Relazione Progetto Robotica

Giuseppe Montanaro 62719

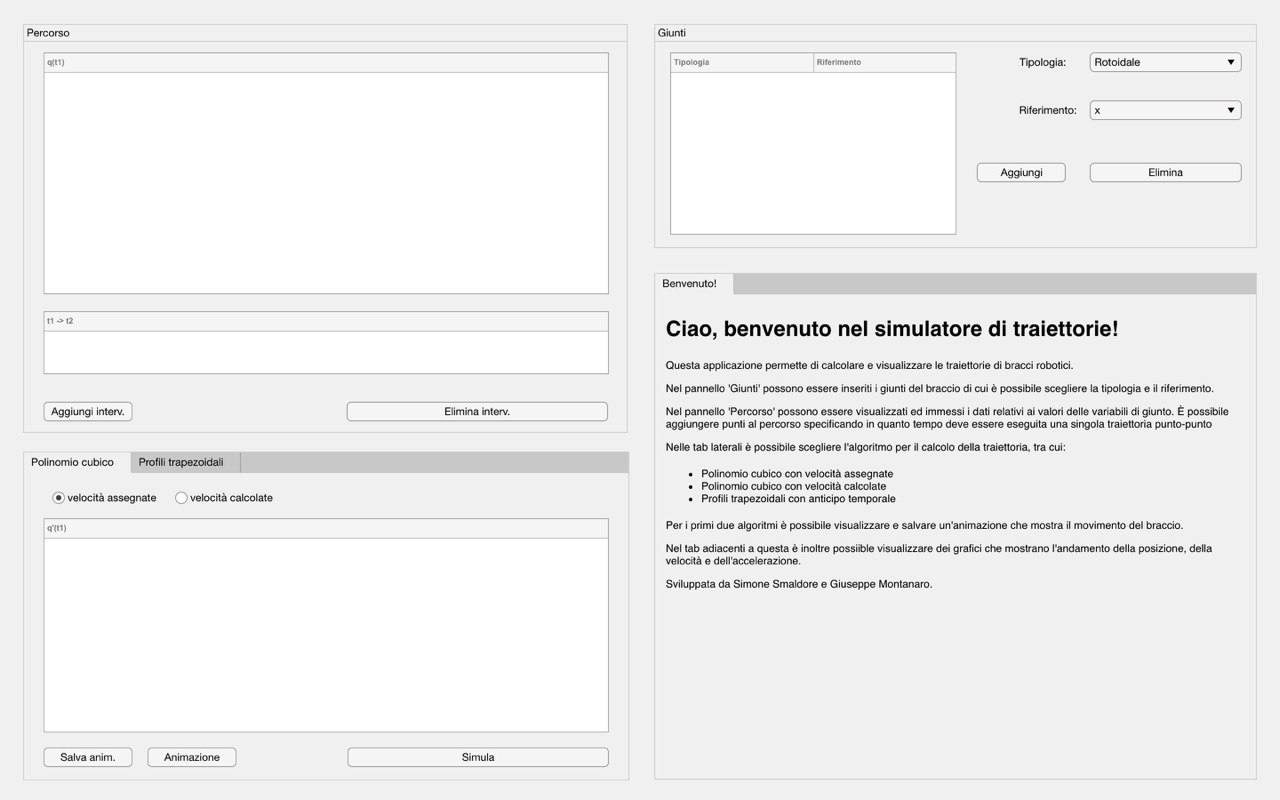
Simone Smaldore 62728

Per il nostro progetto abbiamo sviluppato due applicativi in ambiente MATLAB per la pianificazione di traiettorie su percorso assegnato nello spazio dei giunti. I tre algoritmi tra cui l’utente può scegliere sono:

* sequenza di polinomi cubici con velocità assegnate nei punti di percorso;
* sequenza di polinomi cubici con velocità calcolate nei punti di percorso;
* sequenza di profili trapezoidali con anticipo temporale (passaggio in prossimità dei punti di via ), con anticipo specificato dall’utente

