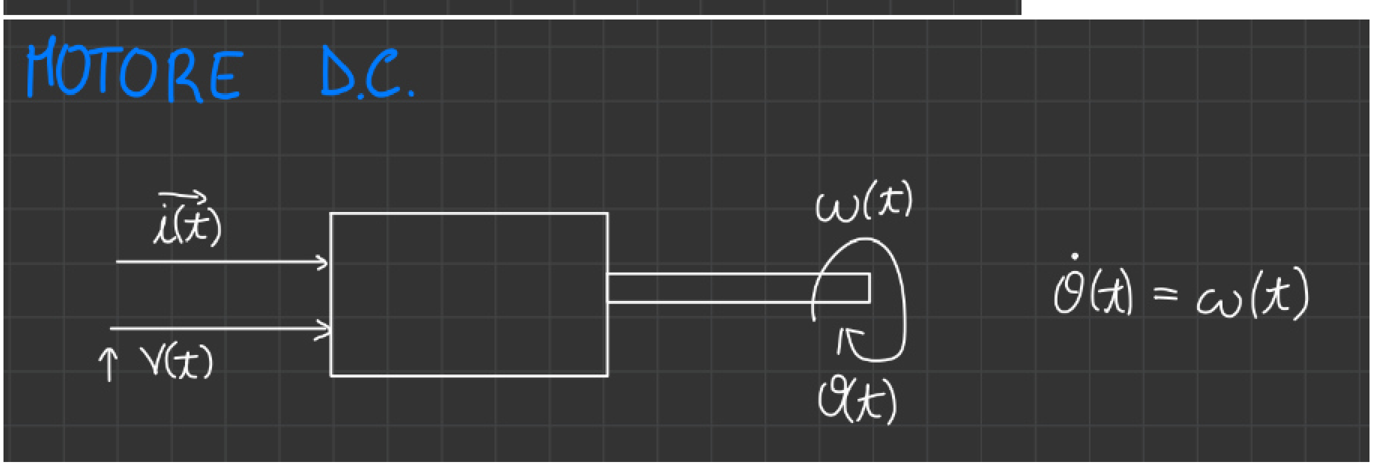


Aggiunta del Motore

Nella realtà la generazione delle coppie che agiscono sui giunti, rotoidali o prismatici, avviene attraverso dei Motori, generalmente elettrici D.C., ma che possono anche essere A.C. oppure motori idraulici o pneumatici.

In un motore elettrico l'elettricità in ingresso, corrente e differenza di potenziale, vengono convertiti in energia meccanica come movimento rotativo.

