

# Controllo Motore 2-DOF

## PREREQUISITI:

- Conoscenza dei thread e dei semafori in YARP. Si può indifferentemente far uso della classe `yarp::os::Thread` o `yarp::os::RateThread` ([documentazione sul web](#)).
- Le classi `Vector` e `Matrix` sono state riscritte ridefinendo gli operatori algebrici  $+$ ,  $-$ ,  $*$ ,  $...$ . Sono disponibili numerosi tutorial sull'overload degli operatori in C++: ad esempio [qui](#). Ne segue che operazioni che prima dovevano essere codificate come `M1=M2.add(M3)`; ora sono rese con un più immediato `M1=M2+M3`;

## DATI:

Sia data la catena cinematica seriale a due gradi di libertà di Figura 1, in cui il primo giunto è rotoidale con variabile di controllo  $\vartheta$  misurata in radianti, mentre il secondo giunto è di tipo prismatico con variabile di controllo  $r$  misurata in metri a partire dall'origine del sistema di riferimento (0,0). Da questa configurazione segue che il moto dell'end-effector  $pe$  appartiene al piano  $xy$  e le sue coordinate cartesiane risultano  $pe_x=r*\cos(\vartheta)$ ,  $pe_y=r*\sin(\vartheta)$ . Si noti come non esistano vincoli su  $r$ , ovvero  $r$  non è limitata superiormente, e che la coppia  $[\vartheta, r]$  individua la rappresentazione, a meno dell'ordine delle variabili indipendenti, di  $pe$  in coordinate polari.

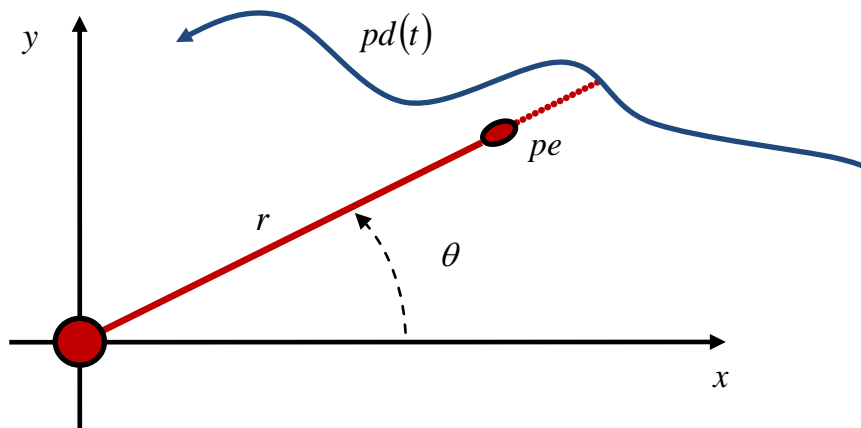


Figura 1

Si definiscano per il prosieguo le seguenti quantità:

$$q = \begin{bmatrix} \theta \\ r \end{bmatrix}, \quad \dot{q} = \begin{bmatrix} \dot{\theta} \\ \dot{r} \end{bmatrix}, \quad \Phi(q) = \begin{bmatrix} \Phi_x(q) \\ \Phi_y(q) \end{bmatrix} = \begin{bmatrix} pe_x \\ pe_y \end{bmatrix}, \quad J = \begin{bmatrix} \partial \Phi_x / \partial \theta & \partial \Phi_x / \partial r \\ \partial \Phi_y / \partial \theta & \partial \Phi_y / \partial r \end{bmatrix}$$

Si riconosce in  $\Phi(q)$  la mappa della cinematica diretta e in  $J(q)$  lo Jacobiano analitico corrispondente.

Si consideri inoltre la traiettoria desiderata  $pd(t) = \begin{bmatrix} pd_x(t) \\ pd_y(t) \end{bmatrix}$  e l'errore cartesiano  $e = pd - pe$ .

### RICHIESTI:

In Figura 2 è rappresentato uno schema a blocchi per il controllo dell'attuazione di una struttura meccanica avente la cinematica descritta precedentemente. Il controllore ha la funzione di azionare i motori con comando in velocità in modo tale che l'end-effector  $pe$  insegua un punto  $pd$  nello spazio cartesiano con traiettoria nota.

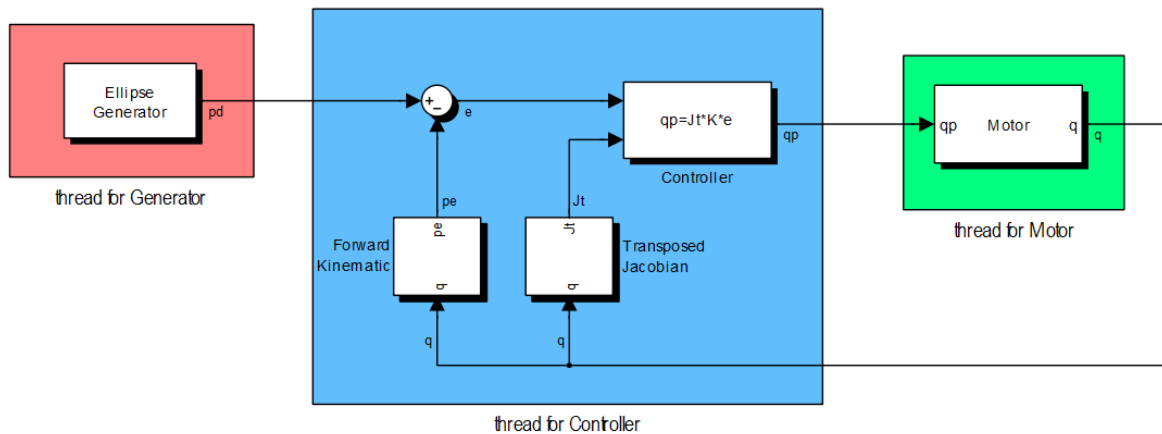


Figura 2

Fornita la struttura del progetto comprensiva della funzione *main()*, della classe di scambio dati *Resource* e di una classe *Generator* con la sua gerarchia polimorfica che produce la traiettoria cartesiana desiderata  $pd(t)$ , si richiede di implementare i restanti due thread YARP seguendo il paradigma del Producer/Consumer con risorse condivise:

1. Il thread **Motor** che simula l'azionamento dei giunti in modo indipendente con un gruppo motore che accetta in ingresso la quantità  $qp$  e ne restituisce in uscita l'integrale  $q$ .

Si suggerisce di implementare la funzione di integrazione a tempo discreto con la formula

di Tustin  $q_t = q_{t-1} + \frac{T_s}{2}(qp_t + qp_{t-1})$ , ove  $T_s$  è il passo di integrazione che coincide con il

periodo del thread. Il passo di campionamento  $T_s$  (dato dalla direttiva di define `SAMPLE_TIME`) sarà da specificare opportunamente in fase di progetto sapendo che la

frequenza di taglio meccanica è 5 Hz e che si stabilisce una frequenza massima di esecuzione del thread pari a 100 Hz.

2. Il thread **Controller** che esegue il ciclo di controllo con periodo  $T_s$ , allo scopo di ottenere l'inseguimento della traiettoria di  $pd$  da parte di  $pe$ , annullando a regime l'errore cartesiano  $e$ , comandando in velocità ( $qp$ ) il motore.

Si suggerisce di implementare la legge di controllo dello Jacobiano trasposto  $qp = J^T \cdot K \cdot e$ , con  $K$  opportuna matrice diagonale dei guadagni che regoli la velocità di convergenza dell'algoritmo.