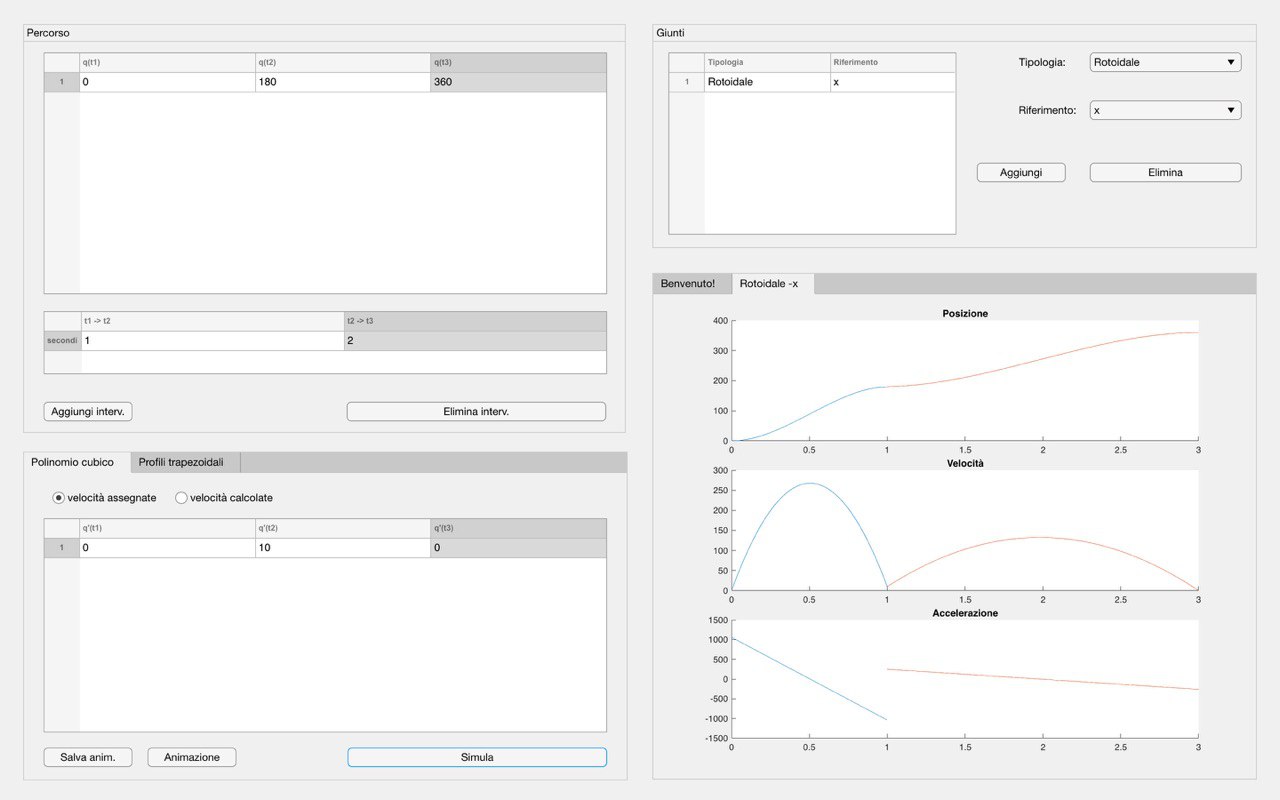
Abbiamo sviluppato una prima versione più semplice da linea di comando per la scrittura ed il testing dei veri e propri algoritmi, mentre in seguito abbiamo progettato un’applicazione con una GUI per l’immissione dei dati in maniera più intuitiva e la visualizzazione dei dati in modo più ordinato. Tutto il codice è scaricabile al seguente [link](https://drive.google.com/drive/folders/1pXfT7Zjib8STOCQ5gP_FJ37OC2pBakDT?usp=sharing). Per eseguire la versione da linea di comando basta eseguire Principale.m mentre per la versione con GUI va avviata l’applicazione main.mlapp. L’applicativo da linea di comando effettua le stesse operazioni della GUI solamente che per l’immissione dei dati utilizza delle apposite funzioni (immettiPercorso, leggiTempi, leggiVelocita, leggiParametriC) mentre per la GUI si utilizzano bottoni e tabelle. Inoltre le funzioni per stampare i vari grafici degli algoritmi cambiano tra le due applicazioni solo per adattarsi ai due ambienti differenti, ma il risultato in output è sempre lo stesso. Per questo motivo andremo ad analizzare solamente l’applicativo scritto con la GUI.