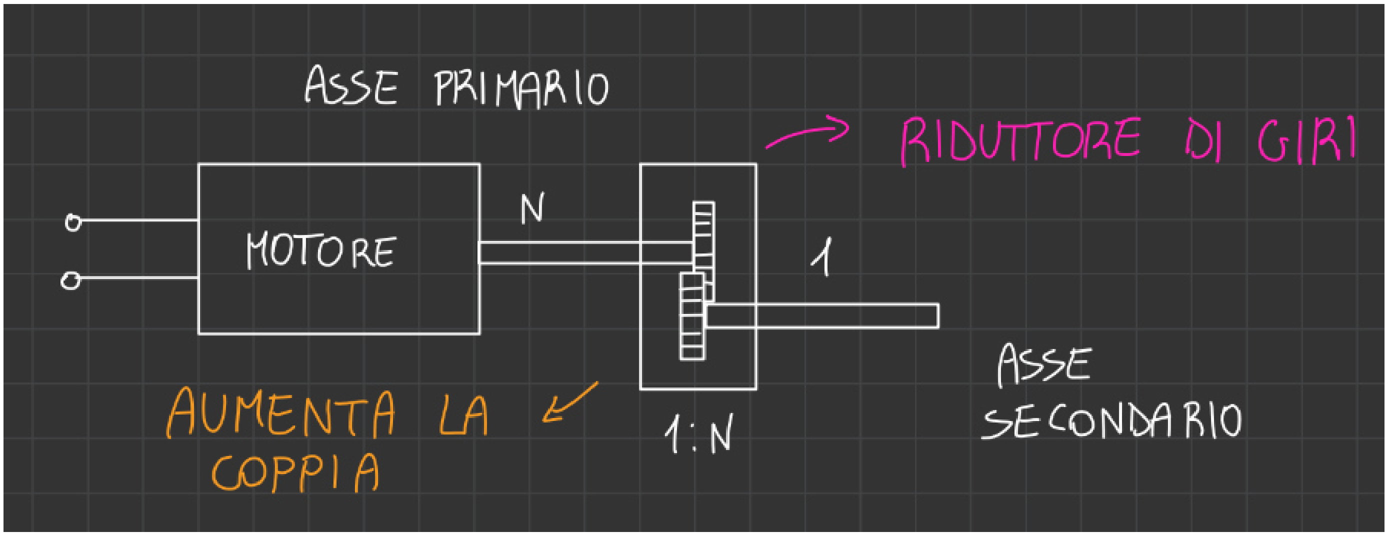




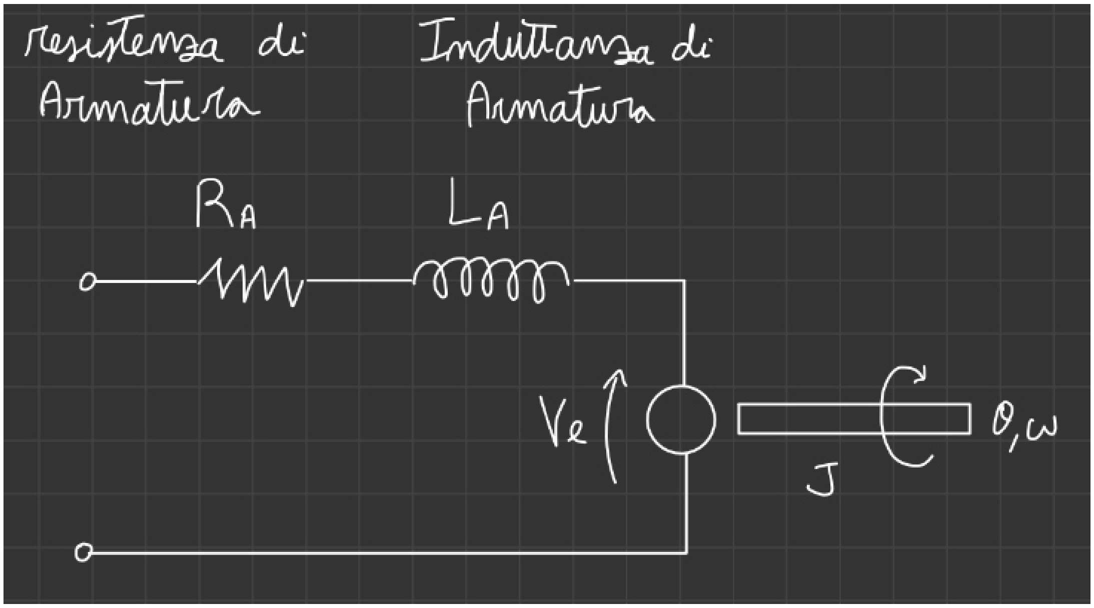
La coppia generata è proporzionale alla corrente $T_i \propto i_i$

La velocità di rotazione ω è "quasi proporzionale" alla differenza di potenziale V.

Al motore è spesso collegato uno o più riduttori di giri che permettono di aumentare la coppia finale erogata a parità di rotazione dell'asse primario. Conoscendo tale velocità di rotazione ed i rapporti di riduzione è possibile calcolare con precisione assoluta anche la velocità di rotazione e la coppia dell'asse finale.



Il modello elettrico del motore può essere rappresentato come segue:



Da cui è possibile ricavare le seguenti equazioni:

- $\text{Ve}(t) = K_e\,\omega(t)$
- $T(t) = K_T\,i(t)$

Con K_e e K_T caratteristici del motore.

