UNIVERSITA' POLITECNICA DELLE MARCHE

FACOLTA' DI INGEGNERIA

Corso di Laurea magistrale in **Ingegneria Meccanica**

SVILUPPO DI UN MODELLO NUMERICO DI CALCOLO PER LA PROGETTAZIONE MECCANICA DI SISTEMI DI ATTUAZIONE DI GIUNTI ROBOTICI

DEVELOPMENT OF A NUMERICAL CALCULATION MODEL FOR THE MECHANICAL DESIGN OF ROBOTIC JOINT ACTUATION SYSTEMS

Relatore: Chiar.mo

Prof. **Massimo Callegari**

Tesi di Laurea di:

Dionigi Micolucci

A.A. **2022** / **2023**

SOMMARIO

ABSTRACT

Questa tesi è stata sviluppata grazie al lavoro di tirocinio svolto presso SMARTENGINEERING s.p.a., capofila del gruppo industriale SMARTINDUSTRY [1].

Il contesto in cui opera l'azienda, quello della consulenza, richiede di rispondere alle esigenze e alle richieste dei clienti in modo rapido e sistematico, senza tuttavia rinunciare alla qualità delle soluzioni proposte. Lavorando nel campo delle macchine automatiche, spesso bisogna studiare soluzioni personalizzate per la progettazione di cinematismi. In quest'ottica è stato pensato di realizzare un modello numerico per poter ottenere, in modo veloce e senza passare per la modellazione al CAD, i parametri cinematici e le coppie agenti sui giunti di catene cinematiche seriali. L'obiettivo è quello di avere uno strumento di supporto alle prime fasi della progettazione, in cui si fanno scelte preliminari sui layout o sugli azionamenti al fine di indirizzare al meglio le successive fasi di progettazione evitando perdite dal punto di vista temporale ed economico e dunque aumentando l'efficacia e l'efficienza delle scelte progettuali stesse.

Pertanto, in questo lavoro verrà svolto uno studio della cinematica e della dinamica delle catene cinematiche aperte per comprendere quali aspetti matematici e fisici trasformare in algoritmi. Terminato questo studio, verrà mostrato come è stato implementato il codice dal punto di vista degli input, del calcolo e infine degli output. Seguirà un capitolo dedicato alla verifica dei risultati ottenuti, e un esempio di come il modello possa risultare utile per gli scopi preposti.

1. INTRODUZIONE

Prima di addentrarci nel vivo della trattazione, è bene contestualizzare il progetto, per poter comprendere meglio il motivo di alcune scelte progettuali e la logica con cui è stato pensato il tool di calcolo.

Pertanto, in questo capitolo verrà fatta una descrizione dell'azienda, chiarendo il settore in cui opera, e verrà fatta una breve introduzione al dimensionamento degli attuatori elettrici, per meglio comprendere l'idea che ha mosso lo sviluppo del tool.

1.1. Contesto aziendale

SMARTENGINEERING s.p.a. è un'azienda di consulenza il cui *core business* è l'ingegneria industriale meccanica e meccatronica con alcune specificità di eccellenza focalizzate nell'automazione, packaging, macchine utensili, automotive e assembly machine.

Ad oggi conta circa 150 ingegneri, con 130 progetti in corso e più di 1000 conclusi per un totale di più di 280000 ore di progettazione totali.

L'azienda può contare su una rete di sedi presenti in tutta Italia, ognuna delle quali ha caratteristiche spiccate in alcuni dei campi menzionati, ma tutte accomunate dalla medesima capacità di trasferire know-how tecnologico in funzione degli obiettivi preposti in condivisione con i clienti.

SMARTENGINEERING fa parte del gruppo industriale SMARTINDUSTRY di cui è capofila.

La propensione dell'azienda alla continua innovazione la porta non solo a lavorare in progetti R&D, ma anche a collaborare con le università, attivare tirocini formativi o anche academy interne e corsi di formazione.

1.2. Dimensionamento degli attuatori elettrici

Il contesto di mercato in cui opera l'azienda la porta ad avere a che fare nella maggior parte dei casi con cinematismi spesso personalizzati sulla base delle specifiche richieste dal cliente. Uno degli aspetti più importanti in tal senso è cercare di fare delle stime iniziali circa il dimensionamento dei motori elettrici per poter indirizzare fin da subito la progettazione nella giusta direzione.

Per tale motivo, andiamo a vedere quelli che sono i parametri principali da prendere in considerazione per il dimensionamento degli attuatori, ciò sarà molto utile per decidere gli output che dovrà produrre il tool, i quali dovranno permettere anche di fare delle stime sul dimensionamento degli attuatori, in quanto aspetto che influenza notevolmente le scelte progettuali.

I motori elettrici, siano essi AC, DC, Brushless ecc. sono dei componenti commerciali. Questo significa che non vengono progettati *ex novo*, ma scelti dai cataloghi dei vari fornitori in funzione delle caratteristiche di cui si ha necessità. Viene da sé che per scegliere correttamente abbiamo bisogno anzitutto di conoscere bene il sistema meccanico che stiamo progettando, comprese le masse in gioco, i carichi da movimentare e i task da eseguire. Sulla base di ciò bisogna ricavare dei parametri che, confrontati con i datasheet del costruttore, portano alla scelta del motore ottimale. Per individuare tali parametri, dunque, è bene osservare i datasheet. Da essi notiamo che sono sempre specificati i seguenti valori:

- **Coppia**: di solito viene specificata sia quella massima che quella nominale;
- **Velocità**: anche in questo caso sono specificati entrambi i valori.

Inoltre, solitamente vengono riportate anche le curve caratteristiche di tali motori (coppia-velocità) con evidenziate le zone di funzionamento costante. Queste sono molto utili in quanto, se si dispone della caratteristica del carico, un metodo di dimensionamento consiste nel verificare che questa sia interamente contenuta all'interno della caratteristica del motore. Talvolta è riportata anche la **potenza**, che però può essere ricavata a partire dalla coppia e dalla velocità.

Alla luce di questo, il progettista che sta lavorando allo sviluppo di un robot solitamente deve, una volta definite le specifiche dimensionali e dinamiche, calcolare le grandezze cinematiche e dinamiche relative ad ogni giunto, manualmente, aiutandosi al più con fogli di calcolo, oppure tramite simulazioni cinematiche e dinamiche con software CAD, dunque passando per la modellazione, la definizione di vincoli cinematici e la simulazione. È quindi un processo abbastanza lungo che richiede una buona dose di calcoli. L'esistenza di uno strumento a cui basta fornire come input la geometria del robot e la legge di moto per avere in output tutti i parametri cinematici e dinamici di cui abbiamo bisogno, rappresenta dunque un grande vantaggio in questa fase della progettazione, permettendo di risparmiare tempo prezioso.

2. STRUMENTI E LOGICA

In questo capitolo, sulla base delle riflessioni fatte, verranno descritti gli strumenti con i quali implementare e verificare il *tool* e la logica con cui esso funziona.

2.1. Software utilizzati

La strumentazione con cui è stato portato avanti il progetto, com'è intuibile, consiste di soli software, utilizzati per le seguenti fasi di sviluppo:

- **Codifica**: abbiamo dovuto scegliere un linguaggio e un ambiente di sviluppo congeniale alla creazione di questo tipo di strumento di calcolo;
- **Verifica**: per accertarci del corretto funzionamento del calcolatore, l'idea è quella di sfruttare un software di modellazione e simulazione per avere un parametro di riferimento attendibile, per poi confrontare i risultati ottenuti in entrambi i modi e valutarne la compatibilità.

In definitiva abbiamo utilizzato tre programmi.

2.1.1. GNU Octave

Per lo sviluppo del tool la scelta è ricaduta su *Octave*. Octave è un software di analisi numerica il cui linguaggio presenta tutte le strutture dati tipiche del linguaggio C e che presenta i seguenti punti di forza che lo rendono ideale per i propositi di questa tesi:

- **Calcolo matriciale**: come vedremo nei capitoli successivi, gran parte dei calcoli interni al *tool* sono di tipo matriciale, dunque, un software che permette il calcolo matriciale è perfetto per lo sviluppo degli algoritmi necessari;
- **Compatibilità con Matlab**: questo calcolatore è in gran parte compatibile con Matlab, questo permette di attingere, in fase di *coding*, ad esempi, documentazione e tutorial sia relativi a Octave che a Matlab, un vantaggio che può tornare molto utile;
- **Compatibilità con i principali sistemi operativi**: il software gira su *windows*, *macOS*, *Unix* e *Linux.*
- **Open source**: non da ultimo, Octave è un software *Open Source*, il che significa che non richiede costose licenze per essere installato, dunque, il tool sviluppato con esso è automaticamente molto accessibile;

2.1.2. PTC Creo

Per ottenere i valori con cui confrontare i risultati del *tool* è stato scelto il software Creo, rilasciato da *Parametric Technology Corporation* (PTC). Trattasi di un programma di modellazione 3D, che permette, oltre che la modellazione di componenti, assiemi e realizzazione di tavole, anche di eseguire simulazioni attraverso il modulo *Mechanism*. Tramite esso, infatti, dopo aver creato un cinematismo come assieme di componenti opportunamente vincolati (in modo da lasciare svincolati i g.d.l. desiderati), si possono assegnare dei servomotori ai vincoli con cui movimentare la macchina. Eseguita la simulazione, specificandone la tipologia e la durata, si può passare alla misurazione delle grandezze cinematiche di nostro interesse, di cui si ottiene l'andamento nel tempo della simulazione. Tali andamenti possono essere facilmente esportati come file Excel.

2.1.3. Excel

Arriviamo dunque all'ultimo dei software utilizzato per questo progetto, ovvero *Excel*. Come detto nel paragrafo precedente, con Creo è possibile esportare i valori ottenuti come file Excel, dunque, l'idea è stata quella di utilizzare dei fogli di calcolo per eseguire il confronto tra questi risultati e quelli ottenuti dallo strumento di calcolo sviluppato, i quali, per essere considerati accettabili devono discostarsi il meno possibile. Nel cap. 5 vedremo come sono stati effettuati i confronti e i risultati ottenuti.

2.2. Funzionamento del tool

È stato definito il problema, è stato contestualizzato, è stato chiarito cosa si intende ottenere e perché e sono stati presentati gli strumenti utilizzati per ottenerlo. Non resta a questo punto che descrivere i principi che permetteranno allo strumento di trasformare determinati input in output, la cui matematica verrà approfondita nel capitolo successivo, mentre l'implementazione verrà analizzata e spiegata in dettaglio nel cap. 4. In particolare, possiamo suddividere il *tool* in quattro fasi principali:

- **Input**: questa è la fase in cui l'utente interagisce con il calcolatore. In particolare, esso dovrà inserire in un apposito *script* i dati relativi al cinematismo (lunghezza delle varie aste, la loro massa e la massa dei giunti) e i dati relativi al moto desiderato dal robot. Come vedremo nel capitolo dedicato all'implementazione, questi verranno trattati diversamente nel caso

puramente planare e in quello spaziale. Lo script di input verrà caricato all'interno del *main code* in modo da avere nel *workspace* le variabili desiderate dall'utente.

- **Controllo**: ogni qual volta vi è un'interazione della macchina con l'utente, bisogna assicurarsi che il codice sia robusto rispetto ad eventuali errori dei dati inseriti. Per questo motivo sono state sviluppate anche delle routine di controllo sui dati immessi, in particolar modo viene verificato che la traiettoria desiderata ricada nello spazio di lavoro del robot e che l'accelerazione richiesta non porti a casi degeneri della legge di moto, che, come vedremo, sarà di tipo trapezoidale.
- **Computo**: per il computo abbiamo due strade, a seconda del tipo di applicazione desiderata:
    - se è importante che il robot segua una traiettoria ben precisa, vengono assegnate le grandezze cinematiche relative al terminale e ricavata la coppia richiesta ai giunti per ottenere tale traiettoria mediante il computo della *dinamica inversa* del cinematismo, che a sua volta, per poter essere effettuata, ha bisogno che sia affrontata prima *l'analisi cinematica inversa*, per ottenere posizioni, velocità e accelerazioni dei giunti;
    - se l'obiettivo è minimizzare il tempo ciclo, si condurrà un'analisi di tipo diretto assegnando le coppie ai giunti e calcolando la traiettoria che deriva dal loro moto;

  In questo lavoro sarà trattato solo il primo caso, in quanto quello di maggior utilità per le attuali necessità del team di progettazione, ma sicuramente anche il secondo caso è di interesse e rappresenterà un'ottima possibilità di sviluppo ulteriore del tool per lavori futuri.

- **Output**: al termine del computo, che verrà eseguito attraverso funzioni nascoste all'utente, i risultati saranno visualizzati attraverso una figura che mostrerà quattro *plot:*
    - Un plot tridimensionale che mostra la geometria del robot e le configurazioni che assume nei vari istanti di tempo per passare dalla posizione iniziale a quella finale, in questa maniera l'utente può verificare che sia la geometria del robot, sia il movimento richiesto siano quelli dichiarati negli input;
    - Un plot che mostra l'andamento della *velocità* dei giunti durante il moto;
    - Un plot che mostra l'andamento *dell'accelerazione* dei giunti durante il moto;
    - Un plot che mostra l'andamento della *coppia* dei giunti durante il moto.

3. STUDIO CINEMATICA E DINAMICA

In questa sezione verranno richiamati gli studi di cinematica e dinamica che confluiranno nell'algoritmo del calcolatore, in quanto è bene avere presente gli aspetti fisici e matematici che regolano il problema prima di implementarli correttamente come algoritmi.

3.1. Scelta del cinematismo: robot SCARA

Il cinematismo che è stato scelto per essere modellato è un robot SCARA (vedi §1.1); la scelta non è stata casuale ma mossa dalle seguenti motivazioni:

- È un meccanismo molto utilizzato nel contesto industriale e soprattutto delle macchine automatiche;
- Nonostante si trovino robot di questo tipo già commercializzati, in quanto azienda di consulenza, SMARTENGINEERING s.p.a. si trova spesso a dover trovare soluzioni alternative e personalizzate per le particolari esigenze dei clienti;
- È un cinematismo che può essere scomposto in sottosistemi, in quanto a livello planare è riconducibile ad un robot 2R o ad uno 3R, questo comporta dei vantaggi:
    - È possibile implementare il tool *step-by-step* alleggerendo il lavoro svolto per lo studio cinematico e per lo sviluppo degli algoritmi;
    - Permette di avere di fatto tre tool differenti e utilizzabili indipendentemente, uno per il 2R, uno per il 3R e infine quello per lo SCARA completo;
    - È un meccanismo che contiene sia giunti di tipo rotoidale che giunti prismatici, il che rende il modello più completo e con un carattere di generalità maggiore.

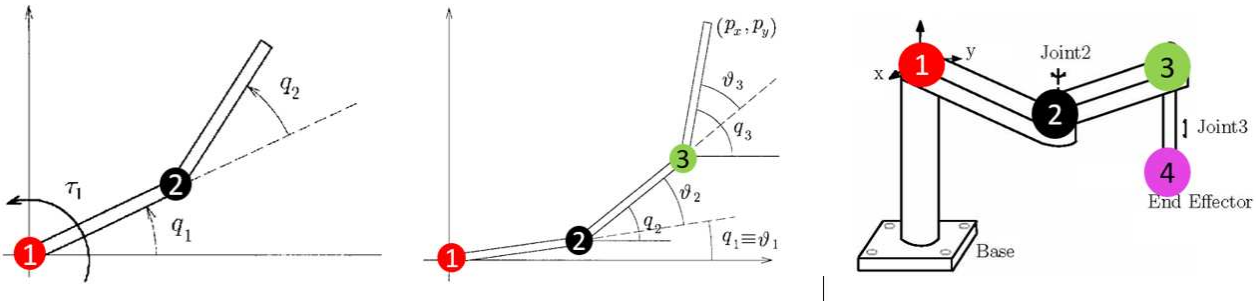


Figure 1 - I cinematismi implementati, da sx: 2R, 3R, SCARA

3.2. Cinematica

Al fine di ottenere il modello cinematico, in questo paragrafo verranno richiamati i principi chiave che permettono lo studio e la comprensione della cinematica dei manipolatori industriali.

Tralasciando i robot paralleli, che sono composti da una cinematica chiusa, i manipolatori industriali sono costituiti da una catena aperta di membri connessi in modo seriale, dal telaio al terminale, e spesso indicati con una numerazione che va da *0* per il telaio a *n* per il terminale. Gli *n+1* membri sono connessi da *n* giunti, azionati ognuno da un attuatore, pertanto, possiamo dire che hanno *n mobilità* o *n assi* e il moto relativo tra un membro e quello adiacente è causato dal giunto che li connette [2] [3].

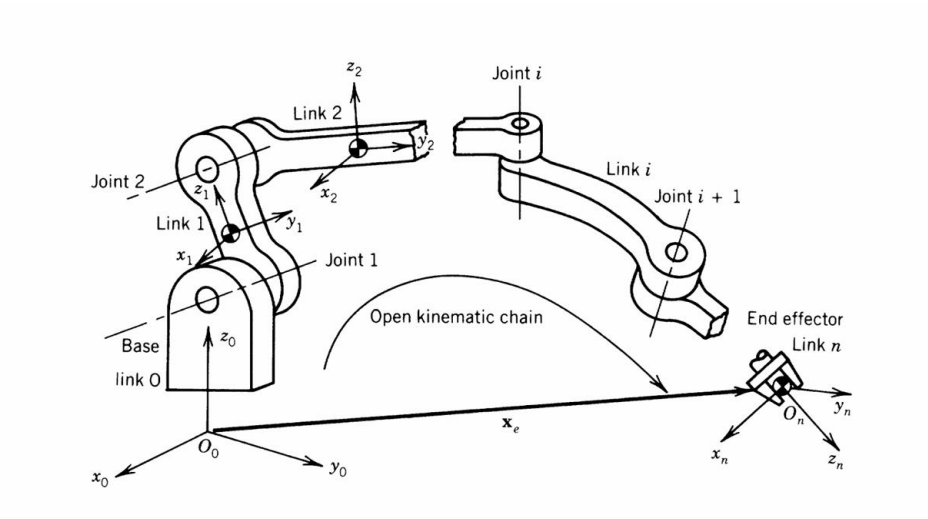


Figure 2 - Schema generale di un manipolatore seriale

È possibile ottenere la posizione e l'orientamento del terminale andando ad assegnare a ciascun membro un sistema di riferimento solidale e effettuando le operazioni di trasformazione da un sistema all'altro. Vien da sé che è importante definire un metodo per la giusta assegnazione delle terne; solitamente si fa ricorso alla *convenzione di Denavit-Hartenberg.*

3.2.1. Convenzione di Denavit-Hartenberg

Questa rappresentazione è utilizzata per descrivere la relazione cinematica tra due membri adiacenti di un manipolatore seriale attraverso *quattro parametri*. Ne esistono diverse varianti, tutte ugualmente valide, in questo lavoro si fa riferimento alla seguente.

Si prendano in esame due membri generici connessi da un giunto, come in figura 14:

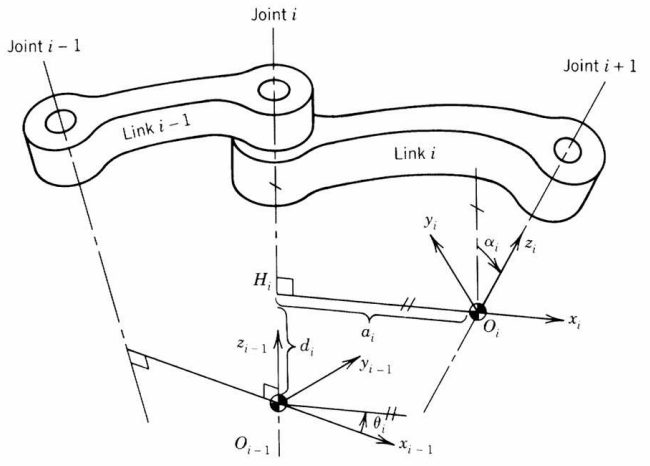


Figure 3 - Disposizione delle terne secondo la convenzione di D-H

Si indichi con *i* la terna solidale al membro *i-esimo*, che viene posizionata in corrispondenza del giunto *(i+1)-esimo*. L'origine O_i di questa terna è posizionata sull'intersezione tra l'asse del giunto *(i+1)-esimo* e la normale comune all'asse stesso e a quello dell'asse precedente. A questo punto viene posizionato l'asse X_i lungo tale normale, l'asse Z_i lungo l'asse del giunto *(i+1)-esimo* e infine viene posizionato l'asse Y_i in modo da completare una terna ortonormale destrorsa.

Possiamo localizzare la posizione di una terna relativamente a quella adiacente in modo determinato attraverso i quattro parametri α_i, a_i, d_i, θ_i evidenziati in figura 15:

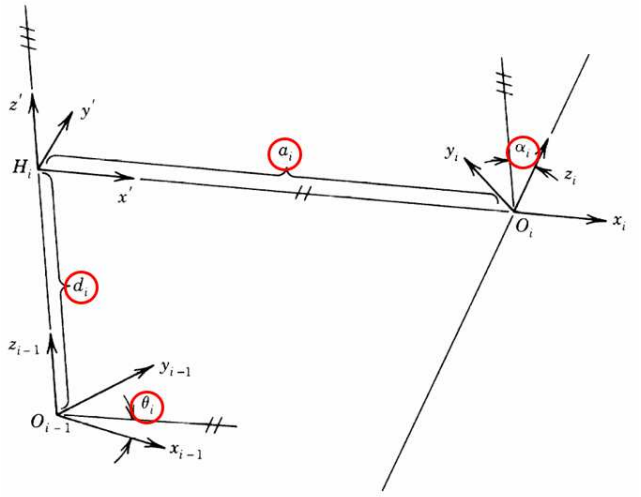


Figure 4 - I quattro parametri di Denavit-Hartenberg

È bene specificare che abbiamo un diverso comportamento di questi parametri in relazione al tipo di giunto:

- Se il giunto è di tipo *rotoidale*, il parametro θ_i varia durante il moto mentre d_i resta costante;
- Viceversa, per giunti *prismatici* abbiamo che d_i varia mentre θ_i resta costante.

In entrambi i casi il parametro che varia è denominato *variabile di giunto* o *di coppia*.

Come possiamo notare nella figura 15, è presente una terna intermedia H_i che ci aiuta nel ricavare la matrice di trasformazione tra il sistema *i-esimo* e quello *(i+1)-esimo* nel seguente modo. Siano indicati con ${}^{i}X$, ${}^{int}X$, ${}^{i-1}X$ dei vettori posizione nei tre sistemi di riferimento; possiamo esprimere la trasformazione di coordinate che porta da ${}^{i-1}X$ a ${}^{int}X$ come:

$$ {}^{i-1}X = {}^{i-1}_{int}T\,{}^{int}X $$

E quella che porta da ${}^{int}X$ a ${}^{i}X$ come:

$$ {}^{int}X = {}^{int}_{i}T\ {}^{i}X $$

Dove le due matrici T sono rispettivamente:

$$ {}^{i-1}_{\text{int}}T = \begin{bmatrix} \cos\theta_i & -\text{sen}\,\theta_i & 0 & 0 \\ \text{sen}\,\theta_i & \cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad {}^{i}_{\text{int}}T = \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & \cos\alpha_i & -\text{sen}\,\alpha_i & 0 \\ 0 & \text{sen}\,\alpha_i & \cos\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Pertanto, possiamo esprimere la trasformazione che porta da ${}^{i-1}X$ a ${}^{i}X$ come:

$$ {}^{i-1}X = {}^{i-1}_{int}T\,{}^{int}_{i}T\ {}^{i}X $$

Dove:

$$ {}^{i-1}_{int}T\,{}^{int}_{i}T = {}^{i-1}_{i}T $$

$$ {}^{i-1}_{i}T = \begin{bmatrix} \cos\theta_i & -\text{sen}\,\theta_i \cos\alpha_i & \text{sen}\,\theta_i \,\text{sen}\,\alpha_i & a_i \cos\theta_i \\ \text{sen}\,\theta_i & \cos\theta_i \cos\alpha_i & -\cos\theta_i \,\text{sen}\,\alpha_i & a_i \,\text{sen}\,\theta_i \\ 0 & \text{sen}\,\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

La matrice così ottenuta permette di esprimere la posizione e l'orientamento del sistema *i* relativamente al sistema *i-1*; se volessimo, all'opposto, passare dal sistema *i-1* a *i*, la matrice da utilizzare sarebbe:

$$ {}^{i}_{i-1}T = \begin{bmatrix} \cos\theta_i & sen\theta_i & 0 & -a_i \\ -sen\theta_i \cos\alpha_i & \cos\theta_i \cos\alpha_i & sen\alpha_i & -d_i sen\alpha_i \\ sen\theta_i sen\alpha_i & -\cos\theta_i sen\alpha_i & \cos\alpha_i & -d_i \cos\alpha_i \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Per il telaio e il terminale, in cui non è possibile individuare una normale comune, si sfrutta la seguente convenzione:

- **Telaio:** la terna può essere posta in qualunque punto dell'asse del giunto 1, a patto che l'asse Z_0 sia parallelo all'asse del giunto stesso:

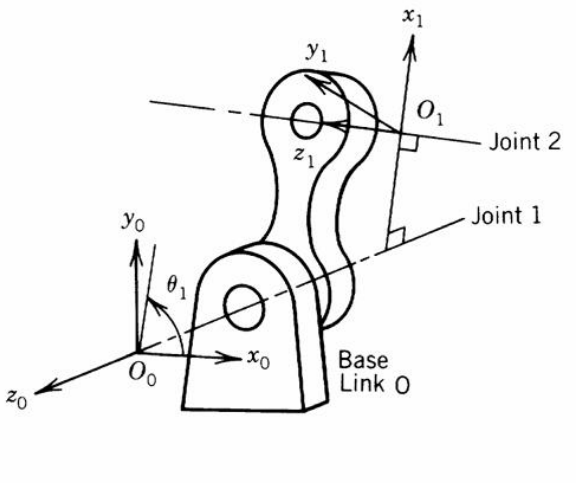


Figure 5 - Convenzione di D-H per il telaio

- **Terminale:** la terna può essere posta in qualsiasi punto del terminale a patto che l'asse X_n intersechi l'asse dell'ultimo giunto perpendicolarmente.

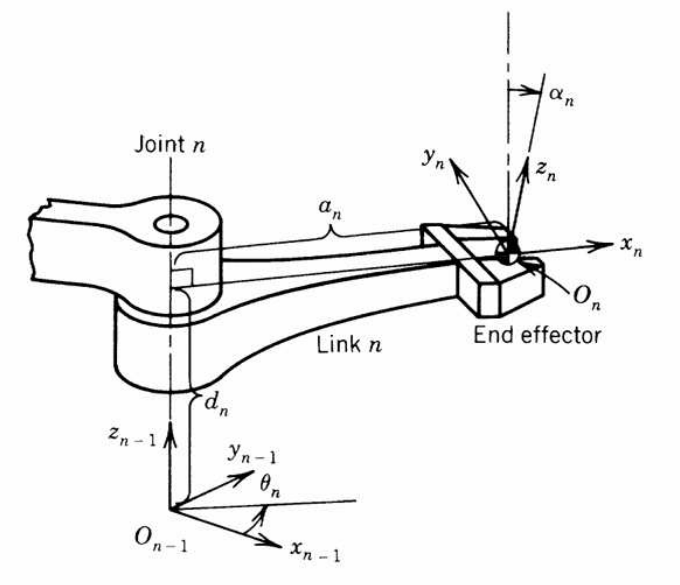


Figure 6 - Convenzione di D-H per il terminale

Abbiamo visto la rappresentazione generale di Denavit-Hartenberg; questa potrà essere adattata in base alla geometria specifica del cinematismo studiato affinché la scelta del posizionamento delle terne sia la più comoda. Nel caso del robot SCARA, ad esempio, avendo gli assi dei giunti rotoidali fra loro paralleli, una buona scelta delle terne risulta la seguente:

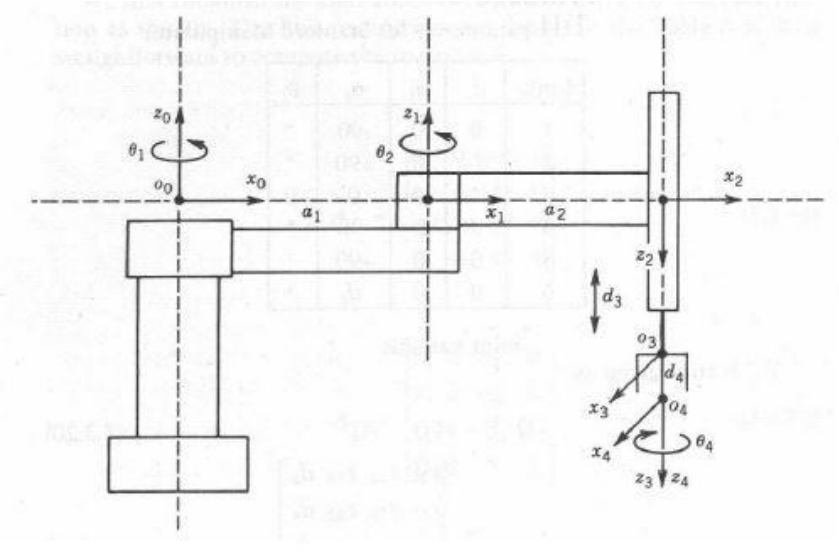


Figure 7 - Posizionamento delle terne nel robot SCARA secondo la convenzione D-H

Con questa scelta delle terne, i parametri D-H sono:

link	θ_i	d_i	a_i	α_i
1	θ_1^*	0	a_1	0
2	θ_2^*	0	a_2	180°
3	0	d_3^*	0	0
4	θ_4^*	d_4	0	0

Figure 8 - Parametri di Denavit-Hartenberg per il robot SCARA

Dove sono state contrassegnate con l'asterisco le variabili di giunto, tutti gli altri sono valori costanti dati dalla geometria del cinematismo. [2] [3]

3.2.2. Analisi di posizione, velocità e accelerazione

Abbiamo visto come le matrici di trasformazione siano ottenibili grazie alla conoscenza di quattro parametri, tre dei quali costanti, uno variabile.

Grazie a tali matrici è possibile condurre due tipologie di analisi in termini di posizione:

- **Analisi cinematica diretta:** consiste nel determinare la posizione e l'orientamento del terminale noti gli spostamenti dei giunti; è un tipo di analisi semplice e che porta sempre ad una e una sola soluzione.
- **Analisi cinematica inversa**: consiste nel determinare quali siano gli spostamenti da imporre ai giunti per far sì che il terminale si trovi in un punto assegnato e con un orientamento prefissato; a differenza dell'analisi diretta questa non è sempre risolvibile in forma chiusa se non per alcuni casi particolari e può portare a nessuna, una o a un numero finito o anche infinito di soluzioni. Ogni soluzione è una possibile configurazione del robot e spesso si opta per una piuttosto che per un'altra sulla base di motivazioni fisiche, come la presenza di ostacoli o l'effettiva possibilità di compiere un dato movimento con coppie realizzabili. A livello matematico il problema viene formalizzato attraverso applicazioni tra due spazi notevoli:

- **Spazio dei giunti**: è lo spazio vettoriale che raccoglie le coordinate di giunto

$$q = \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_i \\ \vdots \\ q_n \end{bmatrix}$$

- **Spazio cartesiano**: è lo spazio vettoriale che contiene la posizione e l'orientamento del terminale.

$$p = \begin{bmatrix} x \\ y \\ z \\ \alpha \\ \beta \\ \gamma \end{bmatrix}$$

Dunque, possiamo esprimere il problema cinematico diretto come:

$$p = \Phi(q)$$

Dove:

$$q \in \Re^n, \quad p \in \Re^m, \quad \Phi : \Re^n \mapsto \Re^m$$

Mentre il problema cinematico inverso si ottiene invertendo l'applicazione precedente:

$$q = \Phi^{-1}(p)$$

Dove:

$$p \in \Re^m, \quad q \in \Re^n, \quad \Phi^{-1} : \Re^m \mapsto \Re^n$$

In genere un manipolatore è risolubile quando è possibile determinare un algoritmo che permetta di ottenere tutte le soluzioni del problema inverso. La risoluzione può avvenire in due modi:

- **Soluzioni numeriche**: sono lente in quanto costituite da algoritmi iterativi; pertanto, poco indicate per la risoluzione in tempo reale della cinematica dei robot;
- **Soluzioni in forma chiusa**: sono basate su espressioni analitiche e sulla soluzione di equazioni polinomiali fino al quarto grado.

In generale i manipolatori in catena aperta con 6 gradi di libertà (o meno) sono risolubili, in forma chiusa solo per casi particolari, in cui molti dei parametri D-H sono nulli o pari a 90°. Questo è proprio il caso dello SCARA; infatti, dai parametri di D-H che abbiamo definito, possiamo calcolare la matrice di trasformazione come:

$$\begin{bmatrix} c_{12}c_4 + s_{12}s_4 & -c_{12}s_4 + s_{12}c_4 & 0 & a_1c_1 + a_2c_{12} \\ s_{12}c_4 - c_{12}s_4 & -s_{12}s_4 - c_{12}c_4 & 0 & a_1s_1 + a_2s_{12} \\ 0 & 0 & -1 & -d_3 - d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \left[\begin{array}{c|c} \mathbf{R} & \mathbf{o} \\ \hline 0 & 1 \end{array}\right]$$

Dove **R** e **o** sono rispettivamente l'orientamento e la posizione del terminale riferiti alla terna fissa. Possiamo arrivare ad una soluzione in forma chiusa se **R** è della forma:

$$\mathbf{R} = \begin{bmatrix} c_\alpha & s_\alpha & 0 \\ s_\alpha & -c_\alpha & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

Con:

$$\alpha = \theta_1 + \theta_2 - \theta_4 = atan2(r_{11}, r_{12})$$

Per calcolare θ_1 e θ_2 basterà proiettare il robot sul piano XY e ricondursi in questo modo al caso del robot planare 2R:

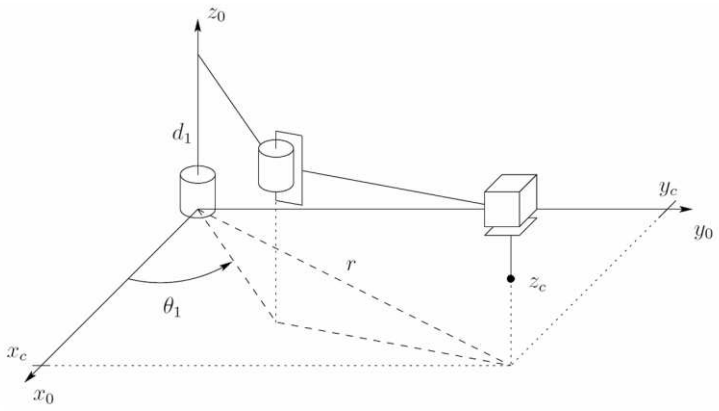


Figure 9 - Robot SCARA proiettato sul piano XY

A questo punto possiamo esprimere l'angolo θ_2 come:

$$\theta_2 = atan2(c_2, \pm\sqrt{1 - c_2})$$

Dove:

$$c_2 = \frac{o_x^2 + o_y^2 - a_1^2 - a_2^2}{2a_1a_2}$$

Mentre ricaviamo θ_1 dall'equazione:

$$\theta_1 = atan2(o_x, o_y) - atan2(a_1 + a_2c_2, a_2s_2)$$

Noti θ_1 e θ_2 possiamo ricavare θ_4 dall'equazione iniziale:

$$\theta_4 = \theta_1 + \theta_2 - \alpha = \theta_1 + \theta_2 - atan2(r_{11} - r_{12})$$

Per quanto riguarda l'ultima variabile di giunto, quella d_3, si ricava semplicemente come:

$$d_3 = o_z + d_4$$

In termini di velocità, possiamo calcolare la velocità dei vari membri in modo progressivo dalla base al terminale aggiungendo le componenti di velocità che nascono nei giunti, dove per velocità dei membri si intendono la velocità lineare dell'origine della terna ad esso associata e la velocità

angolare del membro stesso. Come abbiamo detto, nascono nuove componenti di velocità in funzione del giunto interposto tra i due link consecutivi, pertanto, è doveroso fare una distinzione tra:

- **Giunto rotoidale**: in questi giunti le velocità angolari possono essere sommate (purché espresse nello stesso sistema di riferimento), mentre quelle lineari vanno trasportate tenendo conto delle velocità relative:

$$\boldsymbol{\omega}_{i+1} = \boldsymbol{\omega}_i + \dot{q}_{i+1}\,\hat{\mathbf{z}}_i$$

$$\mathbf{v}_{i+1} = \mathbf{v}_i + \boldsymbol{\omega}_{i+1} \times {}^{U}\left({}^{i}\mathbf{P}_{i+1}\right)$$

- **Giunto prismatico**: in questo caso la velocità angolare è la stessa, mentre la velocità lineare si trasporta tenendo conto delle velocità relative:

$$\boldsymbol{\omega}_{i+1} = \boldsymbol{\omega}_i$$

$$\mathbf{v}_{i+1} = \mathbf{v}_i + \boldsymbol{\omega}_i \times {}^{U}\left({}^{i}\mathbf{P}_{i+1}\right) + \dot{q}_{i+1}\hat{\mathbf{z}}_i$$

Abbiamo già formalizzato il problema cinematico in termini matematici come applicazione tra lo spazio dei giunti e quello cartesiano. Volendo trovare una relazione che lega le velocità nei medesimi spazi, possiamo raccogliere le derivate parziali del problema cinematico nella seguente matrice, detta ***Jacobiana***:

$$\mathbf{J} = \begin{bmatrix} \dfrac{\partial \Phi_1}{\partial q_1} & \cdots & \dfrac{\partial \Phi_1}{\partial q_n} \\ \\ \dfrac{\partial \Phi_m}{\partial q_1} & & \dfrac{\partial \Phi_m}{\partial q_n} \end{bmatrix}$$

Questa matrice lega le velocità dei giunti a quelle del terminale, infatti prendendo:

$$\mathbf{p} = \boldsymbol{\Phi}(\mathbf{q})$$

$$\delta p_m = \sum_{i=1}^{n} \frac{\partial \Phi_m(\mathbf{q})}{\partial q_i}\,\delta q_i$$

E dividendo rispetto al differenziale del tempo *dt* si ottiene:

$$\dot{p}_m = \sum_{i=1}^{n} \frac{\partial \Phi_m(\mathbf{q})}{\partial q_i} \dot{q}_i$$

In cui si nota come la Jacobiana (o Jacobiano) sia funzione delle variabili di giunto.

