**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](http://eris.liralab.it/yarpdoc/de/d8d/group__key__class.html)).
* 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](http://www.tutorialpc.it/c++12.asp). 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 ***θ*** 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 *pex=r\*cos(θ)*, *pey=r\*sin(θ)*. Si noti come non esistano vincoli su *r*, ovvero *r* non è limitata superiormente, e che la coppia [*θ*, *r*] individua la rappresentazione, a meno dell’ordine delle variabili indipendenti, di *pe* in coordinate polari.













Figura

Si definiscano per il prosieguo le seguenti quantità:

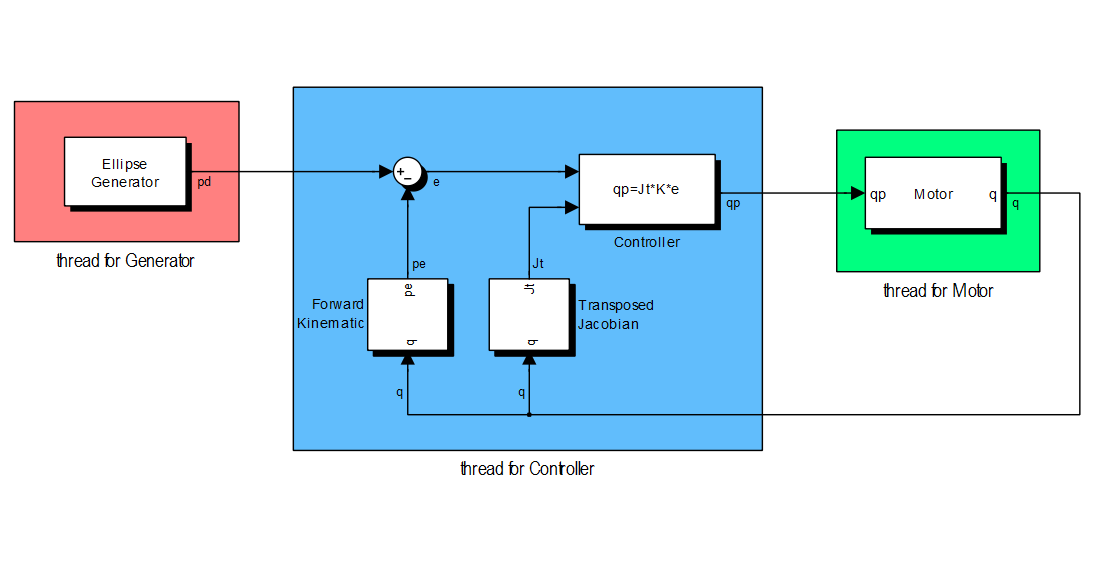


Si riconosce in  la mappa della cinematica diretta e in  lo Jacobiano analitico corrispondente.

Si consideri inoltre la traiettoria desiderata e l’errore cartesiano .

**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

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 , ove ***Ts*** è il passo di integrazione che coincide con il periodo del thread. Il passo di campionamento *Ts* (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.

1. Il thread **Controller** che esegue il ciclo di controllo con periodo *Ts*, 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 , con *K* opportuna matrice diagonale dei guadagni che regoli la velocità di convergenza dell’algoritmo.