Dinamica del Motore

La dinamica del motore è caratterizzata da due equazioni

$$\begin{cases} V(t) = R_A\, i_A(t) + L_A\, \dfrac{\mathrm{d}}{\mathrm{d}t} i_A(t) + \mathrm{Ve}(t) & \text{Eq Sistema Elettrico} \\ J\, \dot{\omega}(t) = -B\, \omega(t) + T(t) & \text{Eq Sistema Meccanico} \end{cases}$$

Da queste è possibile ricavare le equazioni di accoppiamento

$$\begin{cases} V(t) = R_A\, i_A(t) + L_A \dfrac{\mathrm{d}}{\mathrm{d}t} i_A(t) + K_e\, \omega(t) \\ J\, \dot{\omega}(t) = -B\, \omega(t) + K_T\, i(t) \end{cases}$$

Applicando la trasformata di Laplace alle prime

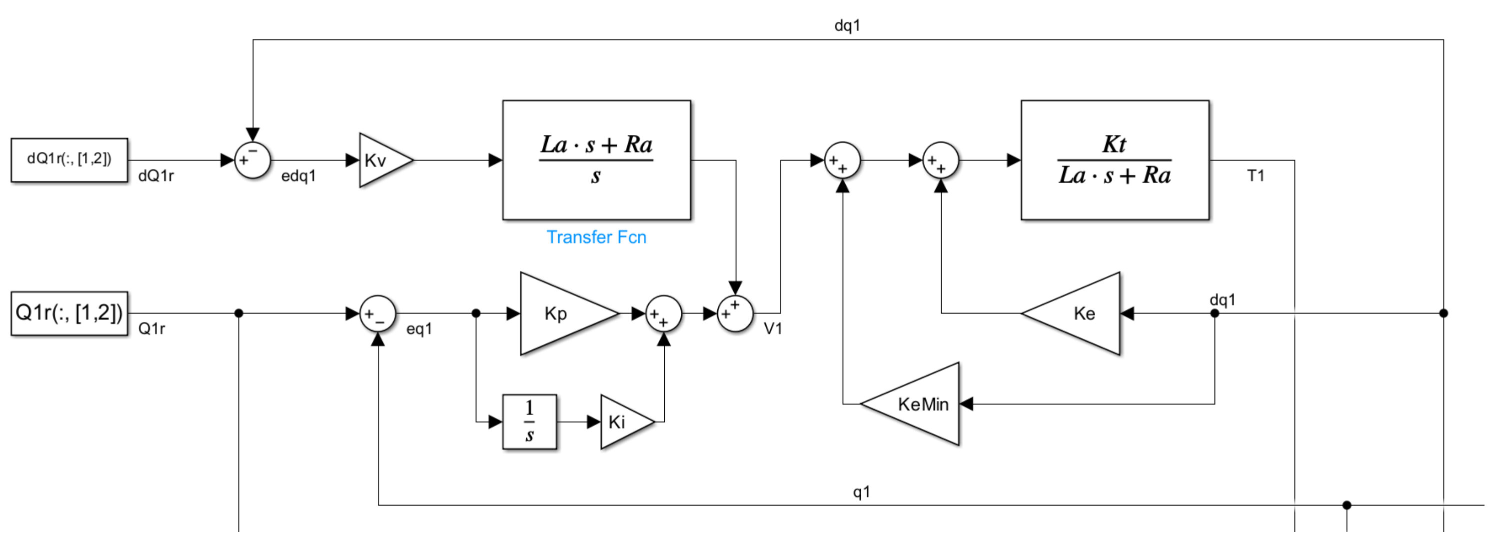
$$\begin{cases} V(s) = R_A\, I_A(s) + s\, L_A\, I_A(s) + \mathrm{Ve}(s) \\ s\, J\, \Omega(s) = -B\, \Omega(s) + T(s) \end{cases}$$

si ricavano facilmente le seguenti formule:

$$I_A(s) = [V(s) - \mathrm{Ve}(s)]\, \frac{1}{s\, L_A + R_A}$$

$$[s\, J + B]\Omega(s) = T(s)$$

Implementazione in Simulink



Il controllore PID stabilisce la tensione V_A che deve essere erogata al motore. Successivamente i guadagni K_T e K_e, insieme ai valori intrinseci del motore L_A e R_A, viene generata la coppia T. Aggiungere il guadagno $K_e \min$ serve ad eliminare l'effetto della retroazione intrinseca del motore dovuta alla legge di Lenz.

La coppia T generata sarà poi inserita in un modello dinamico che tiene conto sia dell'inerzia che dell'attrito intrinseci del motore, ed eventualmente anche del riduttore di giri.