1. Schermata Principale

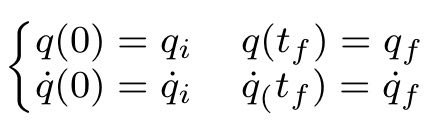


Nella schermata principale abbiamo in basso a destra una tab di benvenuto che ci spiega in linea di massima il funzionamento dell’applicazione (qui successivamente stamperemo anche i grafici). In alto a destra abbiamo una tabella che ci indica i giunti che vogliamo considerare per i nostri algoritmi: Abbiamo inoltre la possibilità di aggiungere giunti rotoidali o prismatici che ruotano o si distendono rispetto ad uno degli assi della terna base. In alto a sinistra possiamo specificare il nostro percorso nello spazio dei giunti andando a specificare il valore delle variabili di giunto nei vari istanti di tempo ( N.B: Per i giunti rotoidali vanno espresse tramite gradi e non con i radianti). Possiamo poi utilizzare la tabella sottostante per aggiungere ed eliminare i vari istanti di tempo che andremo a considerare tempo ( N.B: Come istante iniziale andremo a considerare sempre t=0). Per quanto riguarda la schermata in basso a sinistra, la andremo ad analizzare in maniera più dettagliata per ogni punto visto che cambierà da algoritmo ad algoritmo.

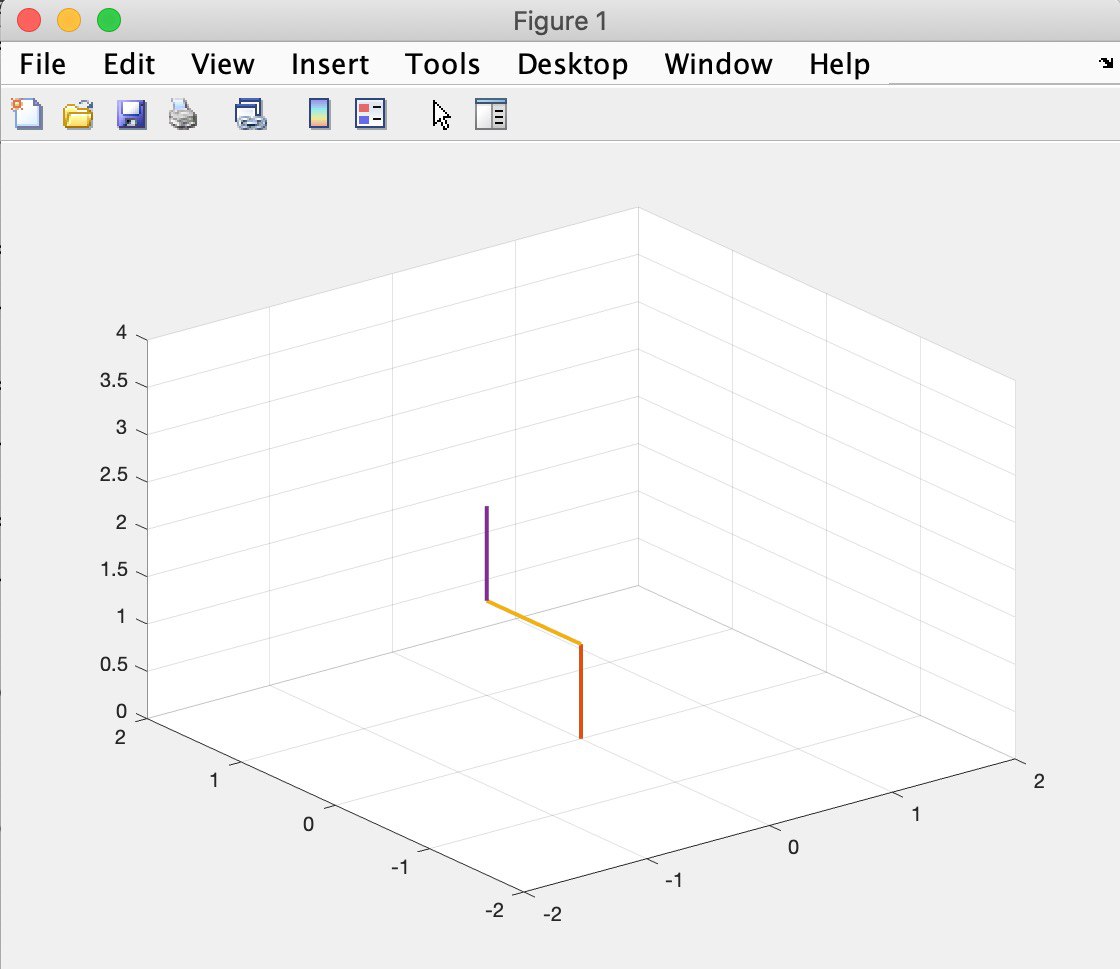
1. Sequenza di polinomi cubici con velocità assegnate nei punti di percorso;