Possiamo distinguere due tipi di Jacobiano:

- **Jacobiano geometrico**: si ha quando si fa riferimento al vettore velocità nello spazio cartesiano:

$$^0\dot{\mathbf{p}} = \begin{bmatrix} ^0\mathbf{v} \\ ^0\boldsymbol{\omega} \end{bmatrix}$$

- **Jacobiano analitico**: si ha quando non esprime direttamente la velocità angolare del terminale:

$$^0\dot{\mathbf{p}} = \begin{bmatrix} ^0\mathbf{v} \\ ^0\dot{\boldsymbol{\Theta}} \end{bmatrix}$$

Per lo SCARA, è possibile definire il seguente Jacobiano geometrico:

$$\mathbf{J} = \begin{bmatrix} -a_1 s_1 - a_2 s_{12} & -a_2 s_{12} & 0 & 0 \\ a_1 c_1 + a_2 c_{12} & a_2 c_{12} & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix} \in \Re^{6\times 4}$$

Tuttavia, nel nostro caso, così come per l'analisi di posizione, l'intento è quello di ottenere le velocità di giunto a partire dalle velocità (note) del terminale. Si parla anche in questo caso di **analisi inversa** e si ottiene proprio invertendo la relazione che lega le velocità attraverso il jacobiano e, dunque, invertendo il jacobiano stesso:

$$\dot{\mathbf{q}} = \mathbf{J}^{-1}(\mathbf{q})\, \dot{\mathbf{p}}$$

È evidente che ciò risulta possibile solo se:

- È stata precedentemente condotta l'analisi di posizione per avere le variabili di giunto *q* necessarie al calcolo dello Jacobiano;
- Lo Jacobiano deve risultare invertibile, questo è possibile se la matrice risulta non singolare e dunque il **determinante** deve risultare **non nullo.** Le configurazioni in cui si ha l'annullamento di questo determinante sono pertanto considerate **singolari** e sono configurazioni in cui si ha una perdita di gradi di libertà dal punto di vista dello spazio cartesiano, ovvero vi sono direzioni in cui non è possibile muovere il terminale qualsiasi sia la combinazione di velocità dei giunti. Ciò avviene spesso ai confini dell'area di lavoro. Nel caso specifico dello SCARA questo si verifica quando il gomito è completamente steso o completamente ritratto, ed è una singolarità comune anche al robot 2R con cui condivide lo stesso moto a livello planare:

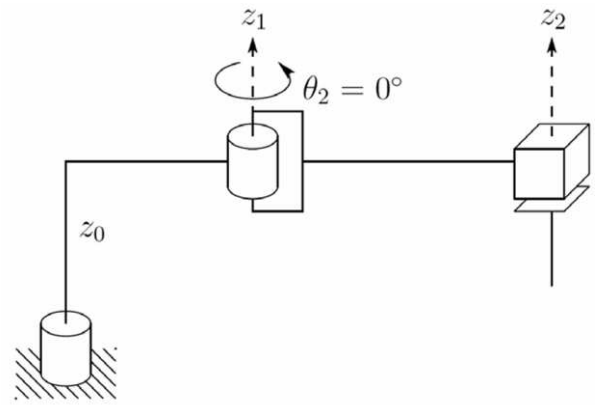


Figure 10 - Configurazione singolare per lo SCARA

Giungiamo infine all'analisi di accelerazione. Analogamente al caso delle velocità è possibile calcolare iterativamente dalla base al terminale tenendo conto della propagazione delle accelerazioni dovuta al tipo di giunto:

- **Giunto rotoidale**: Le accelerazioni angolari si ricavano per semplice derivazione delle velocità angolari, mentre per quanto riguarda quelle lineari, queste si trasportano tenendo conto delle accelerazioni relative e del fatto che non vi sono traslazioni relative tra le terne;

$$\dot{\boldsymbol{\omega}}_{i+1} = \dot{\boldsymbol{\omega}}_i + \ddot{q}_{i+1}\hat{\mathbf{z}}_i + \boldsymbol{\omega}_i \times \dot{q}_{i+1}\hat{\mathbf{z}}_i$$

$$\dot{\mathbf{v}}_{i+1} = \dot{\mathbf{v}}_i + \dot{\boldsymbol{\omega}}_{i+1} \times {}^{U}\left({}^{i}\mathbf{P}_{i+1}\right) + \boldsymbol{\omega}_{i+1} \times \left(\boldsymbol{\omega}_{i+1} \times {}^{U}\left({}^{i}\mathbf{P}_{i+1}\right)\right)$$

- **Giunto prismatico**: le accelerazioni angolari restano invariate, mentre quelle lineari si trasportano tenendo conto delle accelerazioni relative e, in questo caso, anche della traslazione relativa tra le terne.

$$\dot{\boldsymbol{\omega}}_{i+1} = \dot{\boldsymbol{\omega}}_i$$

$$\dot{\mathbf{v}}_{i+1} = \dot{\mathbf{v}}_i + \ddot{q}_{i+1}\hat{\mathbf{z}}_i + \dot{\boldsymbol{\omega}}_i \times {}^{U}\left({}^{i}\mathbf{P}_{i+1}\right) + 2\boldsymbol{\omega}_i \times \dot{q}_{i+1}\hat{\mathbf{z}}_i + \boldsymbol{\omega}_i \times \left(\boldsymbol{\omega}_i \times {}^{U}\left({}^{i}\mathbf{P}_{i+1}\right)\right)$$

Avendo già calcolato lo Jacobiano è possibile condurre l'analisi anche per derivazione del problema cinematico di velocità:

$$\dot{\mathbf{p}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}$$

$$\Downarrow$$

$$\ddot{\mathbf{p}} = \frac{d}{dt}(\mathbf{J}(\mathbf{q})\dot{\mathbf{q}})$$

$$\ddot{\mathbf{p}} = \mathbf{J}(\mathbf{q})\ddot{\mathbf{q}} + \dot{\mathbf{J}}(\mathbf{q},\dot{\mathbf{q}})\,\dot{\mathbf{q}}$$

Da cui, invertendo, si ottiene l'analisi inversa di accelerazione, che permette di ricavare le accelerazioni di giunto a partire da quelle del terminale:

$$\ddot{\mathbf{q}} = \mathbf{J}^{-1}(\mathbf{q})\ddot{\mathbf{p}} - \mathbf{J}^{-1}(\mathbf{q})\dot{\mathbf{J}}(\mathbf{q},\dot{\mathbf{q}})\,\dot{\mathbf{q}}$$

3.3. Dinamica

Solo al termine di una corretta analisi cinematica è possibile condurre l'analisi dinamica del sistema, che permette di ricavare le relazioni che legano il moto del terminale con le coppie applicate ai giunti. Lo studio consiste nel risolvere equazioni differenziali o algebrico-differenziali per ottenere le equazioni del moto; queste possono essere ricavate attraverso vari approcci:

- **Metodo di Newton-Eulero**: consiste nello scrivere le equazioni cardinali della dinamica per ogni membro e risolvere il sistema, ha il vantaggio di un'immediata comprensione fisica del problema e permette di ottenere anche le reazioni vincolari, che tuttavia dovranno essere rimosse con manipolazioni algebriche per ottenere le equazioni del moto in forma esplicita;
- **Metodo di Lagrange**: è un approccio di tipo energetico, che non richiama le forze reattive, fornisce un modello in forma compatta sfruttando le coordinate lagrangiane.
- **PLV**: anche il principio dei lavori virtuali è un metodo energetico; il principio afferma che la condizione necessaria e sufficiente per l'equilibrio di un sistema ideali di corpi rigidi è che sia nullo il lavoro delle forze esterne su di esso agenti, comprese quelle di inerzia, a seguito di spostamenti virtuali, dove per spostamento virtuale si intende uno spostamento infinitesimo, ipotizzato, di durata nulla e arbitrario purché compatibile con i vincoli.

Anche in questo caso i tipi di analisi che è possibile condurre sono due:

- **Analisi diretta**: consiste nel determinare il moto del terminale in funzione di assegnate coppie di giunto, forze e momenti esterni. Comporta la risoluzione di equazioni differenziali non lineari e viene condotta attraverso solutori numerici;
- **Analisi inversa**: consiste nell'assegnare una traiettoria al terminale e ricavare le coppie dei giunti che permettono di ottenerla. È abbastanza immediata in quanto consiste nella valutazione di un'espressione nota in forma chiusa.

L'analisi dinamica del sistema può essere sempre ricondotta ad una equazione vettoriale del tipo:

$$\boldsymbol{\tau} + \mathbf{J}^T \mathbf{F}_e = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{V}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) + \mathbf{D}(\mathbf{q}, \dot{\mathbf{q}})$$

Dove:

- $\mathbf{q}$ $[n \times 1]$ vettore degli spostamenti dei giunti
- $\boldsymbol{\tau}$ $[n \times 1]$ vettore delle forze/coppie di attuazione
- $\mathbf{J}$ $[m \times n]$ matrice Jacobiana
- $\mathbf{F}_e$ $[m \times 1]$ vettore delle forze e momenti esterni
- $\mathbf{M}(\mathbf{q})$ $[n \times n]$ matrice di massa (o di inerzia) del manipolatore
- $\mathbf{V}(\mathbf{q}, \dot{\mathbf{q}})$ $[n \times 1]$ vettore di termini centrifughi e di Coriolis
- $\mathbf{G}(\mathbf{q})$ $[n \times 1]$ vettore di termini gravitazionali
- $\mathbf{D}(\mathbf{q}, \dot{\mathbf{q}})$ $[n \times 1]$ vettore dei termini di attrito (viscoso e secco)

Nel caso in esame non saranno considerati forze e momenti esterni e verranno trascurati i termini di attrito, mentre saranno calcolate le matrici di massa, dei termini centrifughi e dei termini gravitazionali, i quali, come si nota dalla formula, sono funzione degli stati del sistema, dunque, sono necessari i valori di posizione, velocità e accelerazione precedentemente calcolati con l'analisi cinematica. [2] [3]

3.4. Pianificazione di traiettorie

Come è stato detto nel paragrafo precedente, l'analisi dinamica inversa consiste nel determinare le coppie da applicare ai giunti affinché il terminale compia una traiettoria assegnata. È dunque intuibile come sia importante pianificare le traiettorie che si intendono ottenere al fine di avere movimenti effettivamente realizzabili, poco onerosi dal punto di vista dei controllori e degli azionamenti e che non eccitino le frequenze di risonanza proprie del sistema. [3]

In genere è possibile effettuare la pianificazione in due modi:

- Nello **spazio dei giunti**: si programmano in funzione del tempo le variabili di giunto e le loro prime due derivate; ha il vantaggio di non fare ricorso all'analisi cinematica, ma allo stesso tempo non permette di avere una traiettoria desiderata in quanto questa è il risultato del moto dei giunti.
- Nello **spazio cartesiano**: si programmano posizione, velocità ed accelerazione del terminale e si proiettano i risultati nello spazio dei giunti; il vantaggio è quello di avere una traiettoria ben determinata, ma fa ricorso alla cinematica inversa, quindi potrebbe portare a configurazioni singolari.

I tipi di traiettorie che si possono pianificare si possono suddividere in due gruppi:

- **Leggi polinomiali semplici**: viene descritto con un unico polinomio, se si vuole il passaggio per *N* punti, bisognerà usare un polinomio:
    - di grado (*N-1*) se si vuole imporre il solo passaggio;
    - di grado (*2N-1*) se si vuole imporre anche la velocità;
    - di grado (*3N-1*) se si vuole imporre anche la accelerazione.

  I gradi più semplici che si possono utilizzare sono 3, 5 e 7.

- **Polinomi raccordati**: sono costituiti da più polinomi raccordati in modo continuo e i più comuni sono:
    - Traiettoria 4-3-4;
    - Traiettoria 3-5-3;
    - Traiettoria 5-cubica (o spline);
    - Traiettoria parabolico-lineare

3.4.1. Profilo trapezoidale di velocità

In realtà, nella pratica industriale è spesso utilizzato un particolare tipo di sequenza di polinomi raccordati chiamato ***profilo trapezoidale di velocità***. Questo è costituito da una prima fase di **accelerazione** seguita da una a **velocità costante** per poi concludere con una fase di **decelerazione**, il che porta ad una traiettoria costituita da un tratto lineare connesso da due tratti parabolici iniziale e finale: [3]

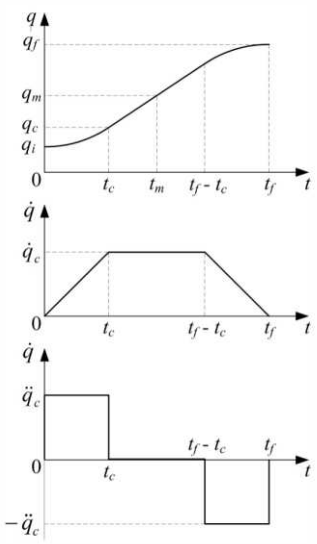


Figure 11 - Il profilo trapezoidale visto in termini di posizione (sopra), velocità (centro) e accelerazione (sotto)

Le velocità iniziale e finale sono richieste nulle e i tratti di accelerazione e decelerazione vengono imposti della stessa durata in modo da avere un profilo simmetrico rispetto al tempo t_m a cui è associata la posizione q_m:

$$t_m = \frac{t_f}{2} \qquad q_m = \frac{q_f + q_i}{2}$$

Mentre con t_c e q_c vengono indicati il tempo e la rispettiva posizione in cui si esaurisce il tratto di accelerazione.

Per ottenere un profilo effettivamente trapezoidale e non degenere bisogna rispettare la relazione:

$$t_c \leq t_m = \frac{t_f}{2}$$

Il che si ha quando:

$$\ddot{q}_c > \frac{4(q_f - q_i)}{t_f^2}$$

Dunque, fissati t_f, q_i e q_f, si sceglie un valore di accelerazione che soddisfi la relazione precedente, si calcola il t_c come:

$$t_c = \frac{t_f}{2} - \sqrt{\frac{t_f^2}{4} - \frac{q_f - q_i}{\ddot{q}_c}}$$

E si calcola la legge di moto per i tre tratti, tale che rispetti la seguente condizione:

$$0 \leq t < t_c \;\rightarrow\; q(t) = q_i + \frac{1}{2}\ddot{q}_c t^2$$

$$t_c \leq t < t_f - t_c \;\rightarrow\; q(t) = q_i + \ddot{q}_c t_c \left(t - \frac{t_c}{2}\right)$$

$$t_f - t_c < t \leq t_f \rightarrow q(t) = q_f - \frac{1}{2}\ddot{q}_c (t_f - t)^2$$

4. IMPLEMENTAZIONE ALGORITMI

In questo capitolo verrà mostrato come la matematica che regola il sistema appena studiata viene implementata a livello di codice Octave per poter ottenere i risultati prefissati. L'idea è quella di avere un *main code* molto pulito, in cui vengono richiamate le funzioni sviluppate esternamente. Dopo che l'utente ha inserito gli input nell'apposito script, basta lanciare il codice principale per ottenere subito i risultati. Come specificato anche nel capitolo precedente, lo sviluppo è stato graduale, a partire da un semplice cinematismo planare 2R per arrivare al modello completo. Durante queste fasi di sviluppo sono state via via migliorate alcune funzioni, ne sono state create di nuove e sono state modificate alcune logiche, cosa che fa capire come l'idea di procedere *step by step* è stata utile a migliorare di volta in volta l'implementazione ed ottenere alla fine un codice di qualità maggiore. Pertanto, per alcuni script, vengono riportate sia le versioni "obsolete", sia quelle migliorate durante le fasi successive dello sviluppo.

4.1. Main code

Come anticipato, il main segue per tutti e tre i casi studiati una struttura molto semplice, costituita da:

- Pulizia del *workspace* e della *command window*;
- *Run* dello script contenente le variabili di input, in modo da inserirle nel workspace;
- Richiamo della funzione che esegue l'interpolazione (nello spazio cartesiano o in quello dei giunti)

```
% Main cinematismo SCARA
clc;
clear;
close all;

% Caricamento input inseriti dall'utente
run Input_utente_SCARA.m;

% Richiamo la funzione che calcola le grandezze cinematiche e dinamiche nello spazio dei giunti
Interpolazione_spazio_cartesiano_SCARA(lung, massa_aste, massa_giunti,phi,phi_p,phi_pp)
```

Figure 12 - Viene riportato il main del calcolatore per lo SCARA come esempio

4.2. Input e generazione della legge di moto

Per quanto riguarda la definizione degli input e della legge di moto, questa è stata eseguita in modo analogo per i cinematismi 2R e 3R, mentre ha subito delle variazioni nel caso dello SCARA. Infatti, nei primi due casi è stato creato un unico script per l'immissione delle variabili relative alla geometria del manipolatore e quelle necessarie alla definizione della legge di moto, questo perché la funzione di interpolazione, come vedremo, contiene al suo interno la funzione per la creazione della legge di moto; per lo SCARA, invece, al fine di rendere il tool maggiormente *customizzabile*, l'idea è stata quella di generare la legge di moto in uno script a parte, il quale produce un file di testo contenente i valori di posizione, velocità e accelerazione, e di caricare questo file nella funzione di interpolazione. In questa maniera il tool funziona con qualsiasi legge di moto creata esternamente e non con una sola, interna ad esso.

```
% Script per la pianificazione della traiettoria da compiere, trattasi di una semplice
% traiettoria punto-punto nello spazio con profilo trapezoidale di velocità

clc
clear
close all

% Definizione punti iniziale e finale
p_i = [.36 .07 .05];  % Punto iniziale [m]
p_f = [.18 .25 .09];  % Punto finale [m]

% Definizione dell'accelerazione e del tempo di moto
acc = 0.1;
t_f = 5;
t = 0:0.01:t_f;

% Calcolo s_max
s_max = norm(p_f-p_i);

% Verifica caso degenere e calcolo del tempo di accelerazione
if (acc > (4*norm((p_f-p_i)))/t_f^2)
  t_acc = t_f/2 - sqrt((t_f^2)/4 - (norm(p_f-p_i))/acc);

  % Calcolo del profilo trapezoidale di velocità
  [s, s_p, s_pp] = prof_trap_vel (s_max, acc, t_acc, t);

  % Calcolo del versore t per un segmento
  vers_t = (p_f-p_i)/norm(p_f-p_i);

  % Calcolo di posizione, velocità e accelerazione nello spazio cartesiano
  pos = [];
  vel = [];
  accel =[];
  for i=1:length(s)
    pos(:,i) = p_i + vers_t*s(i);
    vel(:,i) = vers_t*s_p(i);
    accel(:,i) = vers_t*s_pp(i);
  endfor

  % Export dei valori come file di testo
  % [pos_x, pos_y, pos_z, vel_x, vel_y, vel_z, accel_x, accel_y, accel_z]
  dlmwrite('legge_moto_2.txt',[t;pos;vel;accel]','delimiter','\t')
else
  disp('L''accelerazione richiesta porta ad un caso degenere, aumentare l''accelerazione o il tempo di moto');
endif
```

Figure 13 - Codice per la generazione del file contenente la legge di moto

In tutti e tre i casi è stata scelta una traiettoria rettilinea, con profilo trapezoidale di velocità da un punto iniziale ad uno finale, la cui pianificazione è stata affrontata nel capitolo precedente. Tuttavia, a livello software, la funzione che permette tale pianificazione è stata quella che ha subito più modifiche. Nel caso dei cinematismi 2R e 3R l'idea è stata quella di creare una funzione che, dati in ingresso: il tempo di durata del moto, una lista degli istanti di tempo da 0 all'istante finale, l'accelerazione nei tratti di salita e discesa e le coordinate dei punti di inizio e fine traiettoria, fornisse posizioni, velocità e accelerazioni del terminale scomposte nelle coordinate X e Y (ricordiamo che sono due casi planari). Per ottenere ciò, all'interno della funzione è stato calcolato l'angolo di inclinazione della traiettoria rispetto all'asse delle ascisse, per poter scomporre il valore dell'accelerazione nelle due componenti; segue poi una lunga serie di *if statement* per definire il verso di tali componenti in base alla posizione dei punti iniziali e finale, per terminare con la definizione del tempo di accelerazione e finalmente con il calcolo di posizioni, velocità e accelerazioni così come visto nel paragrafo relativo alla pianificazione. Il codice così prodotto ha mostrato di funzionare bene e di generare la legge di moto voluta scomposta nelle due dimensioni del piano, tuttavia presenta le seguenti criticità:

- Il codice risulta lungo e difficilmente leggibile
- È difficile controllare che il valore di accelerazione imposto non porti ad un profilo degenere (triangolare)
- Problemi nel modificare questa funzione per utilizzarla nel caso tridimensionale.

Proprio quest'ultimo problema è stato determinante nella scelta di cambiare totalmente approccio per questa funzione. Infatti, nel momento in cui è stato necessario modificarla per il caso dello SCARA (non è più un cinematismo planare, ma tridimensionale) sono emerse grandi criticità, sia per poter scomporre l'accelerazione nelle tre dimensioni, sia per le casistiche in più da analizzare per definirne il verso. Inoltre, il codice risultava ancora più lungo, ancor meno leggibile e, a differenza del caso planare, non generava correttamente il profilo trapezoidale. Pertanto, la scelta è stata quella di modificare completamente la logica della funzione, ricorrendo alla parametrizzazione della traiettoria mediante ascissa curvilinea e assegnando la legge di moto ad essa, senza dover scomporre nulla, utilizzando direttamente lo scalare; una volta ottenuti posizione, velocità e accelerazione dell'ascissa curvilinea, è bastato moltiplicare tali valori per il versore tangente la traiettoria per ottenere i valori nelle tre dimensioni. Il codice così implementato presenta i seguenti vantaggi:

- Codice breve e molto leggibile;
- Maggior carattere di generalità, in quanto funzionante per qualunque traiettoria, a patto che sia parametrizzata mediante ascissa curvilinea;
- È facile controllare che non si giunga al caso degenere;
- Funziona in modo perfetto sia con casi bidimensionali che tridimensionali.

```
% Questa è una nuova versione per il calcolo del profilo trapezoidale di velocità;
% in questo caso viene sfruttata l'ascissa curvilinea in modo da aumentare il carattere
% di generalità della funzione (potrà essere usata su qualsiasi traiettoria parametrizzata
% mediante ascissa curvilinea)

function [s, s_p, s_pp] = prof_trap_vel (s_max, acc, t_acc, t)

  % Ricavo il tempo finale dalla lista dei tempi
  t_f = t(end);

  % Inizializzo i vettori
  s = [];     % spostamento
  s_p = [];   % velocità
  s_pp = [];  % accelerazione

  % Calcolo dei valori per i vari tratti
  for i=1:length(t)
      if (t(i)>=0) && (t(i)<=t_acc)  % primo tratto
        s(i) = 0.5*acc*(t(i)^2);
        s_p(i) = acc*t(i);
        s_pp(i) = acc;
      elseif (t(i)>t_acc) && (t(i)<=(t_f-t_acc))  % secondo tratto
        s(i) = acc*t_acc*(t(i)-(t_acc/2));
        s_p(i) = acc*t_acc;
        s_pp(i) = 0;
      elseif (t(i)>(t_f-t_acc)) && (t(i)<=t_f)  % terzo tratto
        s(i) = s_max - 0.5*acc*((t_f-t(i))^2);
        s_p(i) = acc*(t_f-t(i));
        s_pp(i) = -acc;
      endif
    endfor

endfunction
```

Figure 14 - Codice per la pianificazione del profilo trapezoidale di velocità sull'ascissa curvilinea

4.3. Controlli sugli input

Ogni qual volta vi è un'interazione *uomo-macchina* in un software, indipendente dalla modalità di input, sia essa tramite un'interfaccia grafica o tramite uno script, è buona norma assicurarsi il più possibile della validità degli input immessi. L'ambiente di sviluppo, in questo caso Octave, gestisce automaticamente errori di tipo sintattico, errori dovuti al tipo di dato immesso ecc. Tuttavia, non può gestire errori di altra natura, come ad esempio l'aver immesso un punto iniziale o finale che si

trovi al di fuori dello spazio di lavoro del robot. Per questo motivo sono state implementate, dove necessario, delle piccole routine di controllo che avvertono l'utente di questo tipo di errore, che può così essere facilmente corretto.

In particolare, essendo gli input del tool legati alla definizione della geometria del robot e del movimento che deve eseguire, i controlli principali che sono stati eseguiti sono relativi al fatto che la traiettoria sia interamente contenuta nello spazio operativo del robot (che dipende dalle geometria scelta) e che la legge di moto, di tipo trapezoidale, non degeneri in un profilo triangolare (dipende dall'accelerazione e dal tempo di moto richiesti).

Per quanto riguarda il controllo sullo spazio operativo, questo ha subito delle variazioni tra i casi 2R/3R e quello SCARA. Infatti, nei primi due casi, il controllo è stato implementato internamente alla funzione di *interpolazione*, all'inizio di essa, andando a verificare con due *if statement* se il punto iniziale e quello finale della traiettoria fossero compresi tra la massima estensione del braccio robotico e la minima estensione, mostrando, in caso negativo, un messaggio di errore che indica il punto che si trova al di fuori e terminando l'esecuzione del programma. Questo sistema, seppur funzioni bene, presenta alcuni svantaggi:

- Il controllo avviene solo sul punto iniziale e finale, il che potrebbe non essere sufficiente ad assicurare che l'intera traiettoria sia nello spazio di lavoro, soprattutto con traiettorie di natura non rettilinea;
- Righe di codice lunghe e poco leggibili, soprattutto nel momento in cui bisogna aggiungere, oltre che al controllo sul piano, anche quello relativo allo spostamento verticale;
- Routine che poco si adatta al controllo sulla legge di moto caricata come file .*txt*, come visto per il caso dello SCARA.

Questo ha portato, per lo SCARA, a implementare una funzione *ad hoc* che esegue il controllo di un generico punto rispetto all'area di lavoro, sia nel piano XY che lungo la coordinata Z, assegnando il valore 0 se il punto cade all'interno, 1 se è all'esterno. All'interno della funzione di interpolazione, invece, viene fatta girare questa funzione di controllo su tutti i punti della legge di moto caricata, e viene generato il messaggio di errore qualora anche uno solo di questi punti viene associato al valore 1, specificando se la traiettoria esce dall'area spazzata dal robot sul piano XY oppure se supera i

vincoli in altezza dati dalla geometria del manipolatore. Sono state così superate le criticità sorte nella versione precedente, in quanto:

- Il controllo avviene su tutti i punti della traiettoria;
- Il codice risulta più elegante, leggibile ed ottimizzato;
- Funziona per qualsiasi traiettoria nello spazio, inserita come file *.txt*

```
% Questa funzione serve ad effettuare un controllo sul moto richiesto al robot SCARA
% per verificar che questo sia interamente contenuto nello spazio di lavoro
function [check_altezza, check_piano] = controllo_spazio_operativo (cord, lung)

  % Controllo che la traiettoria sul piano xy sia interamente raggiungibile dal braccio del robot
  if norm([cord(1) cord(2)]) < lung(2)+lung(3) && norm([cord(1) cord(2)]) > abs(lung(2)-lung(3))
    check_piano = 0;
  else
    check_piano = 1;
  endif

  % Controllo che il moto verticale sia conforme alla geometria del robot
  if cord(3) > 0 && cord(3) < abs(lung(1)-lung(4))
    check_altezza = 0;
  else
    check_altezza = 1;
  endif

endfunction
```

Figure 15 - Funzione per il controllo sullo spazio operativo

Per quanto concerne la generazione dell'*alert* relativo alla degenerazione del profilo trapezoidale di velocità in profilo trapezoidale, questo avviene nella stessa maniera per tutti i casi, ovvero implementando la formula vista nella sezione dedicata al profilo trapezoidale di velocità (§ 3.4.1). Ciò che cambia è dove viene effettuato il controllo, nei casi 2R/3R avviene nella funzione che genera il profilo trapezoidale di velocità, mentre nel caso SCARA avviene nello script di pianificazione del moto, prima che venga richiamata la funzione per la genesi del profilo.

```
% Verifica caso degenere e calcolo del tempo di accelerazione
if (acc > (4*norm((p_f-p_i)))/t_f^2)
   t_acc = t_f/2 - sqrt((t_f^2)/4 - (norm(p_f-p_i))/acc);
```

Figure 16 - Dettaglio della condizione per evitare degenerazione del profilo trapezoidale

4.4. Interpolazione nello spazio cartesiano

Questa rappresenta la funzione principale del tool, all'interno della quale vengono effettuati tutti i calcoli relativi alla cinematica inversa del robot e della dinamica inversa, al fine di ottenere gli output descritti nel capitolo 2. La struttura della funzione, di cui ci apprestiamo a fare una descrizione è sostanzialmente uguale per tutti e tre i cinematismi visti, salvo le differenze già descritte e alcune modifiche date al diverso numero di link dei robot. Ciò che cambia sono, ovviamente, le sotto-funzioni relative al calcolo della cinematica, della matrice Jacobiana, e della dinamica, che analizzeremo in dettaglio. A scopo di esempio, si fa riferimento ora alla struttura del codice per lo SCARA, segnalando eventuali ulteriori differenze rispetto ai casi precedenti.

Per il codice di questa funzione si rimanda all'appendice in fondo.

La funzione vuole in ingresso solo i parametri relativi alla geometria del robot, in quanto la legge di moto viene calcolata esternamente e caricata all'interno della funzione, ovviamente nei casi 2R e 3R bisogna invece inserire anche i parametri necessari alla creazione della legge di moto, che avviene internamente.

Avviene il *load* della legge di moto in formato *.txt* la quale contiene dieci colonne: nella prima vi è la lista dei tempi, nelle colonne dalla 2 alla 4 vi è la posizione istantanea del terminale nelle tre componenti X,Y, e Z, nelle colonne dalla 5 alla 7 vi sono le velocità e infine, dalla 8 alla 10 abbiamo le accelerazioni.

Segue poi la routine di controllo sullo spazio operativo già decritta e l'estrazione dei vettori relativi ai tempi, alle posizioni, alle velocità e alle accelerazioni dal file contenente la legge di moto.

Viene usata la lunghezza del vettore dei tempi per definire il numero di intervalli di interpolazione e vengono inizializzate le liste (vuote) in cui verranno inserite le grandezze relative ai giunti, calcolate iterativamente dall'algoritmo; viene inoltre inizializzata la posizione attuale del terminale.

Tramite un ciclo *for* si itera tra 0 e il numero di intervalli e si divide ulteriormente il computo con un *if* per separare il calcolo del valore finale, per evitare problemi di indicizzazione.

Ha inizio a questo punto il calcolo vero e proprio.

Tramite la funzione che calcola la cinematica inversa vengono calcolate le due configurazioni dei giunti relative alla posizione attuale del terminale, tra le due viene scelta la configurazione1 e vengono inseriti i valori nei vettori posizione inizializzati in precedenza.

Avendo a disposizione le posizioni dei giunti e la velocità del terminale, viene calcolata la velocità dei giunti tramite la formula per il calcolo delle velocità di giunto vista (vedere § 3.2.2) e per farlo viene richiamata la funzione che calcola la matrice Jacobiana del cinematismo, che viene invertita in quanto si tratta di un'analisi di velocità inversa. I valori ottenuti vengono inseriti nelle relative liste.

Per il calcolo delle accelerazioni di giunto si è fatto riferimento alla formula vista nell'analisi di accelerazione (vedere § 3.2.2) e dunque è stato necessario richiamare non solo la funzione Jacobiana, ma anche la sua derivata; inoltre, sono state sfruttate le velocità di giunto appena calcolate e le accelerazioni del terminale, al termine del computo sono stati inseriti i valori nei rispettivi vettori.

Giungiamo a questo punto al computo delle coppie richieste ai giunti, che avviene per mezzo della funzione che implementa la dinamica inversa del robot; è importante notare che questa funzione necessita di tutti i valori calcolati finora, in accordo con il fatto che per eseguire l'analisi dinamica inversa bisogna prima aver eseguito quella cinematica.

Sono stati calcolati tutti i valori desiderati, tuttavia, per la visualizzazione grafica abbiamo la necessità di calcolare anche la posizione (cartesiana) del secondo giunto in ogni istante di tempo, in quanto tale punto è importante ai fini del plot.

Infine, abbiamo l'aggiornamento della posizione attuale del terminale con cui verranno calcolati tutti i valori appena visti per l'istante di tempo successivo, e l'algoritmo procede iterativamente per tutti gli istanti di tempo del moto.

Terminato il ciclo, tutte le grandezze relative ai giunti sono nelle apposite liste e sono pronte per il plot.

Per il plot, viene creata un'unica figura, nella quale vengono visualizzati tutti i *subplot* di interesse; viene specificato il nome della figura, viene settata so *off* la numerazione della figura in quanto ne abbiamo solo una e vengono settati dei valori per definire la posizione in cui apparirà la figura e la sua dimensione.

Il primo subplot, quello relativo alla visualizzazione grafica del movimento del robot, è l'unico differente, in quanto si tratta di un plot tridimensionale, ottenuto mediante la funzione *plot3*; come parametri in ingresso vengono forniti tre vettori contenenti i valori X, Y e Z dei vari punti che vogliamo plottare, e quello che si ottiene è una curva che unisce questi punti. Ovviamente è presente un *for* per plottare nello stesso grafico tutte le configurazioni assunte dal robot. Sono state evidenziate in verde la configurazione iniziale e in magenta quella finale, mentre le configurazioni intermedie sono in rosso, con una linea più sottile.

Seguono i tre subplot relativi a velocità, accelerazioni e coppie, i cui plot sono semplici plot bidimensionali con il tempo sulle ascisse e la grandezza di interesse sulle ordinate. Si noti che sono state convertite le grandezze angolari da radianti a gradi, e quelle lineari da metri a millimetri, questo come vedremo, serve ad avere coerenza dimensionale con i risultati delle simulazioni descritte nel capitolo successivo.

Tutti i plot sono corredati dal proprio titolo, dalle proprie *label* per gli assi e dalle legende.

Verranno adesso descritte in dettaglio le sotto-funzioni che vengono richiamate nella funzione di interpolazione.

4.4.1. Cinematica inversa

La prima funzione che compare è quella che implementa la cinematica inversa, ovvero che calcola la posizione dei vari giunti in funzione della posizione richiesta al terminale e della geometria del manipolatore. Viene riportata la versione finale, quella relativa allo scara, in quanto presenta al suo interno anche le versioni precedenti: lo SCARA, come visto, può essere considerato, se proiettato sul piano XY, come un robot 2R, da cui eredita la cinematica inversa per il calcolo dei giunti 1 e 2; dal 3R invece eredita il calcolo della posizione del giunto 3, una volta fissato l'orientamento desiderato del terminale; infine è presente il calcolo della posizione del giunto 4, ovvero quello prismatico.

La funzione richiede come input il vettore delle lunghezze dei link del braccio robotico, la posizione del terminale (espressa in coordinate cartesiane) e il suo orientamento rispetto all'asse X espresso in gradi; l'output della funzione sono le due possibili configurazioni che può assumere il cinematismo per la data posizione del terminale, anche questa è un'eredità dell'inversione cinematica del 2R

planare. Entrambe le configurazioni contengono i valori degli angoli assunti dai primi tre giunti e il valore lineare assunto dal giunto prismatico.

```
% Questa funzione implementa la cinematica inversa di un robot scara, prende in
% ingresso la geometria del robot, la posizione dell'end effector nello spazio
% cartesiano e il suo orientamento e restituisce le due configurazioni possibili
% dei giunti per ottenere tale posizione

function [config1, config2] = cin_inv_scara (lung, cord, phi)

  phi = deg2rad(phi);

  cosTheta2 = (cord(1)^2 + cord(2)^2 - lung(2)^2 - lung(3)^2) / (2*lung(2)*lung(3));

  sinTheta2_1 = sqrt(1-cosTheta2^2);
  sinTheta2_2 = -sinTheta2_1;

  config1(2) = atan2(sinTheta2_1, cosTheta2);
  config1(1) = atan2(cord(2), cord(1)) - atan2(lung(3)*sinTheta2_1, lung(2)+lung(3)*cosTheta2);
  config1(3) = phi - config1(2) - config1(1);
  config1(4) = lung(1) - cord(3) - lung(4);

  config2(2) = atan2(sinTheta2_2, cosTheta2);
  config2(1) = atan2(cord(2), cord(1)) - atan2(lung(3)*sinTheta2_2, lung(2)+lung(3)*cosTheta2);
  config1(3) = phi - config1(2) - config1(1);
  config1(4) = lung(1) - cord(3) - lung(4);

endfunction
```

Figure 17 - La funzione che implementa la cinematica inversa dello SCARA

4.4.2. Jacobiana

Vediamo ora come è stata implementata la matrice Jacobiana, il cui calcolo, a livello matematico è stato già affrontato nel paragrafo 3.2.2. La funzione, che ovviamente è diversa per i tre casi, in quanto la dimensione della matrice Jacobiana dipende dal numero di g.d.l. del sistema, non fa altro che assemblare la matrice, calcolando tutti gli elementi di essa a partire dal vettore delle lunghezze delle aste e dal vettore contenente la configurazione (una delle due) ottenuta dalla cinematica inversa.

Per quanto riguarda la Jacobiana derivata, è stata assemblata in modo analogo, ma derivando i vari elementi, motivo per cui sono necessarie in input, oltre che le posizioni di giunto, anche le velocità.

```
% Questa funzione calcola la matrice Jacobiana per un robot SCARA

function J = Jacobiana_scara (lung, config)

  J(1,1) = -lung(2)*sin(config(1)) - lung(3)*sin(config(1)+config(2));
  J(1,2) = -lung(3)*sin(config(1)+config(2));
  J(1,3) = 0;
  J(1,4) = 0;
  J(2,1) = lung(2)*cos(config(1)) + lung(3)*cos(config(1)+config(2));
  J(2,2) = lung(3)*cos(config(1)+config(2));
  J(2,3) = 0;
  J(2,4) = 0;
  J(3,1) = 0;
  J(3,2) = 0;
  J(3,3) = 0;
  J(3,4) = 1;
  J(4,1) = 1;
  J(4,2) = 1;
  J(4,3) = 1;
  J(4,4) = 0;

endfunction
```

Figure 18 - implementazione della Jacobiana dello SCARA

4.4.3. Dinamica inversa

La quarta ed ultima funzione che compare è quella per il calcolo delle coppie dei giunti, ottenute attraverso la dinamica inversa. Per quanto visto nel capitolo relativo allo studio della dinamica del sistema, questa funzione necessita di alcune matrici, assemblate con opportune funzioni, in modo analogo a quanto visto per l'assemblaggio della matrice Jacobiana; stiamo parlando di:

- Matrice di massa;
- Matrice di Coriolis;
- Matrice di gravità.

Per quanto riguarda le differenze tra le varie configurazioni, ciò che cambia tra di esse è la dimensione di tali matrici, per via del fatto che abbiamo gradi di libertà differenti; in particolare, per lo scara completo (4 g.d.l.), abbiamo che la matrice di massa e quella di Coriolis sono di dimensione 4x4, inoltre risulta essere l'unico caso in cui compare la matrice di gravità (1x4), in quanto le configurazioni 2R e 3R sono cinematismi planari che non risentono di essa. È importante notare che per poter implementare queste matrici e la funzione di dinamica inversa che le contiene è fondamentale aver già effettuato l'analisi cinematica.

```
% Questa funzione implementa la dinamica inversa per un robot SCARA, prende come input
% la geometria del robot, le sue grandezze fisiche e le sue grandezze cinematiche e
% restituisce i valori di coppia richiesti ai giunti per ottenere la legge di moto
% desiderata

function tau = Dinamica_inversa_SCARA (lung, massa_aste, massa_giunti, theta, vel_giunti, acc_giunti)
    tau = matrice_massa_SCARA(theta, lung, massa_aste, massa_giunti)*acc_giunti + matrice_coriolis_SCARA(theta, vel_giunti, lung, massa_aste)*vel_giunti + matrice_gravita_SCARA(massa_aste);

endfunction
```

Figure 19 - Funzione per la dinamica inversa dello SCARA

4.5. Output

Finita la fase di implementazione, sono stati testati tutti e tre gli algoritmi, per farlo sono stati scelti arbitrariamente gli input e lanciato il calcolatore, ottenendo gli output desiderati. Vediamo adesso in dettaglio tutti e tre i modelli.

- **Robot 2R**:

  Per il robot 2R sono stati scelti i seguenti parametri:

  - Lunghezza aste: 200 mm
  - Massa aste: 5 kg (questo valore è stato poi modificato in fase di verifica dei risultati con la massa effettiva delle aste modellate su creo)
  - Massa giunti: 1 kg

  Come parametri del moto sono stati scelti:

  - Punto iniziale: (300 mm, 70 mm)
  - Punto finale: (120 mm, 180 mm)
  - Tempo: 10 s
  - Accelerazione: 10 mm/s^2

  Di seguito viene riportato l'output generato del software:

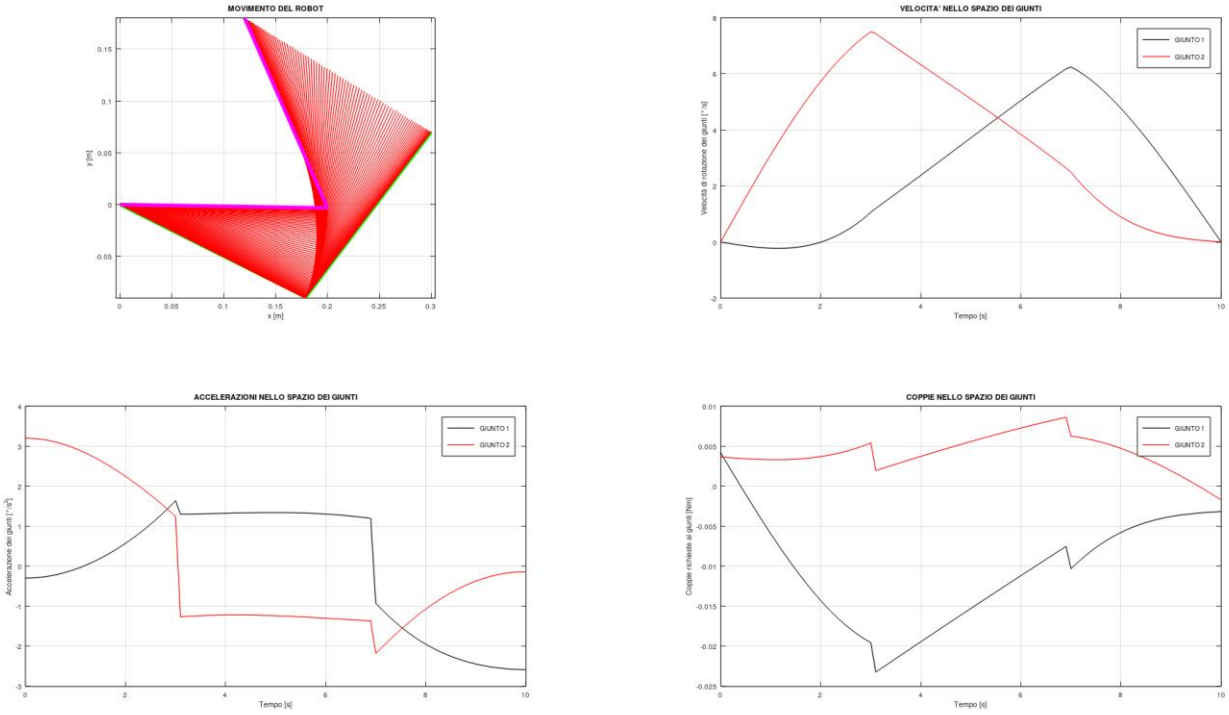


Figure 20 - Output del calcolatore per il robot 2R planare

In un tempo computazionale di pochi secondi, si ottiene un risultato grafico, dunque di immediato impatto, circa il moto del robot. Nel quadrante in alto a sinistra abbiamo la rappresentazione stilizzata del cinematismo, in cui si possono vedere:

- In **blu tratteggiato**: la traiettoria compiuta dal robot, la quale è una traiettoria rettilinea così come era stata pianificata;
- In **rosso continuo**: le configurazioni intermedie assunte dal robot durante il moto, queste sono utili per due motivi; il primo per poter considerare l'area spazzata dal robot durante il movimento e dunque prevedere eventuali collisioni con oggetti esterni, il secondo motivo è che l'infittimento di queste linee nei pressi dei punti iniziale e finale ci fanno capire che quelli sono i tratti di accelerazione e decelerazione imposti con il profilo trapezoidale, dunque ci danno un riscontro sulla bontà della pianificazione della traiettoria.

Nel quadrante in altro a destra, invece, possiamo notare l'andamento della velocità angolare dei giunti, espressa in [°/s].

In basso abbiamo, a destra l'accelerazione angolare dei giunti, espressa in [°/s^2] e a destra i valori di coppia, in [Nm].

Si può notare come il profilo trapezoidale di velocità si ripercuota sui giunti in quanto abbiamo che la velocità si mantiene lineare nel tratto intermedio, e l'accelerazione è costante, mentre la coppia, non annullandosi l'accelerazione a livello dei giunti, ha anch'essa valore non nullo.

- **Robot 3R**:

  Per il robot 3R sono stati scelti i seguenti parametri:

  - Lunghezza aste: 200 mm
  - Orientamento ultima asta: 30° rispetto asse delle ascisse
  - Massa aste: 5 kg (questo valore è stato poi modificato in fase di verifica dei risultati con la massa effettiva delle aste modellate su creo)
  - Massa giunti: 1 kg

  Come parametri del moto sono stati scelti:

  - Punto iniziale: (500 mm, 150 mm)
  - Punto finale: (280 mm, 370 mm)
  - Tempo: 10 s
  - Accelerazione: 20 mm/s^2

  Di seguito viene riportato l'output generato del software:

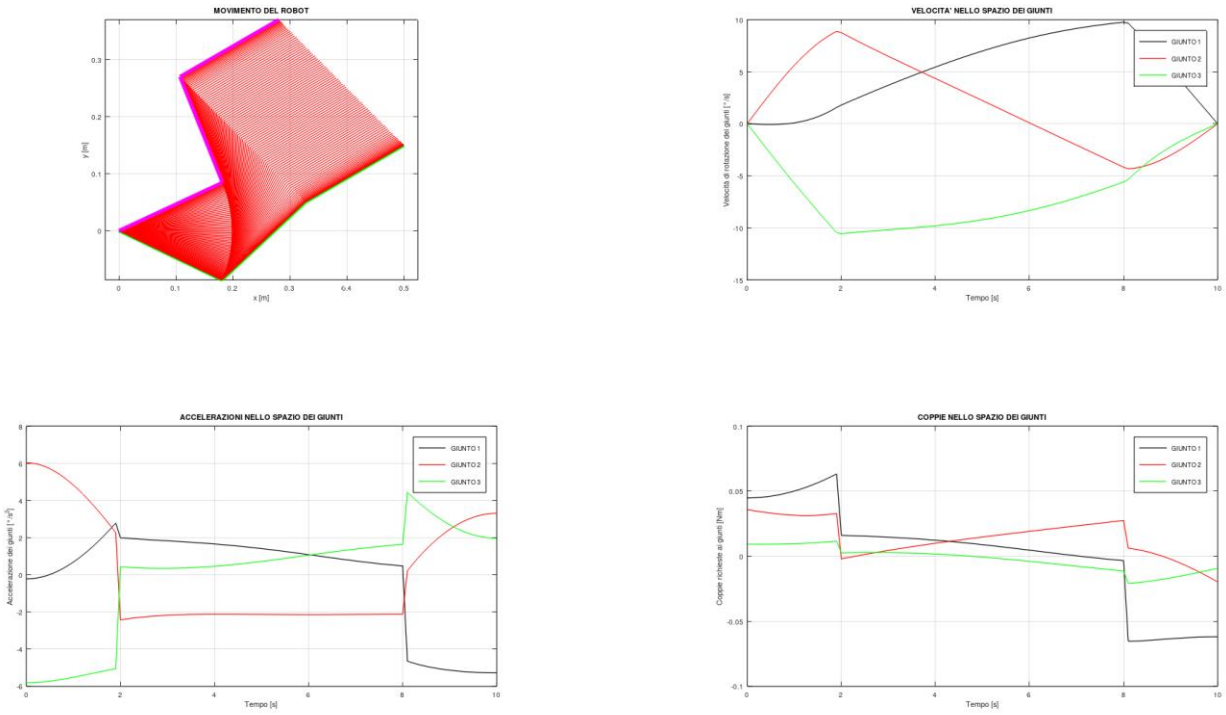


Figure 21 - Output del calcolatore per il robot 3R planare

Anche in questo caso l'interfaccia di output è la stessa, notiamo la presenza del terzo braccio nel quadrante dedicato alla rappresentazione del moto del robot, ma le considerazioni che si possono fare sono del tutto analoghe a quelle viste per il 2R. Possiamo aggiungere che l'orientamento del terzo braccio resta costante ad un angolo di 30° prefissato, così come imposto; questa è stata una scelta, ma è possibile far variare quest'angolo durante il moto semplicemente variando l'input e inserendo un range di angoli al posto di un angolo costante. Per quanto riguarda velocità, accelerazioni e coppie possiamo fare le stesse considerazioni del 2R, con la semplice aggiunta di un terzo andamento che è quello relativo al terzo giunto.

- **Robot SCARA**:

  Per il robot SCARA sono stati scelti i seguenti parametri:

  - Lunghezza aste: 250 mm per le prime due e 40 mm per l'ultima
  - Orientamento ultima asta: 30° rispetto asse delle ascisse
  - Massa aste: 7 kg per le prime due aste e 0.5 kg per l'ultima (questi valori sono stati poi modificati in fase di verifica dei risultati con la massa effettiva delle aste modellate su creo)

- Massa giunti: 1 kg

Come parametri del moto sono stati scelti:

- Punto iniziale: (360 mm, 70 mm, 50 mm)
- Punto finale: (180 mm, 250 mm, 90 mm)
- Tempo: 5 s
- Accelerazione: 100 mm/s^2

Di seguito viene riportato l'output generato del software:

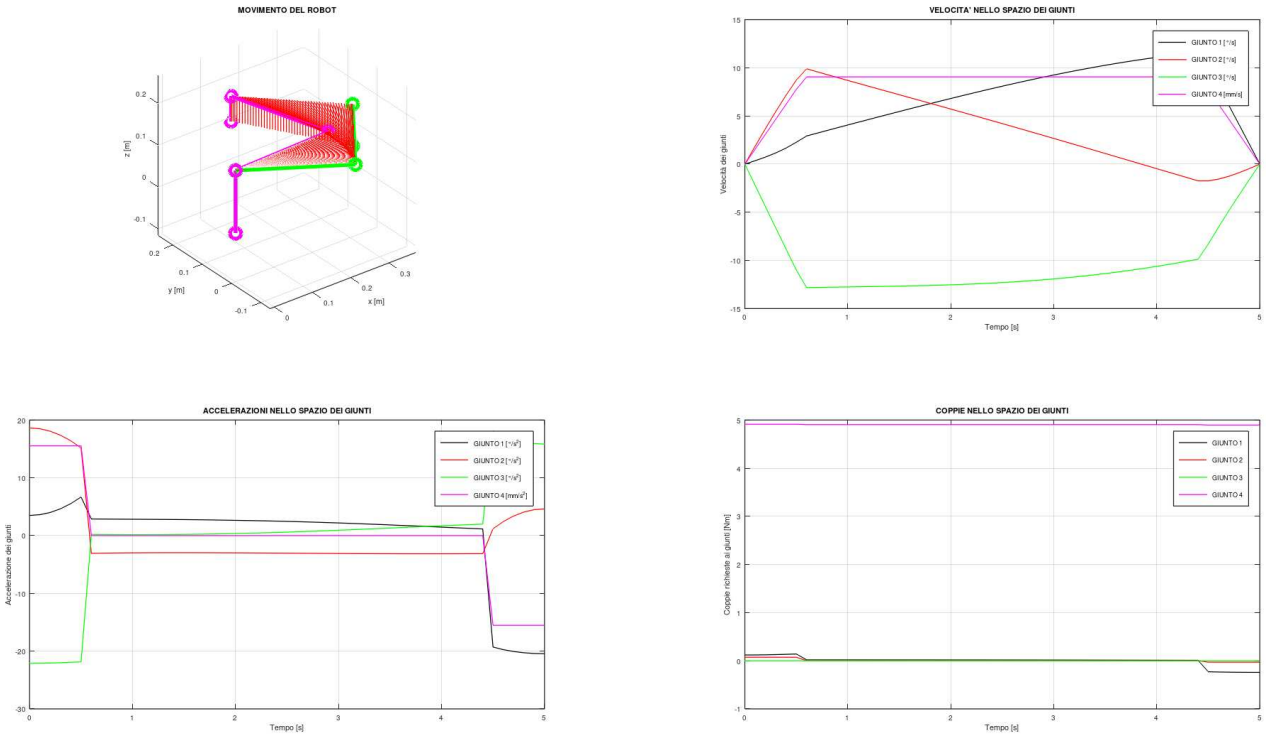


Figure 22 - Output del calcolatore per il robot SCARA

In questo caso, ci sono delle differenze sostanziali rispetto ai precedenti. Il plot in alto a destra mostra, infatti, come in questo caso ci troviamo nello spazio e non più in una condizione di moto puramente planare. Possiamo vedere come la traiettoria sia sempre rettilinea, ma avvenga nello spazio, con i primi due link che spazzano un'area planare, come il robot 2R e il terzo link che invece si muove solo verticalmente. Sono state evidenziate le configurazioni

iniziale (in verde) e finale (in magenta). Per quanto riguarda velocità, accelerazioni e coppie notiamo la presenza di quattro andamenti, questo perché per l'ultimo giunto è stata considerata sia la rotazione, sia la traslazione. Si noti che, per quanto riguarda la traslazione, essendo questa coincidente con l'effettiva traslazione del terminale, l'andamento è esattamente quello di un profilo trapezoidale. Inoltre, si può vedere come la coppia sia sensibilmente maggiore sul giunto prismatico, in virtù del fatto che questo è l'unico giunto in cui, a livello dinamico, agisce la forza di gravità.

5. VERIFICA DEI RISULTATI

Per quanto sia stato pensato, studiato e progettato con rigore, è impensabile utilizzare lo strumento di calcolo implementato senza che prima sia stata svolta una altrettanto accurata fase di verifica dei risultati ottenuti. Esistono vari metodi per verificare la correttezza delle equazioni implementate, in questo lavoro l'idea è stata quella di confrontarne i risultati con un secondo strumento che si sa essere affidabile e se questi combaciano a meno di uno scostamento trascurabile, si può considerare valido il tool e potranno essere considerati attendibili anche risultati ottenuti modificando i parametri. Sono state eseguite tutte le verifiche dal punto di vista cinematico; per quanto riguarda la dinamica, questa sarà oggetto di approfondimenti futuri. Il tool serve infatti a dare delle prime stime di massima circa le coppie in gioco, per fare delle considerazioni preliminari alla progettazione. Per il computo esatto si preferisce al momento utilizzare altri strumenti, a progettazione iniziata, in quanto le effettive geometrie, masse ed inerzie, non approssimabili con i casi ideali e dunque non computabili con il modello hanno un ruolo impattante.

5.1. Simulazione cinematica Creo

Per verificare le grandezze cinematiche del meccanismo è stato utilizzato Creo parametric, con il quale sono state effettuate tre simulazioni, una per ogni configurazione del cinematismo, utilizzando ovviamente gli stessi parametri utilizzati in Octave. Terminata la simulazione, sono state ricavate le misure delle grandezze cinematiche dei vari giunti e sono state confrontate con i risultati ottenuti con lo strumento di calcolo.

5.1.1. Preparazione del cinematismo

Anzitutto è stato modellato il cinematismo. Trattandosi di dover eseguire un'analisi cinematica, è possibile creare un modello altamente semplificato, costituito da semplici link opportunamente vincolati, di cui interessa solo la lunghezza e il movimento relativo. Dunque, l'unico componente modellato è l'asta, ottenuta come semplice estrusione di forma cilindrica. È stato poi creato un assieme nel quale, in base alla configurazione, sono state vincolate due, tre o quattro aste, in particolare:

- **Cinematismo 2R:** per garantire la confrontabilità dei risultati, sono state vincolate due aste di lunghezza 200 mm, utilizzando la stessa lunghezza utilizzata su Octave; la prima è stata

vincolata con il vincolo cinematico *pin* (che permette la sola rotazione attorno agli assi allineati) tra l'asse passante per il diametro della faccia planare dell'asta e l'asse z di riferimento, mentre per vincolare la seconda in modo che il terminale si muovesse analogamente ad Octave, è stato tracciato un segmento i cui punti iniziale e finale coincidono con quelli specificati dall'utente su Octave e sono stati inseriti due vincoli, uno di tipo *cylinder* tra i due assi diametrali delle facce planari delle due aste e uno di tipo *slot* tra il punto centrale dell'altra faccia e il segmento. In questa maniera abbiamo che il cinematismo risulta vincolato in modo che il terminale scorra sul segmento e le due aste si muovano nel piano come desiderato.

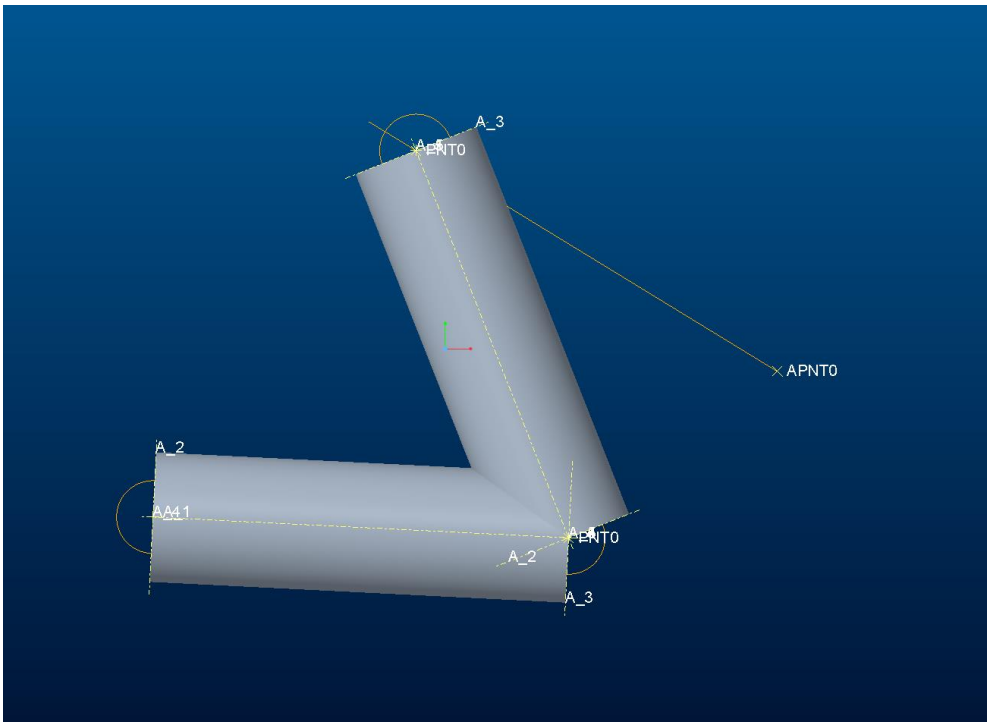


Figure 23 - Il cinematismo 2R modellato con Creo Parametric

- **Cinematismo 3R:** per questo cinematismo le aste vincolate sono state tre, anche in questo caso tutte e tre della medesima lunghezza di 200 mm. Per quanto riguarda i vincoli cinematici, questi sono simili a quelli visti per il precedente, ma questa volta l'ultima asta deve muoversi con orientamento fissato rispetto all'asse X. Per ottenere ciò, sono stati tracciati due segmenti paralleli in modo che sul più esterno potesse scorrere il terminale, dal punto iniziale a quello finale richiesti dall'utente, mentre il segmento più interno fungesse da

guida per l'altro estremo dell'asta, la distanza tra i due segmenti determina l'inclinazione dell'asta. Inoltre, l'ultima asta è anche connessa alle prime due tramite vincolo *cylinder.* Anche in questo caso abbiamo che l'unico movimento possibile è proprio quello desiderato.

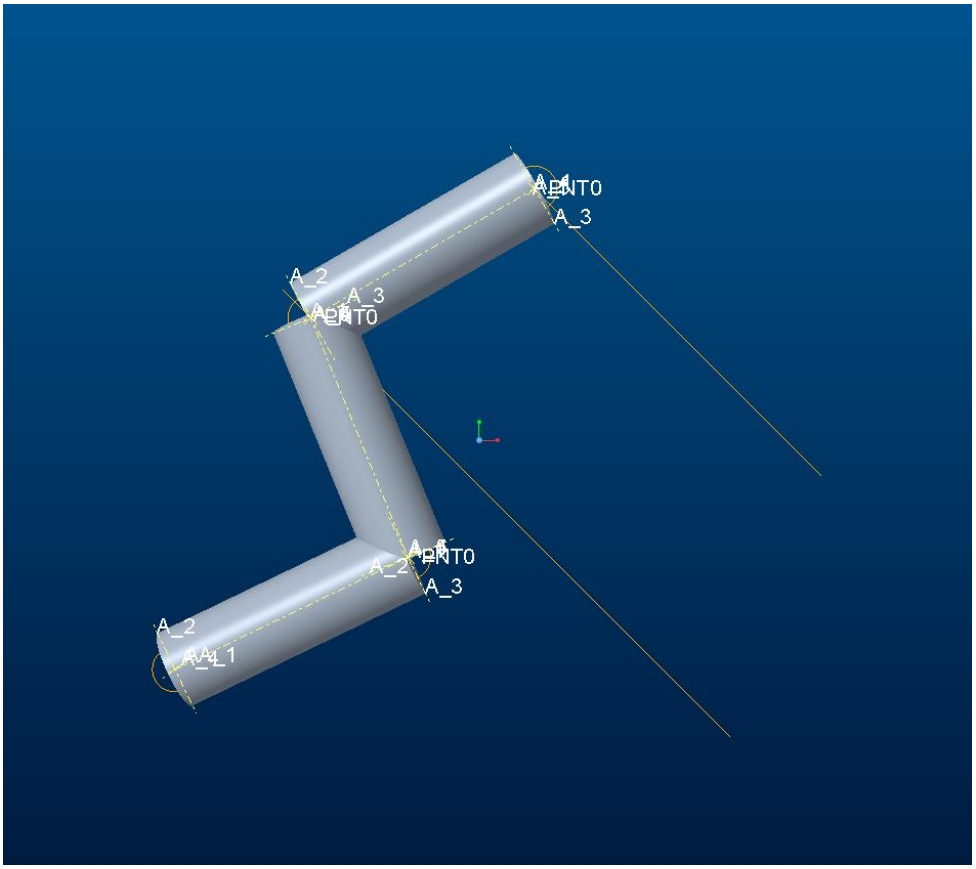


Figure 24 - Il cinematismo 3R modellato con Creo Parametric

- **Cinematismo SCARA:** Per il terzo ed ultimo cinematismo è stato ripreso il cinematismo 3R ma è stata aggiunta una quarta asta più piccola, di lunghezza 50 mm, vincolata sul terzo giunto con un vincolo *cylinder* ma usando in questo caso l'asse longitudinale, così che questa possa scorrere verticalmente, rappresentando di fatto il terminale dello SCARA. È stato mantenuto comunque il terzo braccio per essere più agevole la misura della posizione angolare.

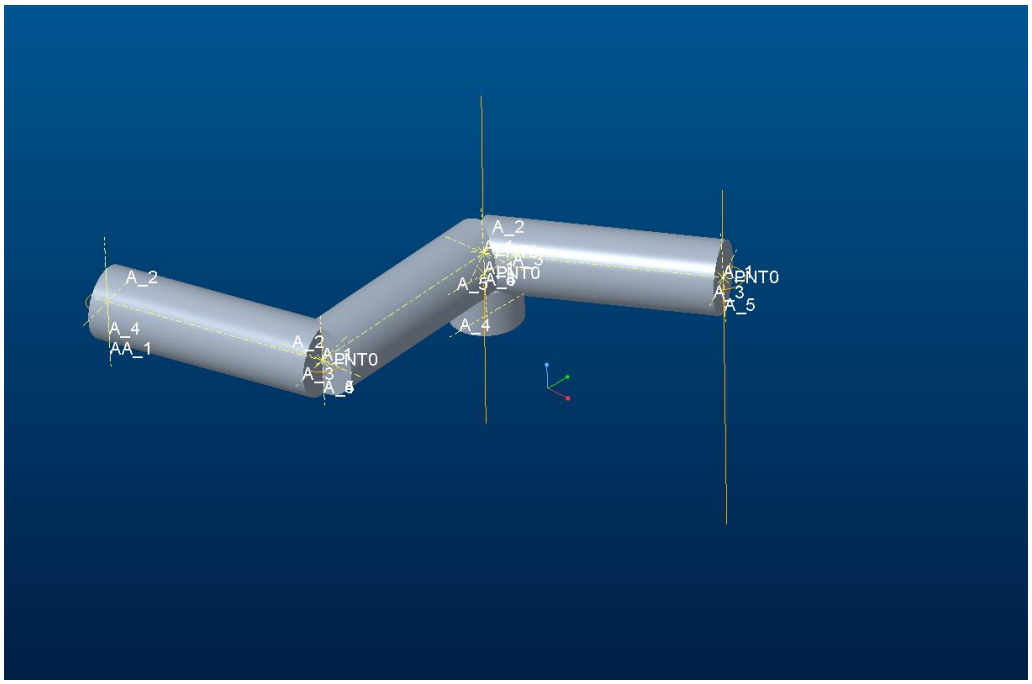


Figure 25 - Il cinematismo dello SCARA modellato con Creo Parametric

5.1.2. Preparazione della simulazione

Il secondo passo consiste nel fornire al cinematismo i motori per poter effettuare la simulazione. A tal fine si ricorre al modulo *mechanism* di Creo parametric, con cui è possibile assegnare arbitrariamente dei motori ai vincoli cinematici precedentemente definiti e specificare se si vuole pilotare tali motori specificando le posizioni, le velocità o le accelerazioni, le quali possono essere costanti oppure fornite tramite delle *table* specificate dall'utente. Nel nostro caso si è scelta quest'ultima opzione; l'idea, infatti, è stata quella di estrarre da Octave le accelerazioni del terminale e usarle come motore per Creo, per poi andare a misurare le posizioni, le velocità e le accelerazioni dei giunti. Per fare ciò è stato creato uno script Octave per estrarre le accelerazioni dalla legge di moto e salvarle in un file di testo da passare a Creo. Per i sistemi 2R e 3R il motore è stato inserito sul vincolo *slot* sul terminale; per lo SCARA, invece, sono stati inseriti due motori, uno sullo slot relativo al terzo giunto, l'altro sul vincolo cilindrico, in modo da controllare sia il moto sul piano che quello verticale. L'ultima parte relativa alla preparazione della simulazione è quella in cui si definiscono il tipo di analisi e suoi parametri; in particolare, è stata settata un'analisi di tipo *kinematic* e sono stati inseriti il tempo di simulazione e il numero di intervalli di interpolazione.

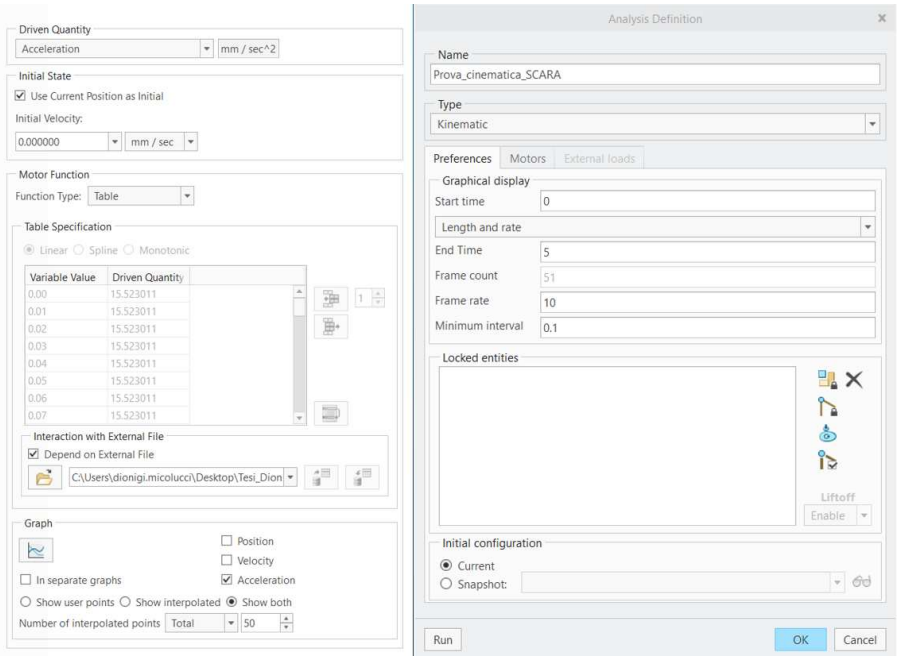


Figure 26 - Settaggio dei motori e della simulazione su Creo mechanism

A questo punto si ha tutto quello che serve per lanciare la simulazione, che dura pochi istanti.

5.1.3. Misura delle grandezze cinematiche di giunto

Tramite il comando *measures* si accede all'interfaccia che permette di definire quali grandezze dovranno essere oggetto di misura e su quali vincoli. Nel caso in esame sono state misurate posizioni, velocità e accelerazioni relative a tutti i giunti del robot e le stesse grandezze sono state misurate anche *sull'end effector* per verificare che la legge di moto sia effettivamente quella desiderata. Definite le misurazioni da effettuare, si carica il risultato della simulazione ed è possibile visualizzare l'andamento delle grandezze scelte nel tempo di simulazione, andamento che può essere esportato sottoforma di file Excel.

5.2. Confronto con Octave tramite Excel

Ottenuti i risultati della simulazione sottoforma di file *Excel*, è stato possibile procedere al confronto di tali risultati, che saranno presi come riferimento attendibile, con quelli ottenuti grazie al calcolatore, i quali saranno considerati validi nel momento in cui rispecchieranno fedelmente i dati della simulazione.

Passiamo ora in rassegna i risultati emersi da questo confronto:

- **Robot 2R**:

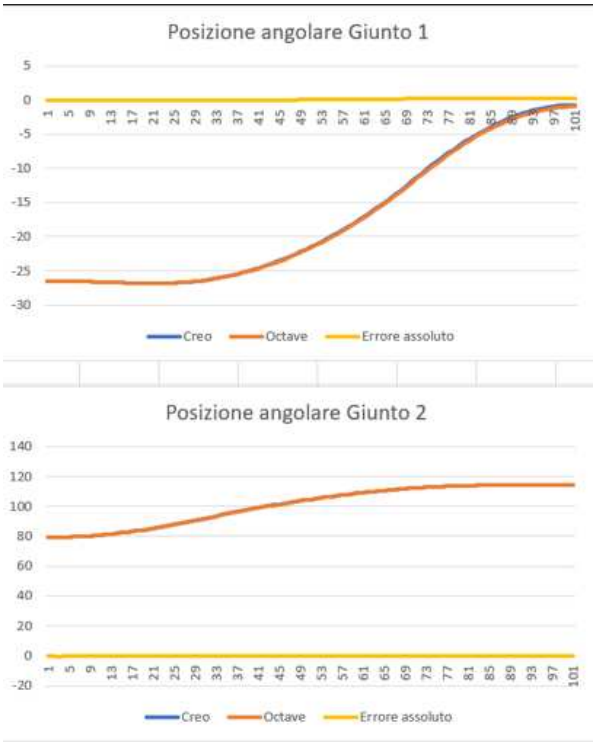


Figure 27 - Confronto delle posizioni di giunto calcolate e simulate per il robot 2R

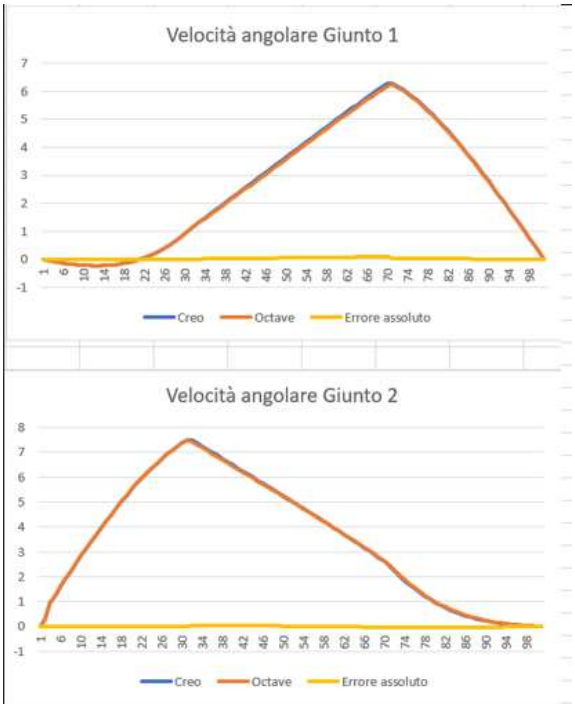


Figure 28 - Confronto delle velocità di giunto calcolate e simulate per il robot 2R

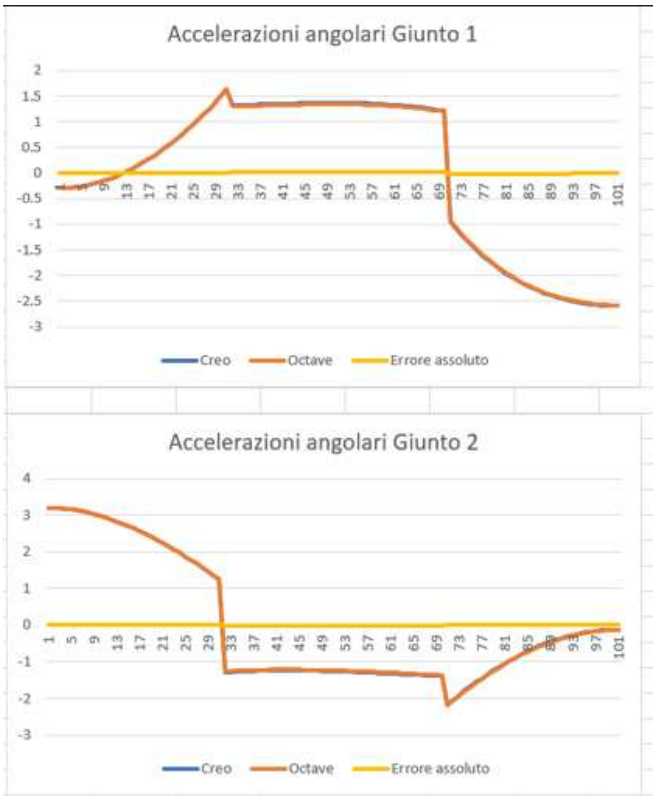


Figure 29 - Confronto delle accelerazioni di giunto calcolate e simulate per il robot 2R

Tramite Excel, oltre ad essere stati ineriti nello stesso grafico i valori ottenuti dal calcolatore e quelli ottenuti con la simulazione, sono state calcolate anche le differenze e plottate in termini di errore assoluto. Già a livello grafico si può notare come i risultati siano molto buoni, con una sovrapposizione quasi perfetta delle curve ed un andamento della curva di errore che risulta costantemente a valori prossimi allo 0. Se andiamo a vedere gli effettivi valori di scostamento dei risultati abbiamo che il massimo di tali valori per ciascuna grandezza esaminata sono:

- 0,26° per le posizioni;
- 0,09°/s per le velocità;
- 0,02°/s^2 per le accelerazioni.

Per quanto lo scostamento sia piccolo, in termini cinematici ci si aspetta che non vi sia alcuna differenza, la motivazione di questi errori è apparsa con maggior evidenza nelle verifiche del robot SCARA come è possibile notare più avanti.

- **Robot 3R**:

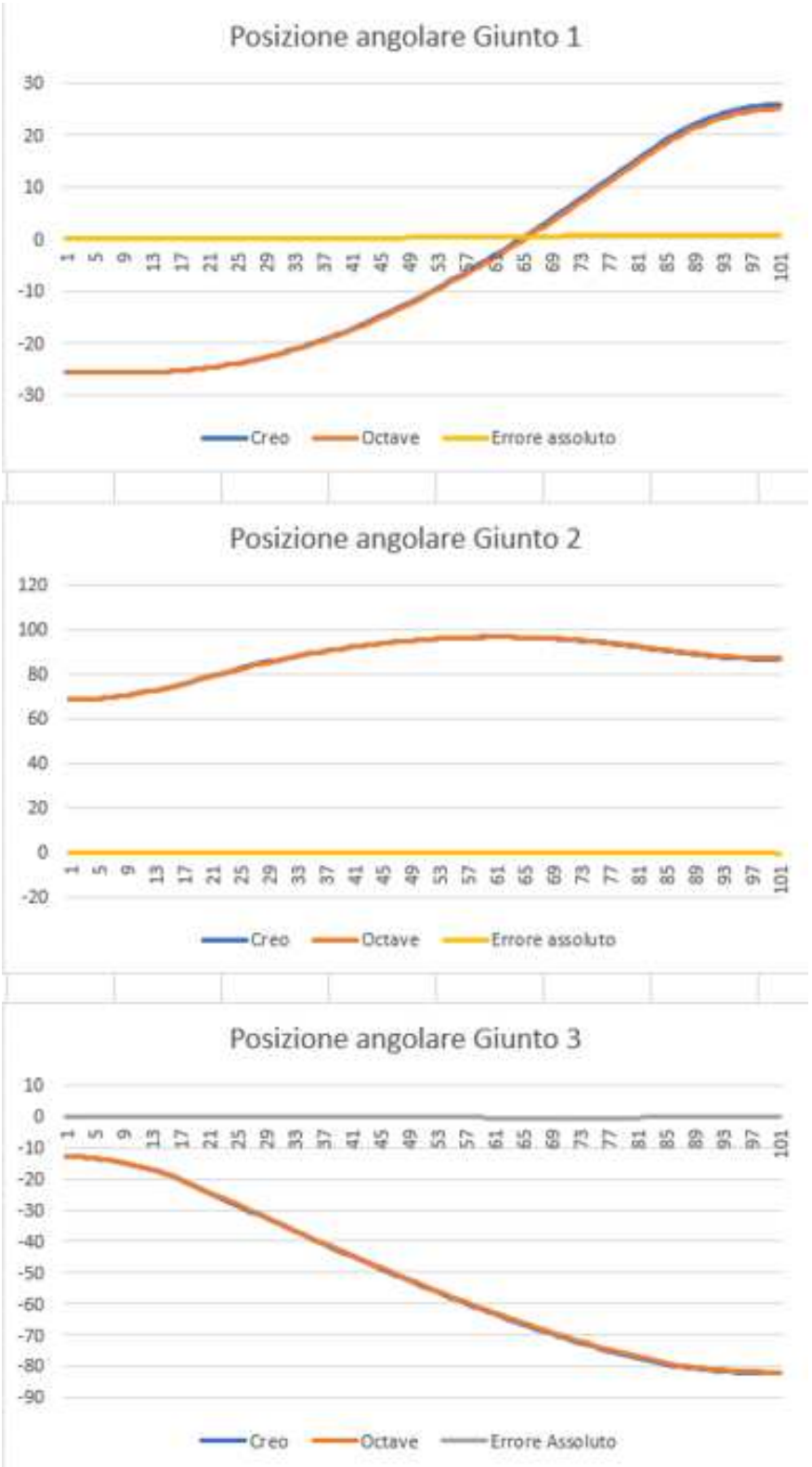


Figure 30 - Confronto delle posizioni di giunto calcolate e simulate per il robot 3R

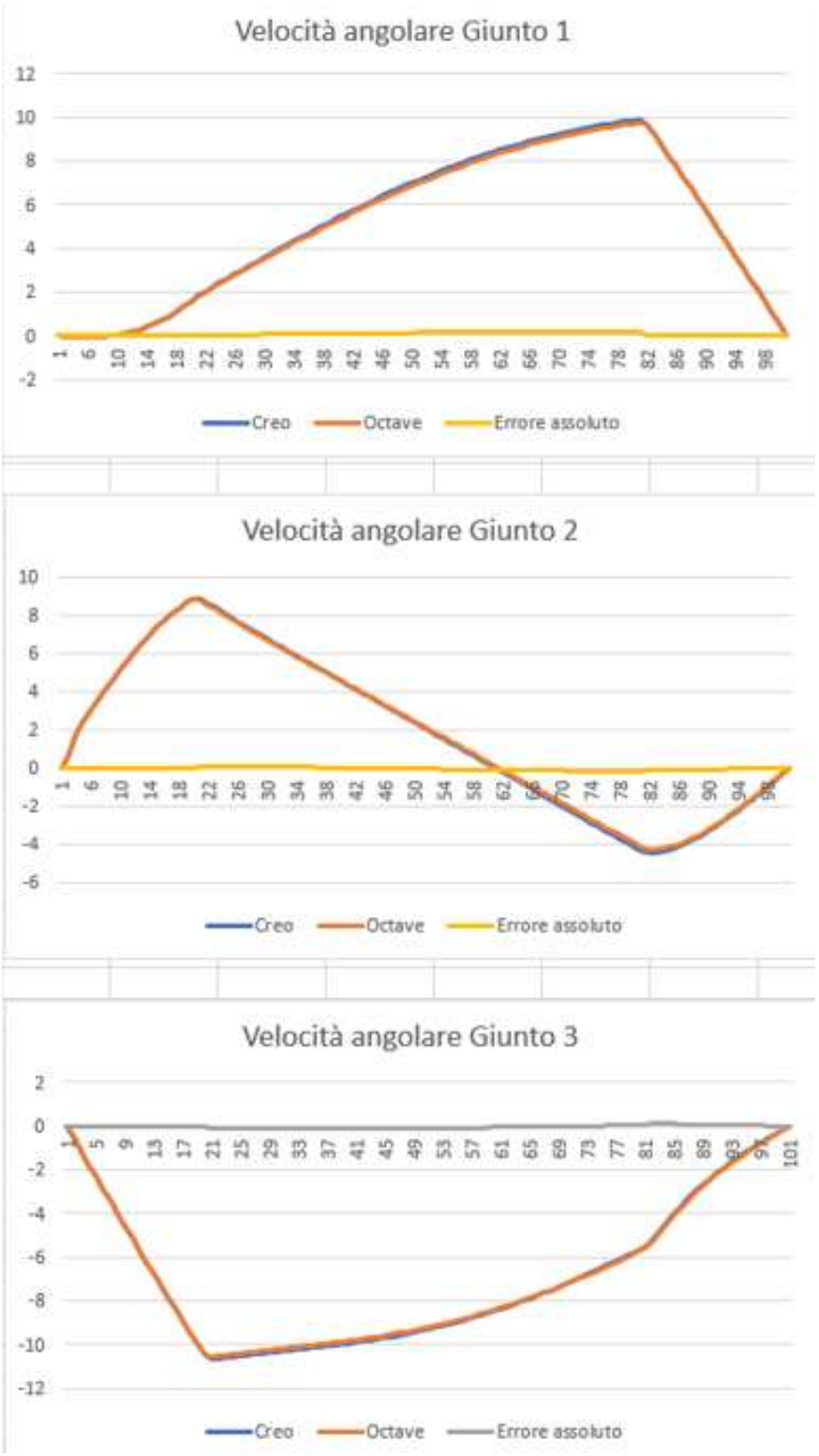


Figure 31 - Confronto delle velocità di giunto calcolate e simulate per il robot 3R

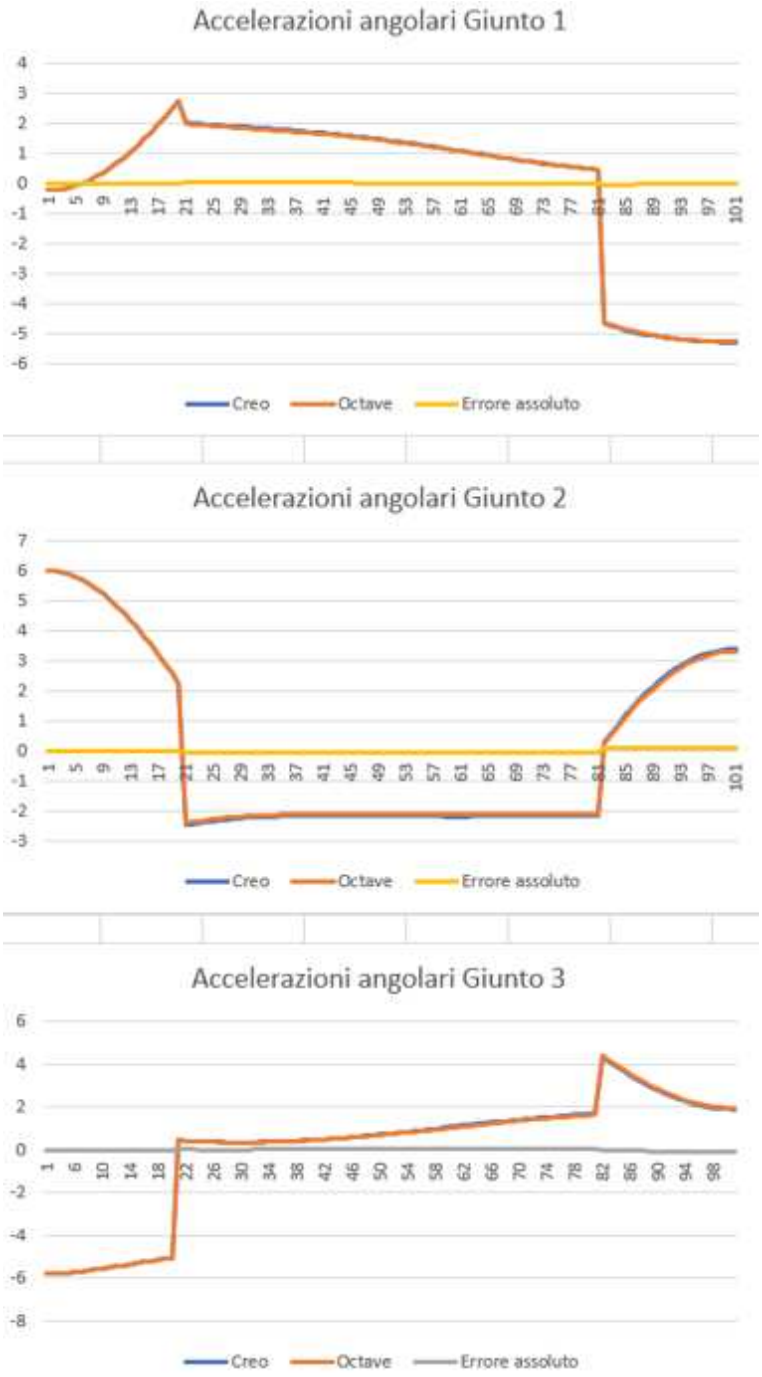


Figure 32 - Confronto delle accelerazioni di giunto calcolate e simulate per il robot 3R

Anche in questo caso abbiamo un ottimo *match* tra i risultati, sia a livello grafico che a livello numerico, seppur gli scostamenti risultino, in questo caso, più elevati rispetto al precedente:

- 0,74° per gli spostamenti;
- O,20°/s per le velocità;

- 0,08°/s^2 per le accelerazioni.

- **Robot SCARA**

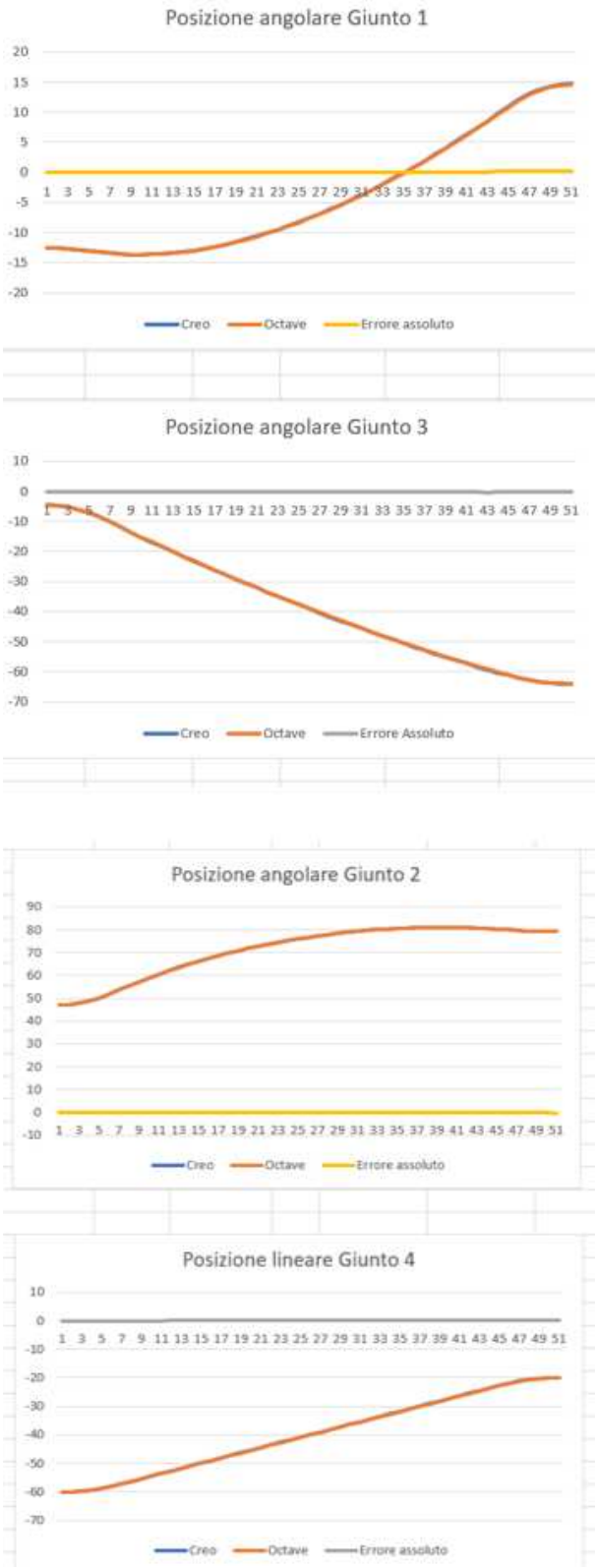


Figure 33 - Confronto delle posizioni di giunto calcolate e simulate per il robot SCARA

Figure 34 - Confronto delle velocità di giunto calcolate e simulate per il robot SCARA

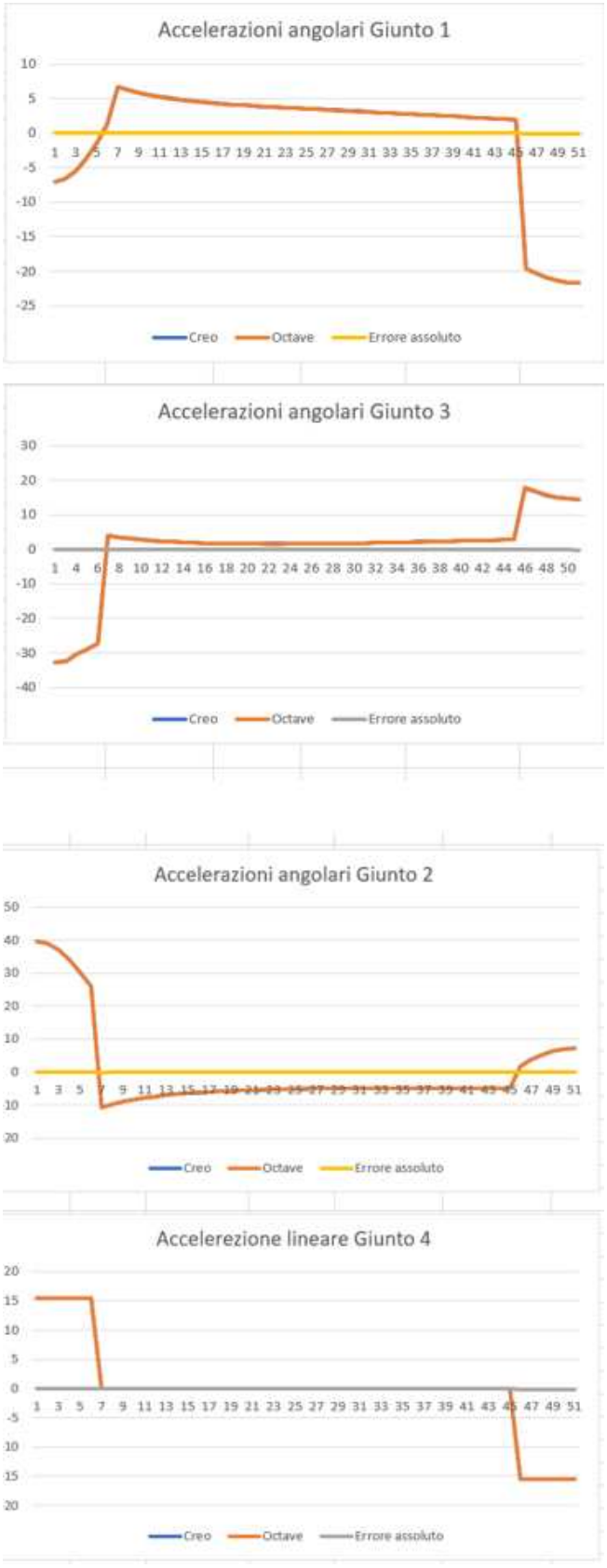


Figure 35 - Confronto delle velocità di giunto calcolate e simulate per il robot SCARA

Questo caso è stato particolarmente importante, sia perché rappresenta il caso completo in esame, sia perché inizialmente ha dato risultati che, seppur seguendo andamenti simili, mostravano un'eccessiva differenza tra di loro. Dopo ulteriori analisi ed approfondimenti circa questa problematica, si è giunti alla conclusione che il problema non fosse da imputare al modello, bensì alle modalità di movimentazione del cinematismo nella simulazione. È emerso infatti che i punti di interpolazione utilizzati risultavano pochi per seguire fedelmente la cinematica del sistema, ed è bastato utilizzare un ordine di grandezza maggiore di punti per far sì che i risultati migliorassero notevolmente, ottenendo i seguenti valori di scostamento:

- 0,12° per le posizioni;
- 0,05°/s per le velocità;
- 0,08°/s^2 per le accelerazioni.

Questo aspetto che è emerso è di grande rilevanza, in quanto ci fa capire come il modello sia perfettamente funzionante e che questi minimi scostamenti sono dovuti semplicemente a motivi numerici del software di simulazione. Se infatti supponessimo di poter aumentare i punti di interpolazione all'infinito, vedremmo questa differenza tendere ad annullarsi.

6. APPLICAZIONE DEL MODELLO NELLA PROGETTAZIONE MECCANICA

Come anticipato nell'abstract, questo modello nasce con il chiaro intento di fornire uno strumento rapido ed efficace, utile soprattutto nelle fasi preliminari della progettazione di cinematismi personalizzati per conto di aziende clienti. Pertanto, risulta di particolare interesse, nella fase conclusiva di questo percorso, andare a verificare se esso possa essere utilizzato in tal senso, ad esempio per valutare preventivamente quale tra due layout del robot SCARA (o di qualsiasi altro meccanismo) sia meno gravoso in termini di coppia sui giunti. Ovviamente seguiranno poi fasi di studio più approfondito, supportate da strumenti più complessi e potenti, ma si potrebbe così informare fin dalle prime fasi il cliente sui vantaggi o svantaggi di una configurazione piuttosto che di un'altra e dunque andare a definire in modo più mirato quelle che saranno le specifiche di progetto.

6.1. Valutazione di layout differenti

È stato deciso in quest'ottica di realizzare concretamente due schemi del robot SCARA, immaginando, prima di addentrarsi nella progettazione di dettaglio dei vari componenti, di dover scartare il layout che risultasse maggiormente gravoso per il primo giunto.

I due schemi che sono stati realizzati, in una modellazione di massima, sono i seguenti:

- **Layout 1**

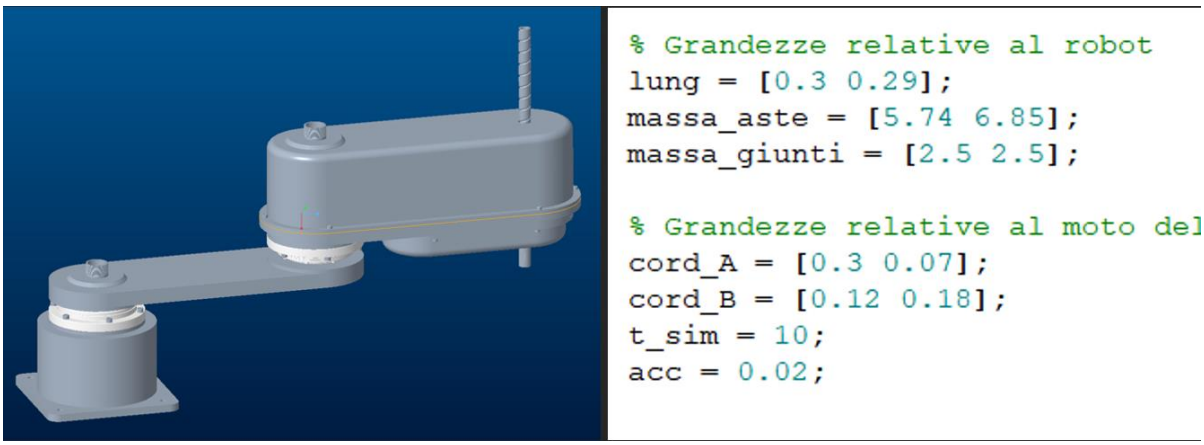
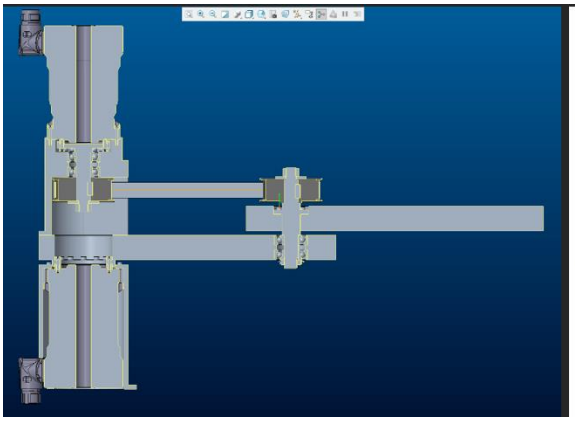


Figure 36 - Configurazione 1 del robot SCARA

In questa prima configurazione abbiamo la presenza di due motoriduttori *Harmonic Drive* da 2,5 kg di massa disposti ognuno sul rispettivo giunto, abbiamo poi due bracci di lunghezza 30 cm e 29 cm con una massa, rispettivamente di 5,74 kg e 6,85 kg.

- **Layout 2**:



```
% Grandezze relative al robot
lung = [0.25 0.25];
massa_aste = [3.7 4];
massa_giunti = [5 0.9];

% Grandezze relative al moto de
cord_A = [0.3 0.07];
cord_B = [0.12 0.18];
t_sim = 10;
acc = 0.02;
```

Figure 37 - Configurazione 2 del robot SCARA

In questa seconda configurazione il motore relativo al secondo giunto è stato spostato in corrispondenza del primo asse e connesso al secondo braccio tramite una trasmissione a cinghia, con l'idea che si potesse avere un miglioramento in termini inerziali e, dunque, una riduzione della coppia, i bracci sono entrambi di 25 cm con un peso di 3,70 kg per il primo e 4 kg per il secondo.

Sono stati dunque presi i parametri da inserire come input nel calcolatore, ovvero:

- Lunghezza delle aste;
- Massa delle aste;
- Massa dei giunti (per questi sono stati considerati il peso dei motori e quello dell'albero interposto tra i due bracci nel secondo caso);
- Moto desiderato (per entrambi i casi è stato considerato il moto da uno stesso punto A ad uno stesso punto B con profilo trapezoidale di velocità)

Il calcolatore è stato avviato è il risultato è stato da subito evidente, con i seguenti valori di coppia massimi relativi al giunto 1:

- 0,032 Nm per il primo layout;
- 0.018 Nm per il secondo layout.

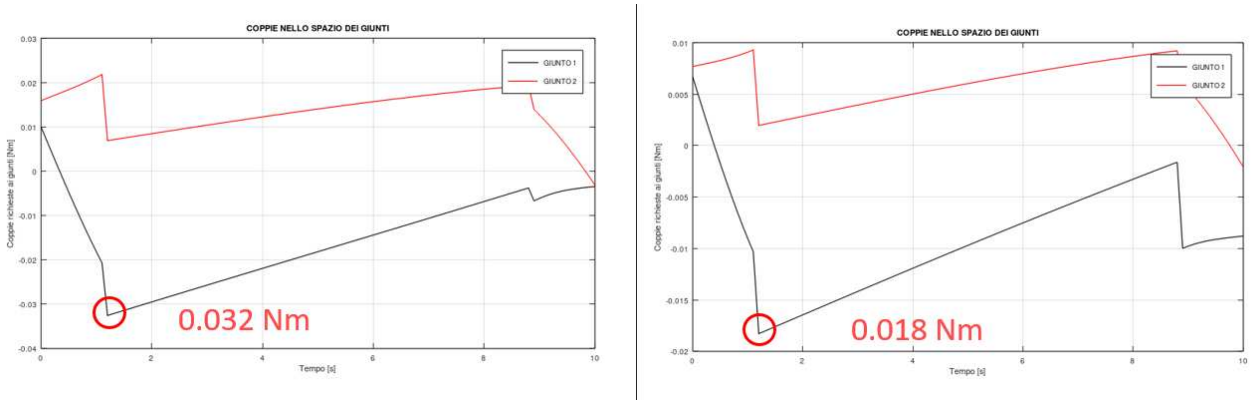


Figure 38 - Confronto delle coppie dei due layout ottenute con il modello di calcolo

6.2. Ottimizzazione del layout scelto

Il tool è stato dunque utile allo scopo previsto e grazie ad esso è possibile dedicarsi all'ottimizzazione del layout. Si riporta a titolo di esempio qualche dettaglio sulla progettazione dei primi due assi del robot, in quanto naturale proseguimento e coronamento di questo lavoro, ma ovviamente l'intera progettazione di dettaglio del robot è un iter lungo e delicato che necessita di uno studio a sé che esula dagli obiettivi di questa tesi.

Partendo dal layout 2 così come appare nel paragrafo precedente, sono state inizialmente effettuate le seguenti modifiche:

- Il supporto del secondo motore è stato fissato alla base del robot e non al braccio del robot, in questo modo non si rischia che vi siano aggrovigliamenti di cavi dovuti alla rotazione del motore su se stesso;
- L'albero flangiato connesso al motore 2 è stato ridimensionato al fine di ridurre lo spreco di materiale per lavorarlo;
- Per alleggerire le sollecitazioni sull'albero che connette i due bracci, sul quale agisce la tensione della cinghia, è stata aggiunta una staffa a L che contenga un secondo cuscinetto per supportare maggiormente l'albero

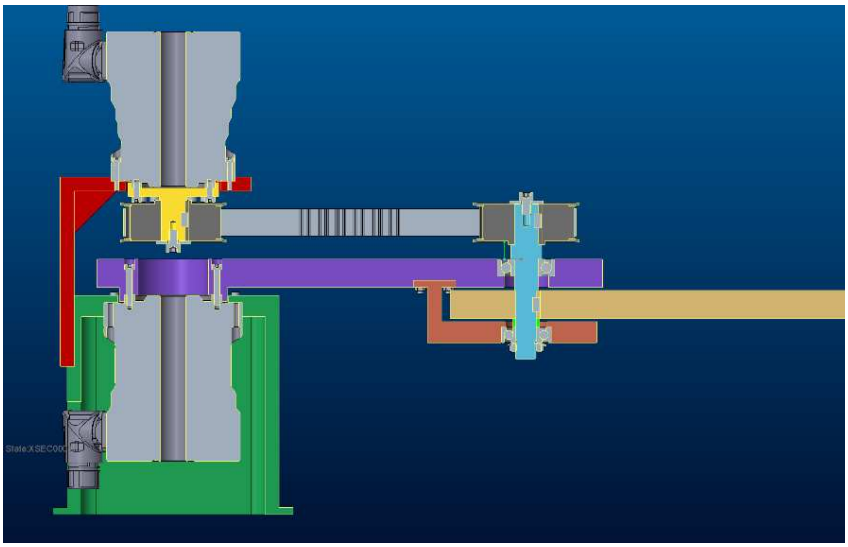


Figure 39 - Primo step di ottimizzazione dei primi due giunti

A seguito di ulteriori considerazioni sono state apportate ulteriori modifiche:

- L'albero flangiato è stato sostituito da una flangia;
- È stata aggiunta una seconda flangia che connette il motore 1 al braccio 1 per semplificare la geometria del braccio;
- Sono state aggiunte delle nervature di rinforzo nei punti maggiormente sollecitati;
- La base del robot è stata semplificata rendendola un saldato ad omega;

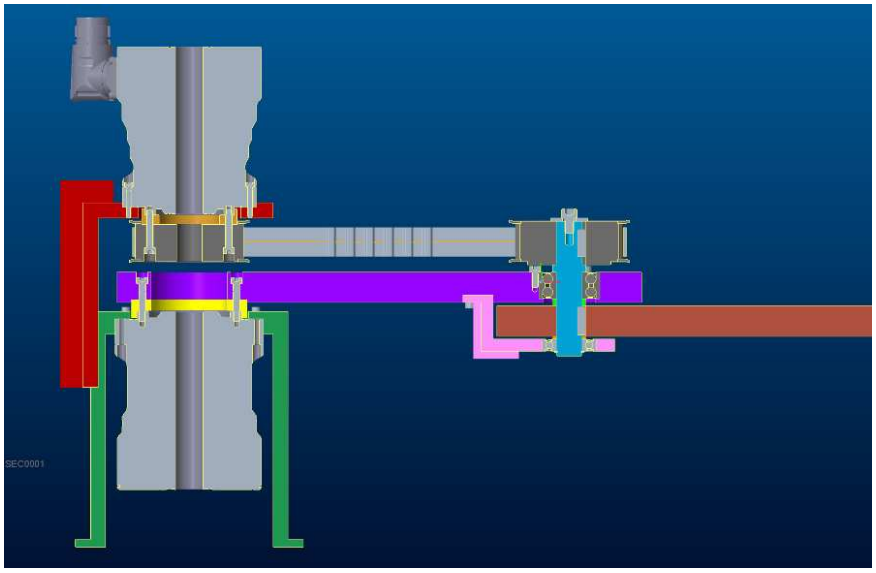


Figure 40 - Secondo step di ottimizzazione dei primi due giunti

Le ultime modifiche apportate ai primi due assi sono:

- È stato aggiunto un galoppino regolabile per il tensionamento della cinghia;
- La flangia che collega il motore 2 alla puleggia è stata definitivamente eliminata ed è stata lavorata la puleggia stessa per essere collegata direttamente al motore;
- La staffa a L di supporto all'albero è stata rinforzata nelle zone più sollecitate e sono state inserite delle spine e delle viti per semplificare il montaggio.

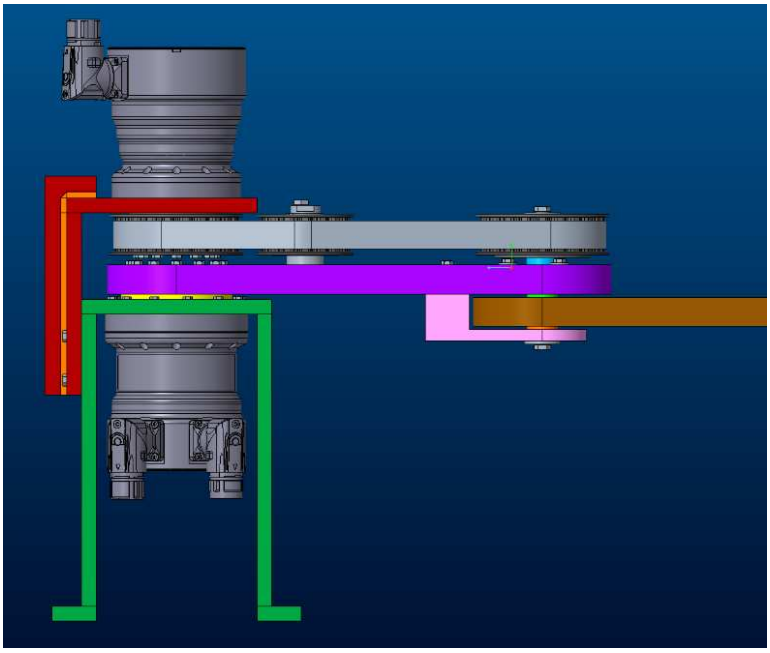


Figure 41 - ultimo step di ottimizzazione dei primi due giunti

7. CONCLUSIONE E SVILUPPI FUTURI

A conclusione di questo percorso abbiamo potuto constatare come l'utilizzo di modelli di calcolo rapidi, leggeri ed efficaci sia una risorsa spendibile nel contesto della progettazione meccanica, utile soprattutto a supporto delle prime fasi di sviluppo, in cui, nell'ambito competitivo della consulenza è essenziale fornire soluzioni di massima, bozze e layout sia in tempi brevi, sia con criterio e rigore, in quanto impostare correttamente i progetti sin dalle loro fasi iniziali aiuta ad evitare perdite di tempo e risorse nelle fasi successive.

Questo lavoro si pone come un primo modello che, nonostante abbia già evidenziato le sue potenzialità, può essere ulteriormente arricchito, esteso e approfondito nel corso di studi futuri.

Le possibili estensioni sono infatti molteplici, ad esempio si potrebbe:

- Realizzare il corrispettivo modello che operi nello spazio dei giunti, in modo da poter essere utilizzato anche per tutte quelle applicazioni che non richiedono traiettorie specifiche, ma ad esempio di massimizzare il tempo ciclo, come ad esempio nel *pick&place.*
- Aumentare la possibilità di personalizzazione del tool, migliorandone l'interazione con l'utente, ad esempio aggiungendo la possibilità di scegliere il numero di aste, il tipo di giunto e la posizione di questi ultimi.
- Approfondire le verifiche anche dal punto di vista dinamico, senza però aggiungere troppa complessità al modello, perdendo di fatto i vantaggi dello strumento.
- Aumentare i controlli, come ad esempio un controllo sulle singolarità.
- Possibilità di mostrare non solo le coppie che dovrebbero fornire i motori, ma anche l'eventuale *duty cycle* che dovrebbero sopportare.

BIBLIOGRAFIA

[1] SMARTENGINEERING, «SMARTENGINEERING,» [Online]. Available: https://www.sengineering.it/chi-siamo/.

[2] G. Legnani e I. Fassi, *Robotica industriale*, CittàStudi Edizioni, 2019.

[3] B. Siciliano, L. Sciavicco, L. Villani e G. Oriolo, *Robotica. Modellistica, pianificazione e controllo*, McGraw-Hill, 2008.

RINGRAZIAMENTI

Sono arrivato al traguardo che ho prima sognato, poi immaginato e infine tanto atteso. Diversamente dalla laurea triennale questa è veramente la fine di un capitolo. Ma allo stesso tempo, davanti a me se ne apre già un altro, ancora più grande, affascinante e meraviglioso, che spero possa darmi ancora più soddisfazioni. Di questi anni porterò un ricordo indelebile fatto di gioie, sofferenze, amore, amicizia… la quantità di esperienze che ho fatto in questi anni accademici è impossibile da raccontare in queste poche righe. Ma se c'è una cosa che posso fare è essere grato.

Desidero ringraziare il mio relatore, il Professor Massimo Callegari, per avermi fatto prima appassionare all'ambito della robotica e poi per avermi permesso di svolgere questo lavoro di tesi, lo ringrazio inoltre per avermi fornito preziosi consigli anche all'ultimo minuto.

Ringrazio SMARTENGINEERING s.p.a. per avermi accolto come tirocinante, ma trattandomi fin da subito come parte della squadra, con grande attenzione e umanità. La ringrazio per gli strumenti, le risorse e il know-how che mi ha messo a disposizione per svolgere questo lavoro. Non da ultimo la ringrazio per avermi fatto diventare parte di loro a tutti gli effetti, dandomi modo di approcciarmi al mondo del lavoro per la prima volta.

Vorrei ringraziare in particolar modo l'ing. Fabio Ricci Curbastro, che, in qualità di tutor aziendale mi ha saputo introdurre al lavoro, consigliare e spronare. Grazie di tutto.

Ringrazio la mia famiglia, che è stata un supporto solido in questa fase di grandi cambiamenti che, come tale, è sempre molto delicata.

Grazie a chi ci è sempre stato e a chi c'è da poco, ma mi vuole bene come se ci fosse da sempre.