Per eseguire il primo algoritmo andiamo a spuntare la casella velocità assegnate e andiamo a indicare le varie velocità nella tabella successiva. Premendo il pulsante simula a questo punto, verranno richiamate le funzioni generaPolinomiCubici e stampaGraficoGUI per l’esecuzione vera e propria dell’algoritmo e la stampa dei grafici di Posizione Velocità e Accelerazione tra i vari istanti di tempo per ogni giunto considerato. Per generare ciascuno dei polinomi interpolatori abbiamo considerato un polinomio di terzo grado q(t) con coefficienti incogniti a0 a1 a2 e a3 e i seguenti vincoli:

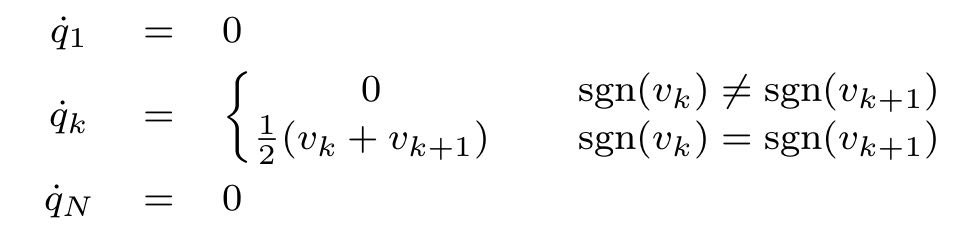


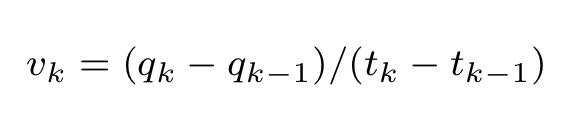
Inoltre andando a cliccare sul pulsante animazione verrà aperta una figura Matlab in 3D che simula la traiettoria rispetto ai valori indicati (N.B La traiettoria verrà simulata correttamente solo per i giunti rotoidali: i giunti prismatici infatti si muoveranno solamente rispetto direzioni della terna base x,y,z). Infine con il pulsante salva animazione si potrà esportare l’animazione in formato mp4.



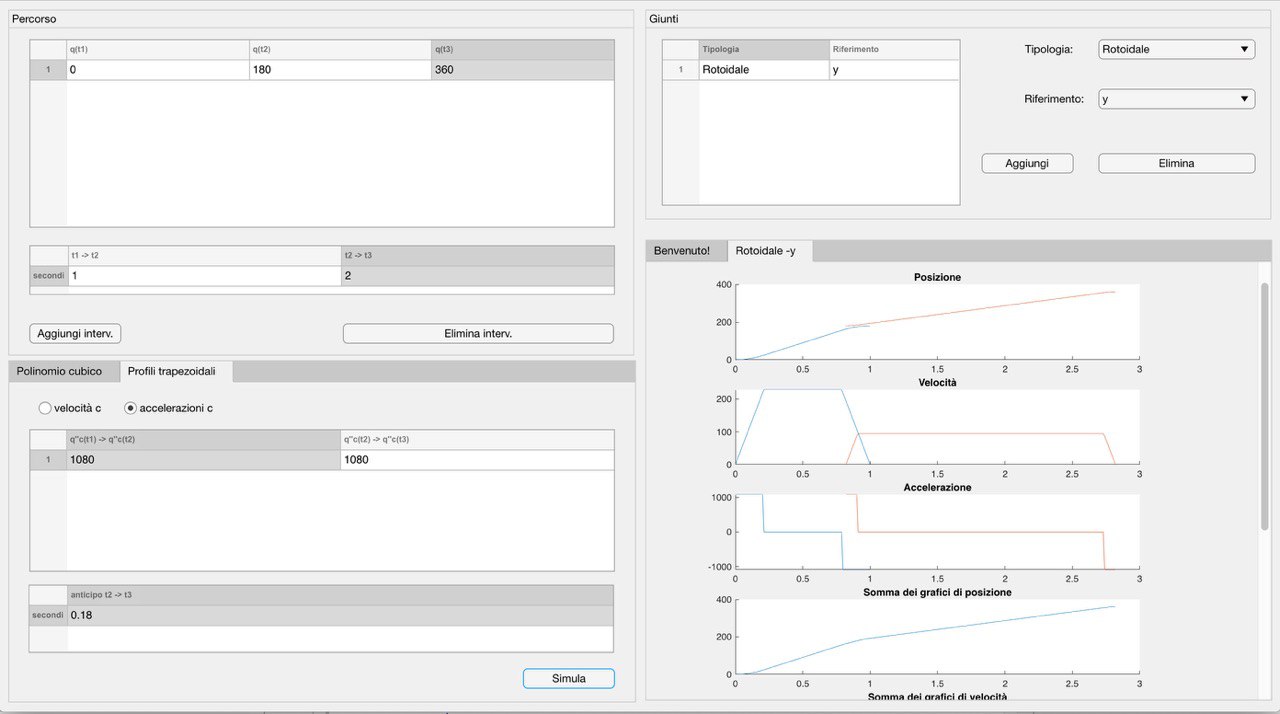
1. Sequenza di polinomi cubici con velocità calcolate nei punti di percorso;

Questo punto è molto simile al precedente: L’unica sostanziale differenza è che una volta spuntata la casella velocità calcolate non si dovrà popolare la tabella successiva visto che le velocità sono calcolate a partire dal percorso e dai tempi assegnati con la funzione calcolaVelocita. Per il resto il funzionamento rimane identico. Per implementare questo metodo abbiamo utilizzato le seguenti formule:

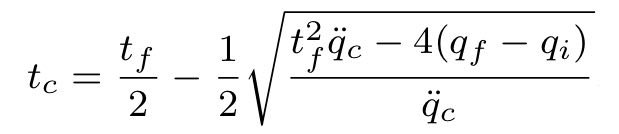


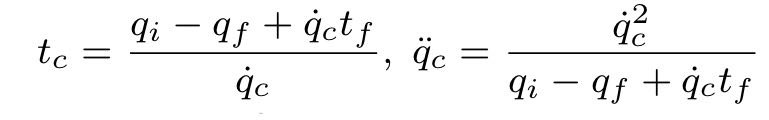


1. Sequenza di profili trapezoidali con anticipo temporale (passaggio in prossimità dei punti di via ), con anticipo specificato dall’utente

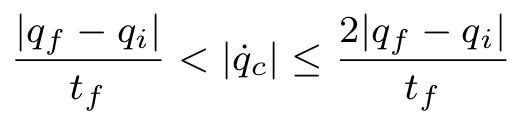
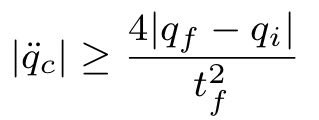


Per quanto riguarda la sequenza di profili trapezoidali i dati presi in input ovviamente cambiano. Si ha la possibilità di specificare le velocità q1c o le accelerazioni q2c. Inoltre poi è possibile specificare gli anticipi per i trapezi (N.B Gli anticipi vanno considerati tra il trapezio k-esimo ed il trapezio k-1-esimo poiché verranno cumulati in automatico dall’algoritmo). Cliccando il tasto simula verrà eseguito il metodo generaTrapezi per calcolare tutti tc, q1c e q2c con le seguenti formule:





Prima di effettuare la simulazione però, vengono effettuate alcune convalide per verificare di aver calcolato un tc consistente (cioè reale e minore o uguale a tf/2). Nel caso anche un solo valore non rispetti i vincoli successivi, l’algoritmo non viene eseguito mostrando un messaggio di errore.



Dopo aver superato la convalida viene eseguito il metodo stampaGraficotrapeziGUI per generare i grafici di posizione, velocità, accelerazione, somma di posizione, somma di velocità e somma di accelerazione per ognuno dei giunti considerati, implementando il metodo con le seguenti formule:

