
**UNIVERSITA' POLITECNICA DELLE MARCHE**

FACOLTA' DI INGEGNERIA

Corso di Laurea Magistrale in Ingegneria Meccanica

# Identificazione dei parametri dinamici di un robot collaborativo industriale

# Dynamic parameters identification of an industrial collaborative robot

Relatore:

Prof. Matteo Claudio Palpacelli

Tesi di Laurea di:

Sofia Fabbretti

Correlatore:

Dott. Ing. Luca Carbonari

A.A. 2021 / 2022

# Indice

# Elenco delle figure

# Elenco delle tabelle

# Capitolo 1

# Introduzione

L'importanza di avere un insieme appropriato di parametri dinamici di un manipolatore robotico risiede nell'applicazione degli stessi nell'ambito degli algoritmi di controllo *model-based* avanzati nonché nelle applicazioni di pianificazione della traiettoria. L'accuratezza di tali parametri riveste un ruolo fondamentale nella precisione, nelle prestazioni, nella sicurezza e nella robustezza di tali algoritmi, in quanto consentono la sostituzione di un sistema meccanico reale con un modello software dello stesso, garantendo risultati affidabili e costi ridotti dovuti all'assenza di test sperimentali. Mentre la struttura geometrica dei robot è resa nota, tali parametri non sono sempre forniti dai produttori e spesso non sono misurabili in maniera diretta. Pertanto, per le motivazioni sopracitate, l'identificazione dei parametri dinamici dei manipolatori ha suscitato un crescente interesse da parte dei ricercatori.

L'obiettivo della presente trattazione è quello di proporre una metodologia per l'identificazione dei parametri dinamici di un robot collaborativo industriale nonché di presentare i risultati ottenuti dall'applicazione della stessa. In particolare si è scelto come caso di studio il robot CRX-10iA/L di produzione FANUC.

Nel primo capitolo è riportato lo stato dell'arte della robotica collaborativa, quindi le normative, la struttura dei cobot ed in particolare quella del manipolatore preso in esame, nonché l'importanza dell'identificazione parametrica in tale contesto.

Di seguito, nel secondo capitolo, sono riportate nel dettaglio la modellazione cinematica, quindi la convenzione di Denavit-Hartenberg adottata, e la modellazione dinamica del robot, implementata attraverso le formulazioni di Newton-Eulero e di Lagrange.

Nel terzo capitolo è riportata la procedura di verifica dei modelli presentati al capitolo precedente descritta nei suoi steps fondamentali, ovvero la realizzazione del modello multibody del manipolatore all'interno del software MSC ADAMS e la messa a punto della tecnica della co-simulazione, ovvero la collaborazione tra i software MSC ADAMS e Simulink nel processo di verifica della correttezza dei modelli.

Di seguito, nel quarto capitolo, è riportata nel dettaglio la procedura di identificazione dei parametri dinamici del manipolatore adottata nella presente trattazione, quindi la costruzione e la riduzione della matrice di regressione, nonché l'identificazione dei parametri e la loro successiva verifica, attraverso l'utilizzo dell'indice di prestazione NRMSE.

Infine, nel capitolo conclusivo, è riportato un breve riepilogo del contenuto del presente elaborato e dei risultati ottenuti, nonché un cenno a possibili sviluppi futuri.

## 1.1 Robotica collaborativa

La robotica collaborativa rappresenta la rivoluzione della robotica industriale. Essa costituisce una delle tecnologie abilitanti dell'Industria 4.0 e riporta la sua massima espressione nell'ambito dei processi produttivi e dell'automazione

industriale, in quanto consente di raggiungere il giusto equilibrio tra automazione e flessibilità, aspetto chiave nell'epoca della personalizzazione di massa. "Collaboratività" significa accesso al sistema robotizzato e allo spazio di lavoro, dove il robot e l'operatore possono compiere azioni funzionalmente legate, simultanee e contestuali, unendo le loro capacità: precisione, ripetibilità e produttività da un lato e *know how*, creatività e abilità di *problem solving* dall'altro. Sarebbe più corretto parlare di applicazioni collaborative piuttosto che di robot collaborativi, in quanto la collaborazione deriva dall'uso di una macchina, oltre che dalle sue caratteristiche. I robot collaborativi, o cobot, sono dispositivi di nuova generazione leggeri e flessibili concepiti per interagire fisicamente con l'uomo in sicurezza e senza barriere o gabbie protettive, a differenza dei dispositivi tradizionali (Figure 1.1 e 1.2). Questo è reso possibile grazie alla presenza di sensoristica integrata (pelle capacitiva, sensori di forza e coppia, sensori di visione, ecc.), limitatori di forza e velocità, membri leggeri ed ergonomici con giunti arrotondati o plastificati e assenza di motori, cavi, componenti meccanici ed elettronici esposti. Pertanto, grazie alle peculiarità sopra riportate, i robot collaborativi garantiscono benefici notevoli all'operatore, nonché alla produzione e all'ambiente di lavoro, quali ad esempio assistenza all'uomo nelle operazioni che richiedono sforzi e ripetizioni non compatibili con il suo benessere ergonomico, supporto cognitivo all'operatore, aumento di produttività e qualità, riduzione dei tempi, ottimizzazione dello spazio di lavoro, intuitività nella programmazione, ecc.

**Figura 1.1** *Esempio di robot non collaborativi.*


**Figura 1.2** *Esempio di robot collaborativo.*

Pertanto è possibile riassumere, in linea generale, punti di forza e debolezza della robotica collaborativa, in modo tale da comprendere le motivazioni dell' evoluzione di tale contesto e le limitazioni che ne conseguono.

Vantaggi:

- miglioramento delle condizioni di lavoro;

- aumento della produttività;
- sfruttamento dell’esperienza delle maestranze specializzate;
- semplificazione del layout produttivo;
- semplificazione dell’installazione;
- facilità di programmazione.

Svantaggi:

- velocità ridotte;
- minori carichi paganti;
- minori precisioni;
- maggiore costo del robot.

A seguito di quanto riportato precedentemente, è possibile quindi affermare che la robotica collaborativa combina i vantaggi dell'automazione con la flessibilità e le abilità cognitive dei lavoratori umani. I robot industriali tradizionali, infatti, possono eseguire i compiti per cui sono programmati in modo continuo e con livelli di precisione, velocità e ripetibilità impossibili da raggiungere per gli operatori, tuttavia, mancano di versatilità e non possono adattarsi in maniera efficiente ad ambienti di lavoro dinamici o a cambiamenti della produzione, risultando quindi inadatti per piccoli lotti di produzione. Al contrario, i lavoratori umani presentano un'innata flessibilità e capacità di adattarsi a eventi imprevisti e di mantenere forti abilità decisionali anche in ambienti dinamici e complessi [17].

### 1.1.1 Normative per robot collaborativi

Poiché nell’ambito della robotica collaborativa operatori e robot lavorano a stretto contatto, sono necessarie normative e procedure di sicurezza che

vadano a salvaguardare l'uomo nonché l'integrità dell'ambiente aziendale. Di seguito nelle Tabelle 1.1, 1.2 e 1.3 sono riportate le leggi e le norme generali maggiormente rilevanti a livello dell'Unione Europea [7].

| | |
|---|---|
| 2006/42/EC | Machinery Directive (MD) |
| 2009/104/EC | Use of Work equipment Directive |
| 89/654/EC | Workplace Directive |
| 2001/95/EC | Product Safety Directive |
| 2006/95/EC | Low Voltage Directive (LVD) |
| 2004/108/EC | Electromagnetic compatibility Directive (EMC) |

**Tabella 1.1** *Direttive EU [7].*

| | |
|---|---|
| EN ISO 12100 | Safety of machinery - General principles for design - Risk assessment and risk reduction |
| EN ISO 13849-1/2 | Safety of machinery - Safety-related parts of control systems - Part 1: General principles for design, Part 2: Validation |
| EN 60204-1 | Safety of machinery - Electrical equipment of machines - Part 1: |

| | |
|---|---|
| | General requirements |
| IEC 62061 | Safety of machinery – Functional safety of safety-related electrical, electronic and programmable electronic control systems |

**Tabella 1.2** *Norme generali indicative [7].*

| | |
|---|---|
| EN ISO 10218-1 | Robots and robotic devices - Safety requirements for industrial robots - Part 1: Robots |
| EN ISO 10218-2 | Robots and robotic devices - Safety requirements for industrial robots - Part 2: Robot systems and integration |
| ISO/PDTS 15066 | Robots and robotic Devices – Collaborative Robots |

**Tabella 1.3** *Norme relative ai robot [7].*

Delle norme precedentemente elencate, alcune risultano avere maggiore rilievo a livello europeo, ed è quindi opportuno illustrarle in maniera più dettagliata.

La Direttiva Macchine 2006/42/CE del Parlamento Europeo definisce i requisiti essenziali, in materia di sicurezza e di salute pubblica, ai quali devono rispondere le macchine in occasione della loro progettazione, fabbricazione e del loro funzionamento, prima della loro immissione sul mercato [8].

La EN ISO 12100 è una norma generale relativa alla sicurezza dei macchinari la quale specifica i principi per la valutazione del rischio e la riduzione dello stesso [9], mentre le EN ISO 10218-1 e EN ISO 10218-2 sono norme più specifiche indirizzate ai robot: la prima specifica i requisiti e le linee guida per la progettazione sicura, le misure di protezione, le informazioni per l'uso dei robot industriali e i rischi di base associati ad essi [10], mentre la seconda specifica i requisiti di sicurezza per l'integrazione di robot industriali e sistemi robotici industriali come definiti nella norma ISO 10218-1 e celle di robot industriali [11]. In particolare la norma ISO 10218-1 definisce quattro scenari di collaborazione uomo-robot (HRC, *Human-Robot Collaboration*), rappresentati graficamente nella Figura 1.3 [17]:

- arresto di sicurezza controllato: il robot lavora prevalentemente in autonomia ma occasionalmente un operatore può entrare nel suo spazio di lavoro provocando l'arresto del dispositivo;
- guida manuale: il robot viene spostato manualmente dall'operatore permettendo la memorizzazione della traiettoria da parte del manipolatore;
- controllo di velocità: l'area di lavoro del robot è suddivisa in zone in modo che esso possa reagire in modo diverso in base alla posizione dell'operatore, fino ad arrestarsi se necessario;
- limitazione di forza e potenza: il robot è in grado di percepire forze applicate alla sua struttura e di arrestarsi se queste superano una soglia prefissata, dissipando le stesse in caso di impatto.

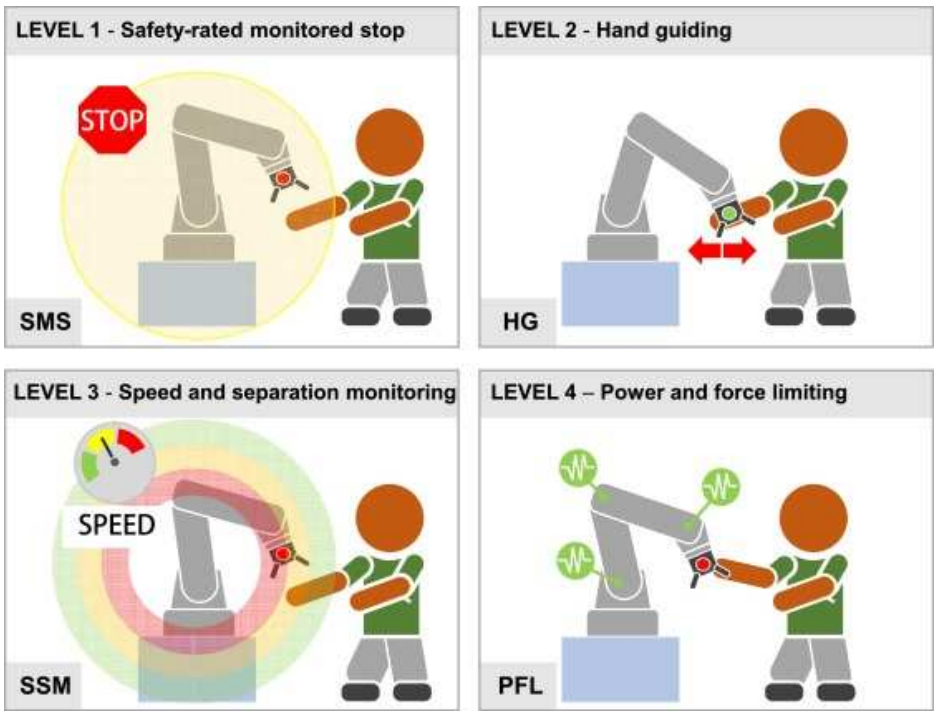


**Figura 1.3** *Scenari di collaborazione uomo-robot previsti dalla ISO 10218-1 [17].*

Infine la ISO/TS 15066:2016 specifica i requisiti di sicurezza per i sistemi robotici industriali collaborativi e l'ambiente di lavoro e integra i requisiti e le linee guida sul funzionamento dei robot industriali collaborativi forniti dalle ISO 10218-1 e ISO 10218-2 [12].

Le norme relative alle operazioni collaborative sono quindi necessarie per garantire la sicurezza umana e il corretto funzionamento delle macchine. Per questo motivo sono spesso integrate ad esse strumenti per l'analisi e la valutazione del rischio, aventi lo scopo di ridurre la probabilità e la gravità dell'evento.

### 1.1.2 Struttura dei robot

Secondo la norma UNI EN ISO 8373, un robot è definito come “una macchina di manipolazione comandata automaticamente, riprogrammabile, multi-scopo, con più gradi di libertà, di tipo fisso o mobile per l’impiego nelle applicazioni di automazione industriale” [13], in grado di modificare l’ambiente in cui opera attraverso regole intrinseche allo stesso dispositivo e dati acquisiti sul suo stato e sull’ambiente. Infatti la robotica è comunemente definita come la scienza che studia “la connessione intelligente tra percezione e azione” [5]. Facendo riferimento a queste definizioni si percepisce come un sistema robotico sia un sistema complesso, a sua volta caratterizzato da quattro sottosistemi, rappresentati nella Figura 1.4: il sistema meccanico, il sistema di azionamento, il sistema di controllo e il sistema di rilevamento. Il sistema meccanico costituisce la parte essenziale di un robot, in quanto comprende l’apparato di manipolazione (bracci meccanici, dispositivi terminali, mani artificiali), il quale viene animato dal sistema di azionamento. La capacità percettiva è affidata invece al sistema di rilevamento, il quale consente l’acquisizione sia di dati interni relativi allo stato robot sia di dati esterni propri dell’ambiente in cui esso opera. Infine la capacità di connettere in modo intelligente la percezione e l’azione è attribuita al sistema di controllo, il quale comanda l’esecuzione dell’azione secondo gli obiettivi prefissati tramite pianificazione e i vincoli imposti dal robot e dall’ambiente [5].

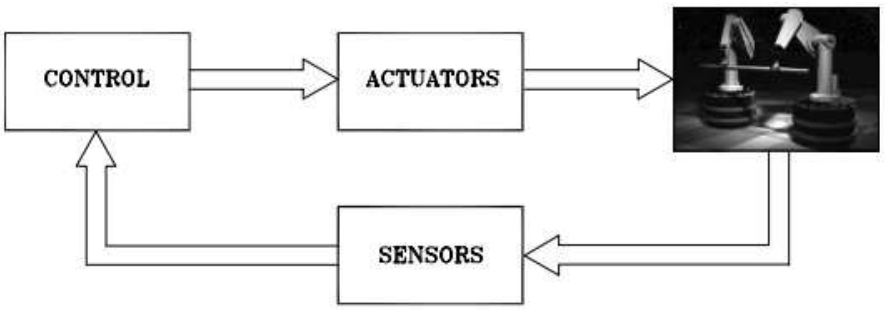


**Figura 1.4** *Sottosistemi di un sistema robotico [5].*

La struttura di un manipolatore robotico è caratterizzata da una sequenza di corpi rigidi (*links*) connessi attraverso articolazioni (*joints*), rappresentati nella Figura 1.5. Tipicamente esso è costituito da un braccio che ne assicura la mobilità, un polso che conferisce manualità e un dispositivo terminale (*end-effector*) che esegue il compito richiesto all'interno dello spazio di lavoro (*workspace*), ovvero lo spazio a cui il terminale può accedere. Le articolazioni, che consentono la mobilità del robot, possono garantire un moto di rotazione se il giunto è rotoidale o un moto di traslazione se il giunto è prismatico. I gradi di libertà garantiti da queste ultime devono essere adeguatamente distribuiti lungo la struttura meccanica, in modo da averne a sufficienza per eseguire il compito richiesto. Poiché l'attività programmata di solito coinvolge posizionamento e orientamento di un oggetto nello spazio, i robot spesso presentano sei gradi di libertà (DOF, *Degrees Of Freedom*).

Tipicamente i manipolatori robotici sono a base fissa, attorno alla quale avviene il movimento del robot, e a catena cinematica aperta, ovvero vi è una sola sequenza di corpi rigidi che collega le due estremità della catena; esistono tuttavia anche manipolatori a base mobile e a catena cinematica chiusa [5].

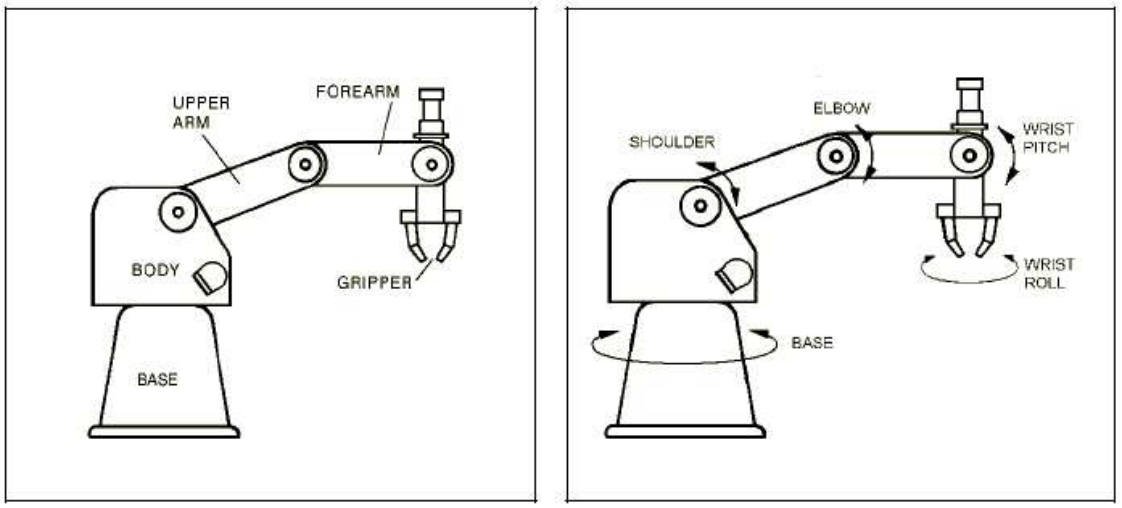


**Figura 1.5** *Esempio di collegamenti (sinistra) e giunti (destra) di un robot [6].*

### 1.1.2.1 FANUC CRX-10iA/L

Il robot collaborativo scelto come caso di studio nel presente elaborato è il CRX-10iA/L di produzione FANUC, rappresentato nella Figura 1.6.

**Figura 1.6** *Robot CRX-10iA/L.*

Questo appartiene alla *CRX series* [1] della casa produttrice, gamma caratterizzata da cinque modelli differenti per capacità di carico (*payload*) e sbraccio (*reach*), riportati nella Tabella 1.4.

| | Capacità di carico [kg] | Sbraccio [mm] |
|---|---|---|
| CRX-5iA | 5 | 994 |

| | | |
|---|---|---|
| CRX-10iA | 10 | 1249 |
| CRX-10iA/L | 10 | 1418 |
| CRX-20iA/L | 20 | 1418 |
| CRX-25iA | 25 | 1889 |

**Tabella 1.4** *Capacità di carico e sbraccio dei modelli della gamma CRX [1].*

Il robot CRX-10iA/L presenta delle caratteristiche proprie dell'intera categoria di appartenenza [1]:

- collaboratività: è la capacità di lavorare a stretto contatto con gli esseri umani anche in assenza di gabbie di protezione, amplificata dalla presenza di sensori a elevata sensibilità che possono attivare un arresto di sicurezza immediato in caso di contatto con un corpo umano;
- massima affidabilità: grazie alla conformità agli standard previsti dalla normativa (ISO 10218-1) i robot possono totalizzare in media fino a 3850 ore prima che sia consigliato effettuare la manutenzione preventiva;
- facilità di programmazione: attraverso l'ausilio del *teach pendant* è possibile utilizzare la funzione di programmazione tramite movimentazione manuale, che consente di spostare il robot nella posizione desiderata e salvarla sul tablet con la funzionalità *drag&drop*;
- installazione rapida: i robot dispongono di semplici software di *plug in* per poter installare le interfacce di utilizzo delle periferiche (pinze, telecamere, ecc.) e grazie al loro peso ridotto e alla loro compattezza possono essere facilmente installati in un'ampia gamma di applicazioni.

Il robot FANUC CRX-10iA/L è dotato di sei assi e sei giunti rotoidali, ed è quindi caratterizzato da sei gradi di libertà nello spazio. Si tratta di una catena cinematica aperta in cui i giunti fungono da collegamento tra un *link* e l'altro. La struttura del robot è riportata nella Figura 1.7 di seguito [3].

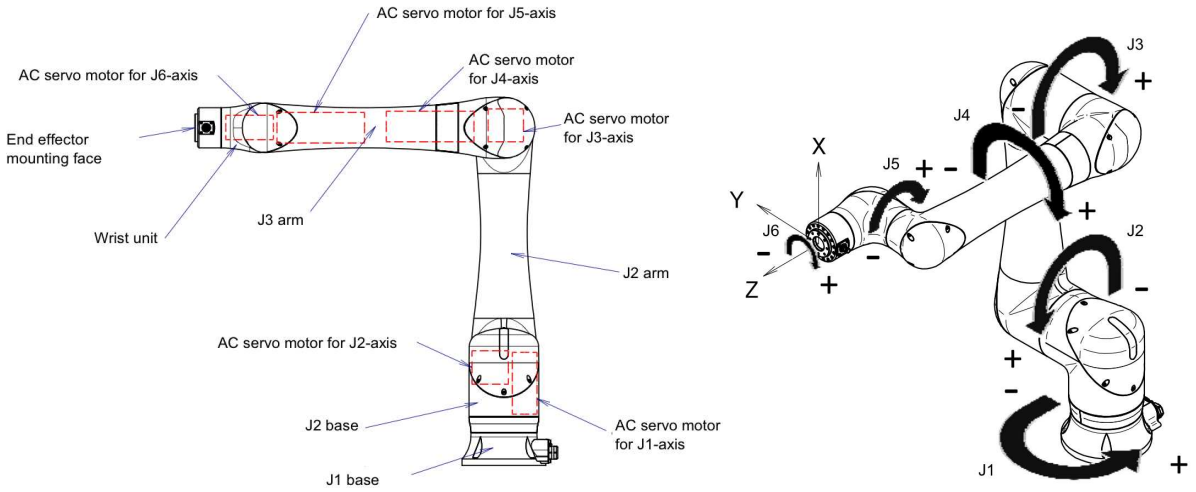


**Figura 1.7** *Configurazione del robot[3].*

Lo spazio di lavoro del robot e il range di movimento e le velocità massime dei giunti sono riportati rispettivamente nelle Figure 1.8 e 1.9 [2].

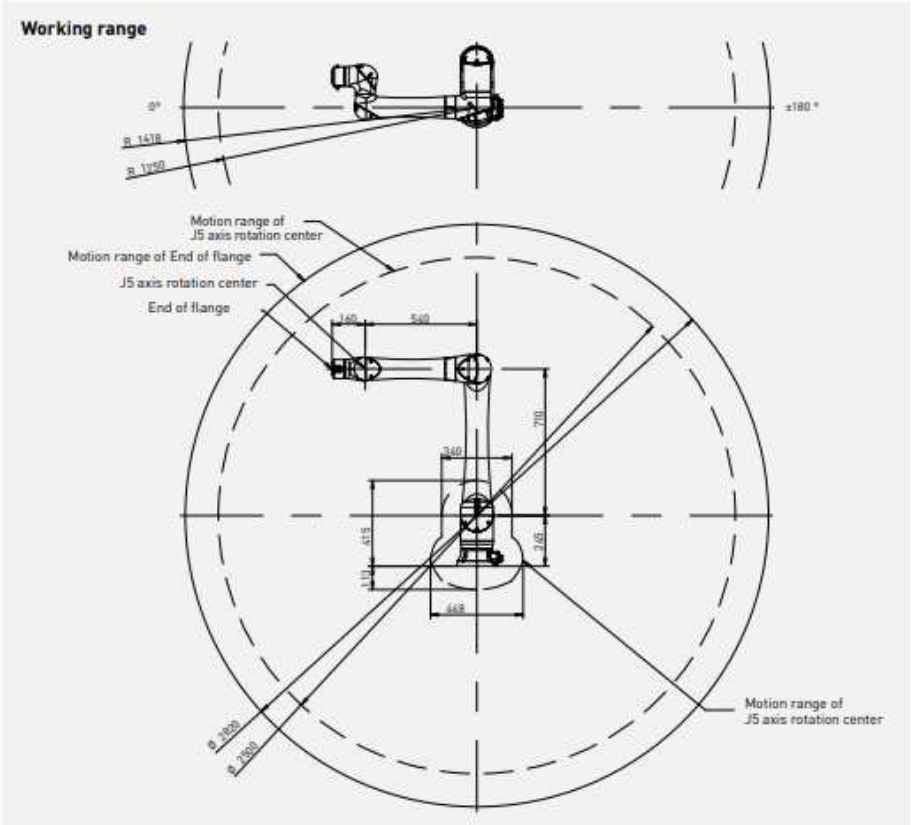


**Figura 1.8** *Spazio di lavoro del robot [2].*

| Controlled axes | Repeatability (mm) | Mechanical weight (kg) | Motion range (°) | | | | | | Maximum speed (°/s) *2 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | J1 | J2 | J3 | J4 | J5 | J6 | J1 | J2 | J3 | J4 | J5 | J6 |
| 6 | ± 0.04* | 40 | 360 | 360 | 540 | 380 | 360 | 450 | 120 | 120 | 180 | 180 | 180 | 180 |

**Figura 1.9** *Range di movimento e velocità massime dei giunti [2].*

Ulteriori specifiche generali del CRX-10iA/L sono riportate nella successiva Figura 1.10 [3].

| Item | | Specification | |
|---|---|---|---|
| Model | | CRX-10*i*A | CRX-10*i*A/L |
| Type | | Articulated type | |
| Controlled axes | | 6-axis(J1, J2, J3, J4, J5, J6) | |
| Reach | | 1249 mm | 1418 mm |
| Installation | | Floor, Upside-down, Wall & Angle mount (Note 2) | |
| Motion range | J1-axis | 380° (6.63 rad) | 360° (6.28 rad) |
| | J2-axis | 360° (6.28 rad) | |
| | J3-axis | 570° (9.95 rad) | 540° (9.95 rad) |
| | J4-axis | 380° (6.63 rad) | |
| | J5-axis | 360° (6.28 rad) | |
| | J6-axis | 380° (6.63 rad) | |
| Maximum speed (NOTE 3) | | 1000mm/s (NOTE 4) (maximum speed 2000mm/s (NOTE 5)) | |
| Maximum load at wrist | | 10kg | |
| Allowable load moment at wrist | J4-axis | 34.8Nm | |
| | J5-axis | 26.0Nm | |
| | J6-axis | 11.0Nm | |
| Allowable load inertia at wrist | J4-axis | 1.28kg.m$^2$ | |
| | J5-axis | 0.90kg.m$^2$ | |
| | J6-axis | 0.30kg.m$^2$ | |
| Repeatability (NOTE 6) | | ±0.05mm | |
| Robot mass | | 40kg | 40kg |
| Dust proof and drip proof mechanism (NOTE 7) | | Conform to IP67 | |
| Acoustic noise level | | Less than 70dB (NOTE 8) | |
| Installation environment | | Ambient temperature: Operating 0 to 45℃ (NOTE 9)<br>Storage,Transport -10 to 60℃<br>Ambient humidity: Normally 75%RH or less (No dew or frost allowed)<br>Short time 95%RH or less (Within 1 month)<br>Permissible altitude: Above the sea 1000m or less<br>Free of corrosive gases (NOTE 10)<br>Vibration acceleration : 4.9m/s$^2$ (0.5G) or less (NOTE 11)<br>Environment without fire | |

**Figura 1.10** *Specifiche tecniche del robot CRX-10iA/L [3].*

## 1.1.3 Identificazione parametrica

Essendo lo scopo di questo elaborato l'identificazione dei parametri dinamici di un robot collaborativo industriale, risulta essere necessario riportare dei cenni relativi allo stato dell'arte in questo campo, in modo tale da poter comprendere l'utilità e l'importanza dell'identificazione parametrica nel settore della robotica nonché individuare alcune delle tecniche maggiormente utilizzate per il raggiungimento di tale obiettivo.

I modelli ad alta fedeltà dei robot industriali (HF, *High Fidelity*) vengono spesso utilizzati per prevedere il comportamento dei manipolatori nell'esecuzione di diversi *tasks* e in diverse condizioni operative ma potrebbero avere un ruolo fondamentale anche nel rilevamento e previsione dei guasti. In questo contesto

è quindi necessaria una profonda conoscenza del robot preso in esame, così da poter eseguire simulazioni in maniera affidabile. Tuttavia, come spesso accade in ambito industriale, i parametri cinematici sono di solito forniti dal costruttore mentre questo non si verifica nel caso dei parametri dinamici. Questo potrebbe portare ad una valutazione errata della movimentazione del robot così come dei rischi legati ad un'applicazione collaborativa. Per superare queste limitazioni sono stati quindi sviluppati diversi algoritmi e strategie per l'identificazione dei parametri dinamici di un manipolatore industriale, alcuni dei quali sono riportati di seguito.

In [14] viene presentata una procedura di identificazione parametrica basata sulla linearizzazione della formulazione dinamica del robot rispetto ad un insieme di parametri dinamici adeguatamente definiti, attraverso il calcolo della matrice di regressione e la sua successiva riduzione per mezzo di algoritmi di *QR decomposition* o *SVD* (*Singular Value Decomposition*).

In [15] la procedura proposta consente di identificare i parametri dinamici in un punto operativo dell'area di lavoro del manipolatore attraverso la realizzazione di un moto perturbato attorno al punto operativo stesso.

Infine in [16] sono presentati tre metodi basati sui minimi quadrati linearizzati, i quali sono tra gli approcci standard maggiormente utilizzati per la risoluzione di sistemi sovradeterminati: il metodo dei minimi quadrati ordinari (OLS), i minimi quadrati pesati (WLS) e i metodo dei minimi quadrati riponderato in modo iterativo (IRLS).

Lo stato dell'arte nel contesto dell'identificazione parametrica è quindi ricco di approcci differenti al problema, tra i quali è necessario scegliere quello opportuno a seconda dell'impostazione del problema stesso.

# Capitolo 2

# Modellazione cinematica e dinamica

Modellazione e simulazione sono aspetti di notevole importanza nell'ambito della robotica, in quanto consentono di comprendere a fondo il comportamento dei sistemi reali nonché di evitare la sperimentazione effettiva, che spesso si presenta come dispendiosa in termini di tempo e denaro. Inoltre, essendo l'obiettivo di questa trattazione identificare i parametri dinamici del robot collaborativo FANUC CRX-10iA/L, l'implementazione di un modello del manipolatore risulta essere indispensabile per il raggiungimento del fine ultimo del presente elaborato.

Di seguito sono presentati i modelli cinematico e dinamico del robot preso in esame in questa trattazione, nonché la convenzione di Denavit-Hartenberg sulla base della quale entrambi sono realizzati, implementati attraverso l'ausilio del software MATLAB.

## 2.1 Convenzione di Denavit-Hartenberg

Per poter affrontare lo studio cinematico e dinamico del robot preso in esame, è necessario comprendere come ogni singolo giunto del meccanismo debba essere controllato cinematicamente per poter ottenere una data posizione dell'*end-effector* rispetto alla base del meccanismo o a qualsiasi altro membro dello stesso. Per fare questo è necessario dotare ogni membro *i* del meccanismo preso in esame di un sistema di riferimento (o *frame*) i-esimo ad

esso solidale, caratterizzato da un'origine (indicata con $O_i$) ed una triade di vettori ortogonali ($\hat{x}_i\ \hat{y}_i\ \hat{z}_i$). Sebbene i sistemi di coordinate solidali ad ogni membro possano essere posizionati arbitrariamente è vantaggioso, sia in termini di coerenza che di efficienza computazionale, aderire ad una convenzione per la localizzazione delle terne. La rappresentazione di Denavit-Hartenberg (D-H) è un metodo sistematico che consente di descrivere attraverso quattro parametri la relazione cinematica tra una coppia di membri adiacenti connessi in modo seriale. Bisogna osservare che esistono molte convenzioni che vanno sotto questo nome, pur differendo per lievi dettagli l'una dall'altra; in questa trattazione si è scelto di utilizzare la notazione secondo cui il giunto i-esimo collega il link i-esimo ed il successivo, mentre l'asse di rotazione $z_i$ è allineato con l'asse del giunto *i + 1* [18].

Questa notazione prevede la numerazione dei membri del robot da "1" a "n" a partire dal telaio, convenzionalmente indicato come membro "0"; analogamente sono indicate le coppie cinematiche che uniscono i vari membri.

Per prima cosa si definiscono le terne solidali ai membri i-esimi secondo la seguente procedura:

- si sceglie l'asse $z_i$ giacente lungo l'asse del giunto *i+1*;
- si individua l'origine $O_i$ della terna *{i}* nell'intersezione dell'asse del giunto *i+1* con la normale comune tra l'asse stesso e l'asse del giunto precedente (se i due assi sono paralleli l'origine può essere scelta in un qualsiasi punto della retta che funge da distanza);
- si sceglie l'asse $x_i$ diretto lungo la normale comune del passo precedente con verso positivo dal giunto *i* al giunto *i+1*;
- si sceglie l'asse $y_i$ in modo da completare la terna ortonormale destrorsa.

Per quanto riguarda i membri *0* e *n* non esistono le normali comuni, in quanto è presente un solo asse di giunto, pertanto le terne sono definite come segue:

- per il telaio, l'origine della terna può essere scelta in un punto arbitrario sull'asse del giunto 1, con un unico vincolo relativo al posizionamento dell'asse $z_0$ che deve essere parallelo all'asse del giunto;
- per l'ultimo membro (*n*), l'origine della terna può essere scelta in qualunque punto del terminale, a patto che l'asse $x_n$ intersechi l'asse dell'ultimo giunto ad angolo retto.

Quindi la posizione e l'orientamento del sistema di riferimento *i* rispetto alla terna *i-1* sono completamente determinati dai seguenti quattro parametri, rappresentati nella Figura 2.1:

- $a_i$: distanza tra l'asse $z_i$ e l'asse $z_{i-1}$, misurata lungo la direzione positiva di $x_i$;
- $d_i$: distanza tra l'asse $x_i$ e l'asse $x_{i-1}$, misurata lungo la direzione positiva di $z_{i-1}$;
- $\alpha_i$: angolo intorno a $x_i$ tra l'asse $z_{i-1}$ e l'asse $z_i$, positivo in senso antiorario;
- $\theta_i$: angolo intorno all'asse $z_{i-1}$ tra l'asse $x_{i-1}$ e l'asse $x_i$, positivo in senso antiorario.

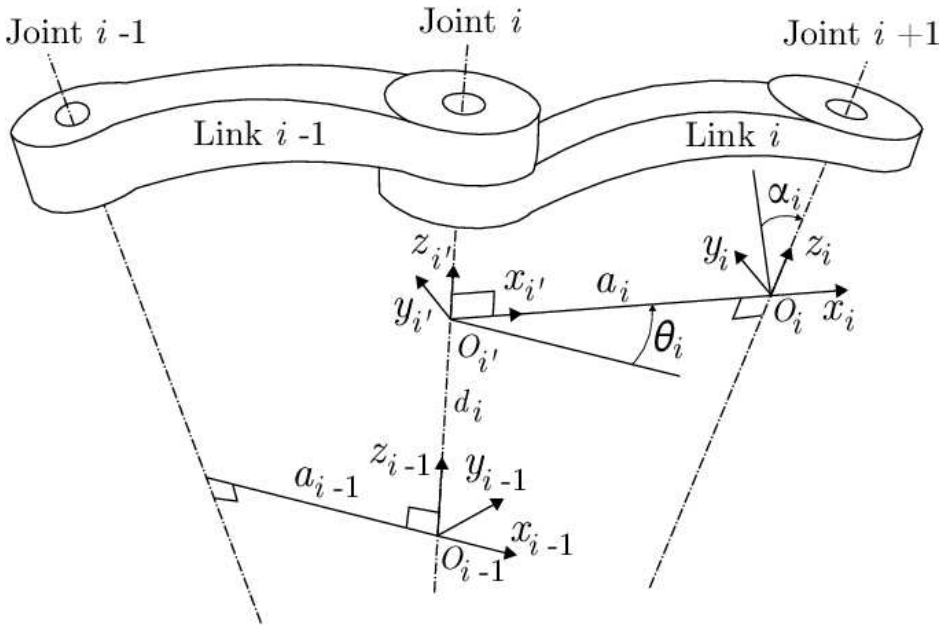


**Figura 2.1** *Parametri di Denavit-Hartenberg [18].*

È importante notare che se il giunto i è rotoidale, la rotazione relativa $\theta_i$ cambia durante il moto del giunto, mentre gli altri parametri rimangono fissi; se invece il giunto è prismatico il disassamento $d_i$ si modifica durante la movimentazione mentre gli altri valori non subiscono variazioni. Quindi in entrambi i casi uno dei due parametri varia a causa del moto (denominato “variabile di giunto”, spesso indicata con “$q_i$”) mentre i restanti tre parametri rimangono costanti, definendo la geometria del membro e del suo accoppiamento.

Di seguito sono riportate graficamente le terne locali posizionate nel FANUC CRX-10iA/L (Figura 2.2) nonché la tabella che riporta i parametri di Denavit-Hartenberg dello stesso (Tabella 2.1).

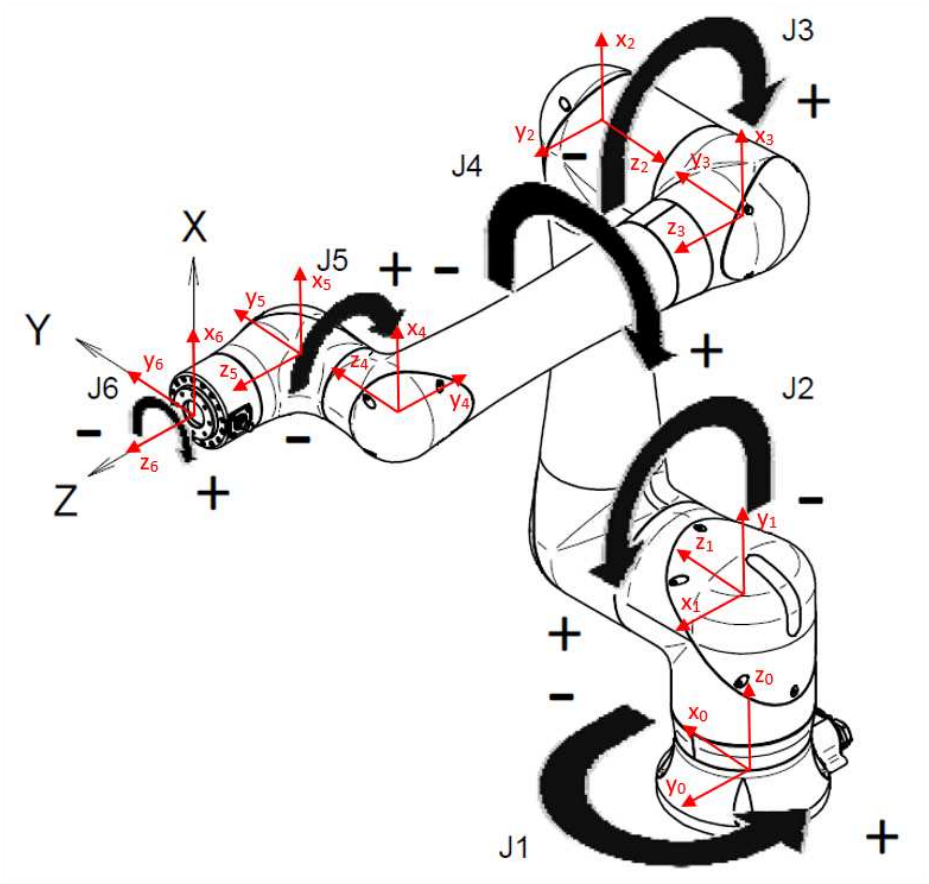


**Figura 2.2** *Rappresentazione grafica delle terne solidali ad ogni membro del FANUC CRX-10iA/L.*

| i | $\alpha_i$ [°] | $a_i$ [m] | $d_i$ [m] | $\theta_i$ [°] |
|---|---|---|---|---|
| 1 | 90 | 0 | 0.2503 | q1 |
| 2 | -180 | 0.71 | 0.2604 | q2 |
| 3 | -90 | 0 | 0.2604 | q3 |
| 4 | -90 | 0 | 0.540 | q4 |
| 5 | 90 | 0 | 0.150 | q5 |
| 6 | 0 | 0 | 0.160 | q6 |

**Tabella 2.1** *Parametri di Denavit-Hartenberg del robot FANUC CRX-10iA/L.*

A partire dai parametri precedentemente definiti è quindi possibile esprimere la trasformazione di coordinate tra i sistemi di riferimento *{i}* e *{i-1}* attraverso il calcolo della matrice di rototraslazione omogenea, ovvero è possibile esprimere le coordinate della terna *{i}* rispetto alla terna *{i-1}*, operazione utile soprattutto per descrivere la posizione e l'orientamento del membro terminale rispetto alla base. La struttura tipica delle matrici di trasformazione omogenee è quella riportata di seguito:

$$ {}^{i-1}_{i}T = \begin{bmatrix} & {}^{i-1}_{i}R & & {}^{i-1}O_{i} \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (2.1) $$

dove ${}^{i-1}_{i}R$ è la matrice di rotazione della terna i-esima che rappresenta l'orientamento della stessa rispetto alla precedente, e ${}^{i-1}O_{i}$ descrive la posizione dell'origine del sistema di coordinate *{i}* rispetto alla terna *{i-1}*, entrambe funzioni della variabile di giunto.

Per ricavare la matrice di trasformazione omogenea è opportuno seguire i passi successivi:

- si introduce un nuovo sistema di coordinate, allineato con la terna *{i-1}*;
- si trasla la nuova terna di un quantitativo corrispondente a $d_i$ lungo l'asse $z_{i-1}$, operazione descritta dalla seguente matrice di trasformazione omogenea:

$$ {}^{i-1}_{int}T = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & 0 \\ \sin\theta_i & \cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.2} $$

- si trasla ulteriormente il sistema di coordinate di una distanza $a_i$ lungo l'asse *x* della terna stessa, operazione descritta dalla seguente matrice di trasformazione omogenea:

$$ {}^{int}_{i}T = \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & \cos\alpha_i & -\sin\alpha_i & 0 \\ 0 & \sin\alpha_i & \cos\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.3} $$

- si ottiene quindi la matrice di trasformazione omogenea risultante attraverso la seguente moltiplicazione:

$$ {}^{i-1}_{i}T = {}^{i-1}_{int}T\,{}^{int}_{i}T = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.4} $$

dove la matrice

$$ \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 \\ \sin\theta_i & \cos\theta_i & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.5} $$

rappresenta la rotazione attorno all'asse $z_{i-1}$ dell'angolo $\theta_i$ della terna intermedia rispetto alla *{i-1}*, il vettore

$$\begin{bmatrix} 0 \\ 0 \\ d_i \end{bmatrix} \tag{2.6}$$

descrive la traslazione lungo l'asse $z_{i-1}$ della distanza $d_i$ della stessa terna relativamente a *{i-1}*, la matrice

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha_i & -\sin\alpha_i \\ 0 & \sin\alpha_i & \cos\alpha_i \end{bmatrix} \tag{2.7}$$

rappresenta la rotazione attorno all'asse $x_i$ dell'angolo $\alpha_i$ del sistema di coordinate *{i}* rispetto alla terna intermedia e il vettore

$$\begin{bmatrix} a_i \\ 0 \\ 0 \end{bmatrix} \tag{2.8}$$

descrive la traslazione lungo l'asse $x_i$ della distanza $a_i$ della terna i-esima rispetto alla precedente.

## 2.2 Analisi cinematica

Come illustrato precedentemente, i meccanismi robotici sono sistemi di corpi rigidi collegati attraverso dei giunti. La posizione e l'orientamento di un corpo

rigido nello spazio sono indicati congiuntamente attraverso il termine “posa”. L’obiettivo dell’analisi cinematica di un robot è quindi la descrizione della posa, della velocità, dell’accelerazione e delle derivate di ordine superiore della posa (rispetto al tempo o a qualsiasi altra variabile) dei corpi che costituiscono il meccanismo, senza prendere in considerazione forze e/o coppie che inducono il moto dello stesso. Per ricavare queste informazioni è necessario dotare ogni membro *i* del meccanismo preso in esame di un sistema di riferimento i-esimo ad esso solidale, così da poter esprimere la posa di un corpo come la posa di un sistema di coordinate solidale ad un membro rispetto a quello solidale ad un altro membro [18].

Il modello cinematico di un robot può essere utilizzato per risolvere due problemi distinti:

- analisi cinematica diretta: consiste nel determinare la posa del membro terminale del sistema (o di qualsiasi altro punto della struttura) in funzione degli spostamenti noti dei giunti. Per catene cinematiche aperte il problema diretto garantisce un assetto finale univoco;
- analisi cinematica inversa: consiste nel determinare l’insieme di spostamenti dei giunti che portano il terminale del meccanismo in un punto assegnato, con orientamento prefissato. È un problema non lineare che può portare a soluzioni multiple o duplicate ed in generale la soluzione non è calcolabile in maniera esplicita se non in casi particolari.

### 2.2.1 Cinematica diretta

Il problema cinematico diretto consiste nel determinare la posizione e l’orientamento del membro terminale del sistema meccanico (o di qualunque altro punto della struttura) in funzione degli spostamenti (noti) dei giunti:

$$T = {}^{0}_{1}T(q_1)\,{}^{1}_{2}T(q_2)\ldots{}^{i-1}_{i}T(q_i)\ldots{}^{n-1}_{n}T(q_n) = {}^{0}_{n}T(q_1,\ldots,q_n) \quad (2.9)$$

Il problema è risolto calcolando direttamente la formula precedente, che diventa un prodotto di matrici costanti; pertanto esso, che è già posto in forma esplicita, è facilmente risolubile e fornisce sempre una sola soluzione nel caso di strutture seriali [18].

In questa trattazione si è scelto di implementare solamente la cinematica diretta a discapito di quella inversa che, come illustrato precedentemente, si presenta difficoltosa nella risoluzione in forma esplicita.

La risoluzione della cinematica diretta relativa al robot FANUC CRX-10iA/L è riportata nell'appendice A.3.

## 2.3 Analisi dinamica

L'implementazione del modello dinamico di un manipolatore gioca un ruolo di estrema importanza in relazione ai problemi di simulazione del moto, di analisi di strutture di manipolazione e di determinazione di algoritmi di controllo: la simulazione del moto di un manipolatore consente di provare strategie di controllo e tecniche di pianificazione di traiettoria senza la necessità di riferirsi a una struttura di manipolazione fisicamente disponibile, l'analisi del modello dinamico può essere di aiuto nel progetto meccanico di prototipi e la conoscenza di forze e coppie richieste per l'esecuzione di movimenti tipici fornisce informazioni utili al progetto di giunti e trasmissioni e alla scelta di attuatori.

Lo studio del comportamento dinamico dei manipolatori porta allo sviluppo di una serie di equazioni differenziali (equazioni dinamiche ottenute in funzione del minimo numero di coordinate indipendenti), o algebrico-differenziali (equazioni dinamiche ottenute in funzione di un insieme di coordinate dipendenti), chiamate equazioni del moto, che consentono di modellare il comportamento dinamico del sistema quando esso è sottoposto a sollecitazioni esterne e ad azioni sui giunti. Tali equazioni possono essere ricavate mediante approcci differenti a seconda delle caratteristiche del sistema stesso, in base alla finalità dello studio e al tipo di coordinate utilizzate: tra le metodologie più utilizzate possono essere citate Newton-Eulero, il PLV (Principio dei Lavori Virtuali) e Lagrange. La formulazione di Newton-Eulero prevede la scrittura delle equazioni cardinali della dinamica per ogni membro e la risoluzione del sistema così impostato. Questa ha il vantaggio di garantire un'immediata comprensione del significato fisico dei passaggi intermedi e consente di ricavare anche le forze di reazione interne; queste ultime tuttavia devono essere poi eliminate attraverso manipolazioni algebriche aggiuntive per ricavare le equazioni dinamiche in una forma esplicita. Il principio dei lavori virtuali ha invece il vantaggio di non far comparire le reazioni vincolari nella formulazione, conducendo quindi ad equazioni compatte ed efficienti. Infine la formulazione Lagrangiana utilizza considerazioni di tipo energetico, ovvero nella definizione delle equazioni si ricorre allo studio dell'energia cinetica e potenziale del sistema, fornendo un modello in forma compatta utilizzando un insieme di coordinate generalizzate.

Il modello dinamico del robot può essere utilizzato per risolvere due problemi distinti:

- analisi dinamica inversa: consiste nell'assegnare la traiettoria del robot (posizioni, velocità, accelerazioni ed eventuali forze/momenti esterni sul

terminale) e nel determinare le coppie richieste nella realizzazione del moto. Questa tipologia di problema è di risoluzione immediata in quanto consiste nella valutazione di un'espressione nota spesso in forma chiusa;

- analisi dinamica diretta: consiste nel determinare come si muove il manipolatore quando è sottoposto all'azione di forze o coppie sui giunti assegnate (ed eventuali forze/momenti esterni). Questa analisi porta alla risoluzione di sistemi di equazioni differenziali non lineari ed accoppiate, che conducono spesso a problemi di tipo numerico.

### 2.3.1 Formulazione di Newton-Eulero

Le rappresentazioni scelte per realizzare il modello dinamico del robot FANUC CRX-10iA/L sono il metodo di Newton-Eulero e il metodo di Lagrange.

Il metodo di Newton-Eulero prevede che per ogni membro del cinematismo vengano risolte le equazioni cardinali della dinamica. Per fare ciò è utile rappresentare il diagramma di corpo libero di un generico membro *i*, riportato nella Figura 2.3, così da poter visualizzare il sistema di forze e coppie agenti sullo stesso [19].

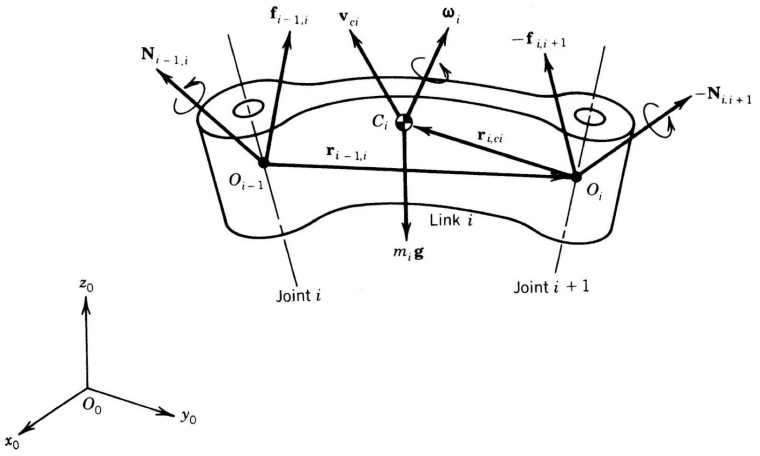


**Figura 2.3** *Diagramma di corpo libero di un generico membro [19].*

Nella figura precedente:

- il vettore coppia $N_{i-1,i}$ e il vettore forza $f_{i-1,i}$ rappresentano le forze interne generate dall'accoppiamento con il membro precedente e analogamente il vettore coppia $N_{i,i+1}$ e il vettore forza $f_{i,i+1}$ rappresentano le forze interne generate dall'accoppiamento con il membro successivo;
- $C_i$ è il baricentro del corpo, caratterizzato dal vettore velocità $v_{ci}$ e dal vettore posizione $r_{i,ci}$ rispetto alla terna centrata in $O_i$;
- $r_{i-1,i}$ rappresenta il vettore posizione della terna *{i}* rispetto alla terna *{i-1}*;
- $\omega_i$ è il vettore velocità angolare del membro i-esimo, di massa $m_i$.

La prima equazione cardinale della dinamica, che rappresenta l'equilibrio alla traslazione, è:

$$f_{i-1,i} - f_{i,i+1} + m_i \boldsymbol{g} = m_i \dot{\mathbf{v}}_{C_i} \tag{2.10}$$

Questa è riconducibile a tre equazioni scalari, nelle quali $\dot{\mathbf{v}}_{C_i}$ rappresenta l'accelerazione del baricentro.

La seconda equazione cardinale della dinamica, ovvero l'equilibrio dinamico alla rotazione del membro i-esimo, è la seguente:

$$N_{i-1,i} - N_{i,i+1} - \left( r_{i-1,i} + r_{i,C_i} \right) \times f_{i-1,i} + \left( -r_{i,C_i} \right) \times \left( -f_{i,i+1} \right) = I_i \dot{\boldsymbol{\omega}}_i + \boldsymbol{\omega}_{\mathbf{i}} \times I_i \boldsymbol{\omega}_{\mathbf{i}} \tag{2.11}$$

dove $\dot{\boldsymbol{\omega}}_{\mathbf{i}}$ rappresenta l'accelerazione angolare del corpo i-esimo e $I_i$ la matrice di inerzia baricentrica dello stesso.

Il secondo membro dell'equazione è ottenuto calcolando la derivata del momento angolare del corpo, ovvero

$$\frac{d}{dt}\left( I_i \boldsymbol{\omega}_i \right) = I_i \dot{\boldsymbol{\omega}}_i + \boldsymbol{\omega}_i \wedge \left( I_i \boldsymbol{\omega}_i \right) \tag{2.12}$$

A seguito di quanto illustrato precedentemente è possibile affermare che il comportamento dinamico di ogni singolo membro del sistema meccanico è descritto dalle due equazioni del moto sopra riportate, in cui tuttavia risultano incognite le reazioni vincolari. Il modello dinamico di tutto il manipolatore è invece descritto dalle 2n equazioni vettoriali del moto ottenute considerando tutti i membri della catena cinematica.

Tuttavia le equazioni di Newton-Eulero appena derivate non sono in forma appropriata per l'utilizzo nell'analisi dinamica e nel progetto dei controlli in quanto non descrivono esplicitamente una relazione ingresso/uscita tra le variabili interessate. Per prima cosa è bene notare che le equazioni coinvolgono le forze $f_{i-1,i}$ ed i momenti $N_{i-1,i}$, i quali includono sia le forze/coppie ai giunti, sia le reazioni vincolari interne che si trasmettono attraverso i giunti stessi. Per derivare relazioni dinamiche esplicite ingresso/uscita occorre pertanto separare le coppie d'ingresso del giunto dalle forze/coppie esercitate dal vincolo. Inoltre tali equazioni sono espresse nei termini delle accelerazioni angolari dei membri e delle accelerazioni lineari dei loro baricentri: poiché tali grandezze sono fra loro legate da relazioni cinematiche è necessario esprimerle in funzione di variabili indipendenti, quali gli spostamenti lineari o angolari dei giunti. In altri termini occorre imporre la congruenza degli spostamenti introducendo le equazioni cinematiche di posizione, velocità ed accelerazione.

In questo modo si ottengono infine equazioni dinamiche in forma chiusa nei termini delle coppie/forze $\boldsymbol{\tau}$ sui giunti e dei relativi spostamenti **q**.

La modellazione del robot FANUC CRX-10iA/L attraverso le equazioni di Newton-Eulero è riportata nell'appendice A.4.

## 2.3.2 Formulazione di Lagrange

Il secondo metodo utilizzato per la realizzazione del modello dinamico del robot preso in esame è la formulazione di Lagrange. Sono state implementate entrambe le modellazioni dinamiche (Newton-Eulero e Lagrange) in quanto la seconda introduce vantaggi soprattutto in termini di efficienza computazionale, riducendo notevolmente i tempi di calcolo nel processamento delle equazioni ricavate.

Il metodo di Newton-Eulero, basato su formule dinamiche elementari e sull'analisi di forze e momenti di vincolo agenti tra i membri, viene definito come approccio alla dinamica "basato sull'equilibrio delle forze" mentre la formulazione Lagrangiana viene descritta come approccio " basato sull'energia" [19].

Per ottenere le equazioni del moto del manipolatore questo metodo prevede il calcolo della Lagrangiana per ogni membro, così definita:

$$L(q,\dot{q}) = T(q,\dot{q}) - U(q) = \sum_{i=1}^{n} \left( T_i(q,\dot{q}) - U_i(q) \right) = \sum_{i=1}^{n} L_i(q,\dot{q}) \quad (2.13)$$

dove $T$ e $U$ sono rispettivamente l'energia cinetica e l'energia potenziale del meccanismo e $q$ e $\dot{q}$ sono rispettivamente i vettori posizione e velocità dei giunti.

Le equazioni dinamiche del moto possono quindi essere ricavate utilizzando l'equazione di Lagrange per ogni coordinata generalizzata $q_1, \ldots, q_n$ (dove $n$ è il numero di gradi di libertà del sistema):

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = \tau_i \quad (2.14)$$

dove $\tau_i$ è la coppia di attuazione del giunto corrispondente [18].

L'energia cinetica di ogni corpo può essere ottenuta attraverso la seguente

relazione:

$$T_i = \frac{1}{2} m_i \boldsymbol{v}_{C_i}{}' \boldsymbol{v}_{C_i} + \frac{1}{2} \boldsymbol{\omega}_i{}' I_i \boldsymbol{\omega}_i \tag{2.15}$$

dove $m_i$ è la massa del membro, $\boldsymbol{v}_{\mathrm{C_i}}$ è la velocità del baricentro, $\boldsymbol{\omega}_{\mathrm{i}}$ è la velocità angolare del corpo e $I_i$ è la matrice d'inerzia dello stesso. È bene notare che il primo addendo della relazione precedente rappresenta l'energia cinetica di traslazione mentre il secondo l'energia cinetica di rotazione.
L'energia potenziale di ogni corpo può essere invece espressa come:

$$U_i = m_i \boldsymbol{g}\, {}^0\boldsymbol{r}_{\mathrm{C_i}} \tag{2.16}$$

dove $\boldsymbol{g}$ è il vettore gravità e ${}^0\boldsymbol{r}_{C_i}$ è il vettore posizione del baricentro del membro i-esimo espresso rispetto alla terna di base {0} [19].
La modellazione del robot FANUC CRX-10iA/L attraverso la formulazione Lagrangiana è riportata nell'appendice A.5.

# Capitolo 3

# Verifica dei modelli

Al fine di verificare la correttezza dei modelli illustrati nel capitolo precedente, è necessario implementare una procedura di verifica che consenta di ottenere un confronto immediato tra i risultati estrapolati da questi ultimi e quelli ottenuti tramite simulazioni realizzate su un software apposito. Per fare questo la procedura di maggiore impiego è la tecnica della co-simulazione basata sulla collaborazione dei software MSC ADAMS e Simulink, strumento utile in molti ambiti quali ad esempio il miglioramento del ciclo di sviluppo prodotto, la progettazione di un sistema meccatronico, l'implementazione di modelli matematici o fisici o di sistemi di controllo, ecc [20]. In particolare questa tecnica consta di tre fasi rilevanti, rappresentate nella Figura 3.1:

1. progettare e assemblare un sistema meccanico in un software CAD;
2. importare l'assieme nell'ambiente di simulazione ADAMS;
3. collegare il modello ADAMS con l'ambiente Simulink attraverso le variabili di stato.

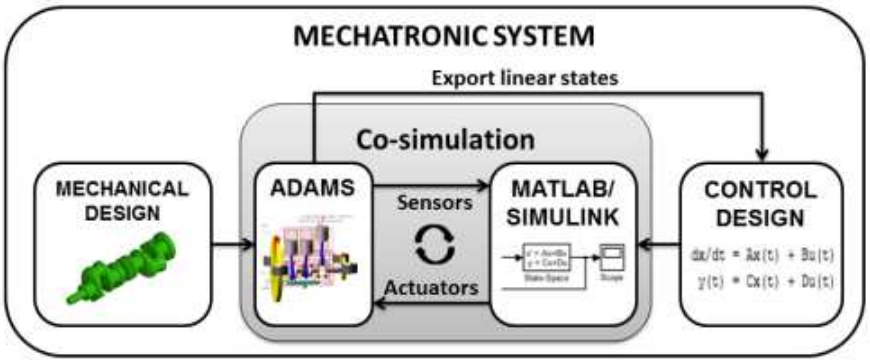


**Figura 3.1** *Co-simulazione di un sistema meccatronico [20].*

Attraverso questa tecnica è stato possibile realizzare il modello multibody del robot CRX-10iA/L nonché le successive simulazioni per la verifica dei modelli sopra citati e quindi il raggiungimento del fine ultimo di questa trattazione.

## 3.1 Realizzazione del modello multibody

Il primo passo per l'implementazione di una procedura di verifica dei modelli illustrati consiste quindi nell'importazione nell'ambiente di simulazione ADAMS del modello CAD del robot CRX-10iA/L, ottenuto in questo caso attraverso il portale dell'azienda produttrice. Il modello così importato è rappresentato nella Figura 3.2.

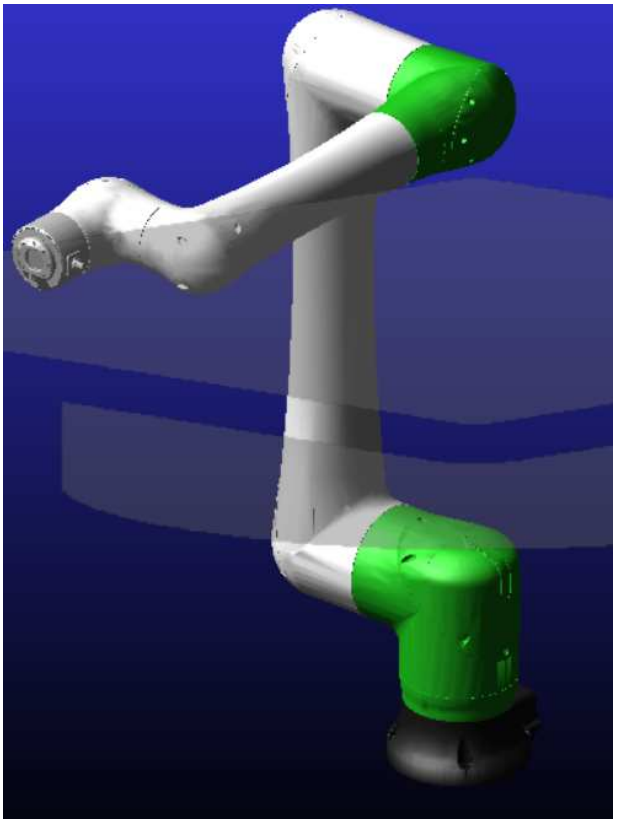

**Figura 3.2** *Importazione del modello CAD del robot CRX-10iA/L in ADAMS.*

I corpi che costituiscono il manipolatore sono così nominati (a partire dalla base):

- J1BASE_UNIT
- J2BASE_UNIT
- J2ARM_UNIT_LONG
- J3CASING_UNIT
- J3ARM_UNIT
- J5CASING_UNIT
- J6FLANGE_CAM_UNIT

Successivamente all'importazione del modello CAD, è necessario introdurre in ADAMS gli elementi necessari alla realizzazione delle simulazioni e all'ottenimento delle grandezze desiderate, quindi i giunti (*Joints*), le leggi del moto dei giunti (*Motions*), i sistemi di riferimento secondo la convenzione di Denavit-Hartenberg e i sistemi di riferimento centrati nei centri di massa dei membri (*Markers*), le proprietà di massa di questi ultimi (definite attraverso la modalità *Geometry and Density* del software, ipotizzando una densità di 1000 $kg/m^3$) e le variabili di stato (*State Variables*). Per quanto riguarda i giunti, questi sono stati inseriti nell'ambiente di simulazione attraverso la modalità *Revolute joint* tra le coppie di membri adiacenti, ad eccezione del primo membro il quale è stato fissato a telaio (*ground*) attraverso un *Fixed joint* (i giunti inseriti sono evidenziati nella Figura 3.3); di seguito sono stati introdotti i *Motions* attraverso il comando *Rotational Joint Motion* nei giunti rotativi precedentemente realizzati, evidenziati nella Figura 3.4.

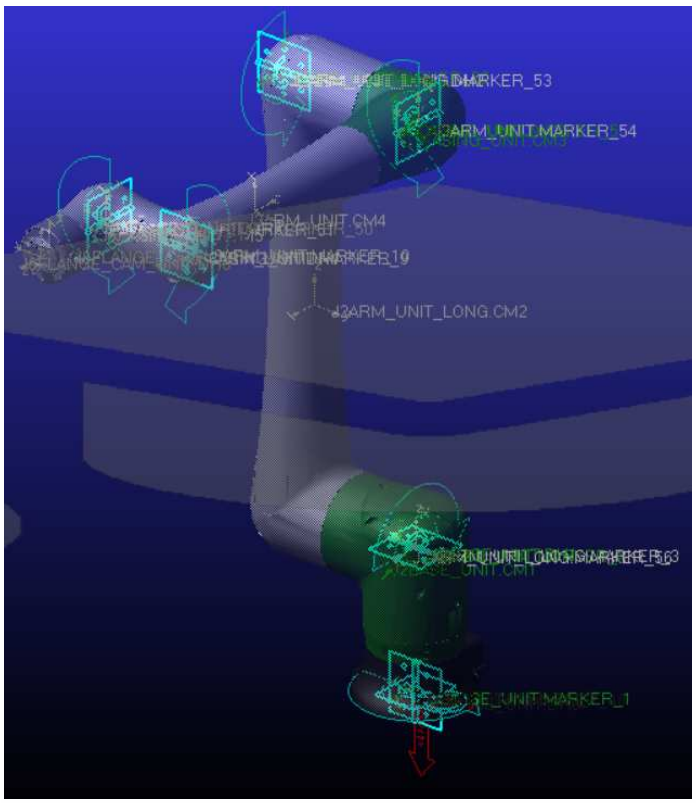


**Figura 3.3** *Giunti inseriti nel modello ADAMS.*

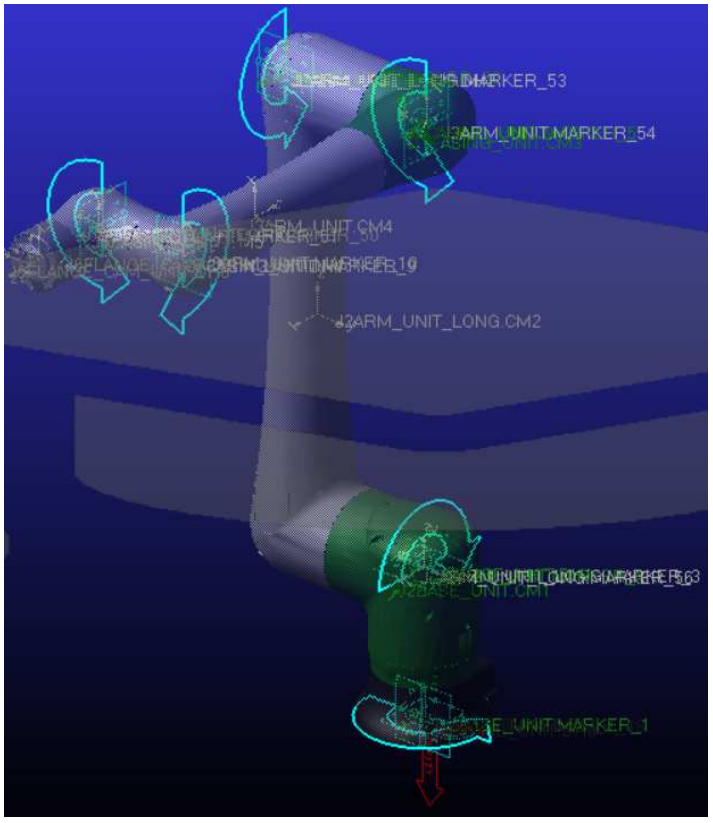


**Figura 3.4** *Motions inseriti nel modello ADAMS.*

Per quanto riguarda i sistemi di riferimento, sono state introdotte le terne secondo la convenzione di Denavit-Hartenberg come riportato al capitolo precedente e le terne centrate nei centri di massa dei membri costituenti il robot, individuati attraverso il modello CAD dello stesso, le quali sono poi state selezionate come *Center of Mass Marker* e *Inertia Reference Marker* nelle impostazioni di ogni corpo.

Infine, per consentire il collegamento tra i software ADAMS e Simulink, sono state create le seguenti variabili di stato:

- INPUT: di valore nullo, necessaria ai fini dell'esecuzione della co-simulazione;
- SPOST_1, SPOST_2, SPOST_3, SPOST_4, SPOST_5, SPOST_6: spostamenti angolari attorno all'asse z (*Angle about z*) tra i membri adiacenti;
- VEL_1, VEL_2, VEL_3, VEL_4, VEL_5, VEL_6: velocità angolari attorno all'asse z (*Angular Velocity about z*) tra i membri adiacenti;
- ACC_1, ACC_2, ACC_3, ACC_4, ACC_5, ACC_6: accelerazioni angolari attorno all'asse z (*Angular Acceleration about z*) tra i membri adiacenti;
- COPPIA_1, COPPIA_2, COPPIA_3, COPPIA_4, COPPIA_5, COPPIA_6: componenti lungo l'asse z delle coppie calcolate rispetto alle terne locali di Denavit-Hartenberg.

Dopo aver introdotto i precedenti elementi, è possibile procedere con le simulazioni desiderate, impostando quindi le leggi del moto dei singoli giunti in *Motions*.

## 3.2 Verifica del modello cinematico

Essendo l'obiettivo dell'analisi cinematica di un robot la descrizione della posa, della velocità, dell'accelerazione e delle derivate di ordine superiore della posa (rispetto al tempo o a qualsiasi altra variabile) dei corpi che costituiscono il meccanismo, senza prendere in considerazione forze e/o coppie che inducono il moto dello stesso, si è scelto di verificare la correttezza del corrispondente modello semplicemente confrontando posizione e orientamento del membro terminale nei modelli in MATLAB e ADAMS. Di seguito sono riportate le configurazioni del robot ottenute introducendo a titolo esemplificativo uno spostamento angolare di 45° nel secondo giunto nel modello cinematico implementato in MATLAB (Figura 3.5) e nel modello multibody in ADAMS (Figura 3.6).

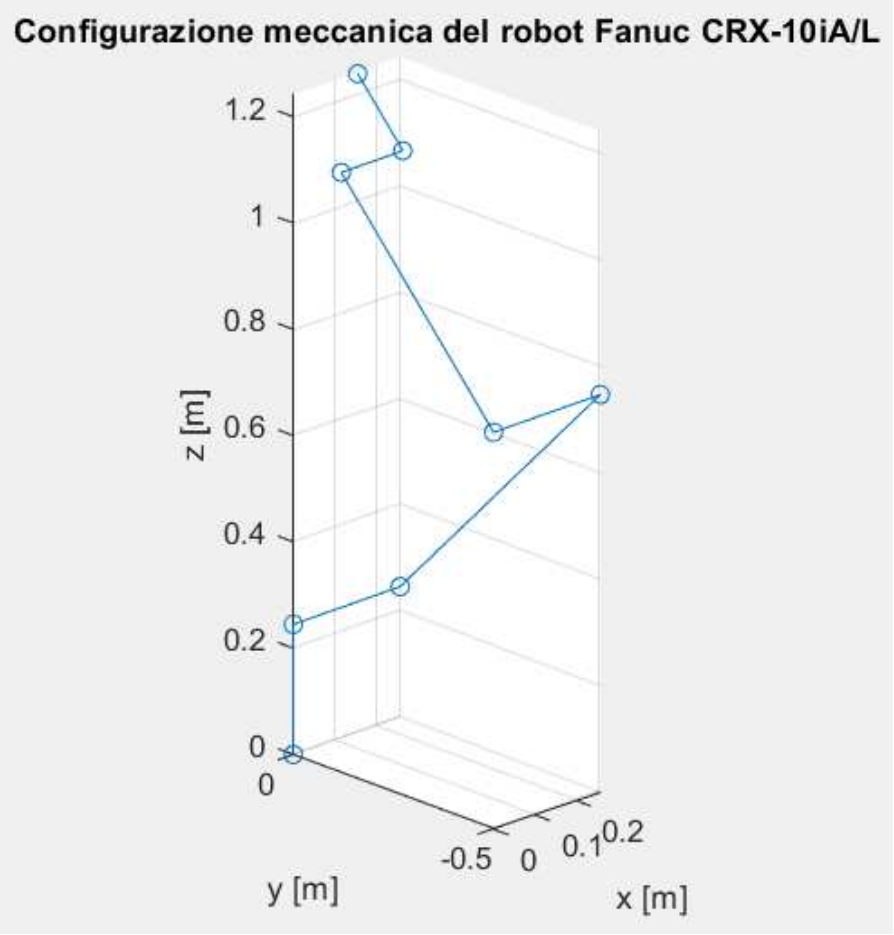


**Figura 3.5** *Configurazione del robot nel modello cinematico implementato in MATLAB.*

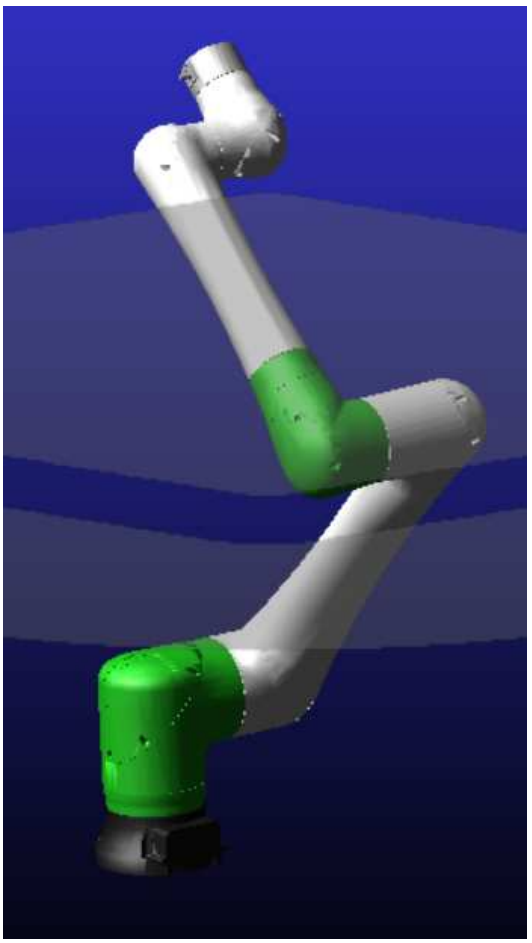

**Figura 3.6** *Configurazione del robot nel modello multibody implementato in ADAMS.*

Confrontando, sia visivamente che numericamente, posizione e orientamento del membro terminale, possono essere visualizzati gli stessi valori numerici, i quali dimostrano la correttezza del modello cinematico implementato riportato nell' appendice A.3.

## 3.3 Verifica dei modelli dinamici

Relativamente alla verifica dei modelli dinamici implementati in MATLAB facendo uso delle formulazioni di Newton-Eulero e Lagrange, si è scelto di verificare la correttezza di questi ultimi attraverso l'utilizzo combinato dei software ADAMS e Simulink.

Per prima cosa è necessario scegliere la movimentazione da far eseguire al robot, possibilmente non casuale ma riconducibile ad una traiettoria tipica in un'applicazione industriale, come ad esempio *pick and place*. Per fare questo è stato realizzato un *General Point Motion* (disattivando gli altri *Motions* precedentemente creati) tra l'*end-effector* e il *ground*, in modo tale da poter scegliere la movimentazione da far eseguire al membro terminale. Di seguito, nella Figura 3.7, è riportato il dettaglio della traiettoria scelta, in cui è doveroso evidenziare la presenza della funzione STEP, selezionata in quanto legge ad elevata continuità e differenziabilità.

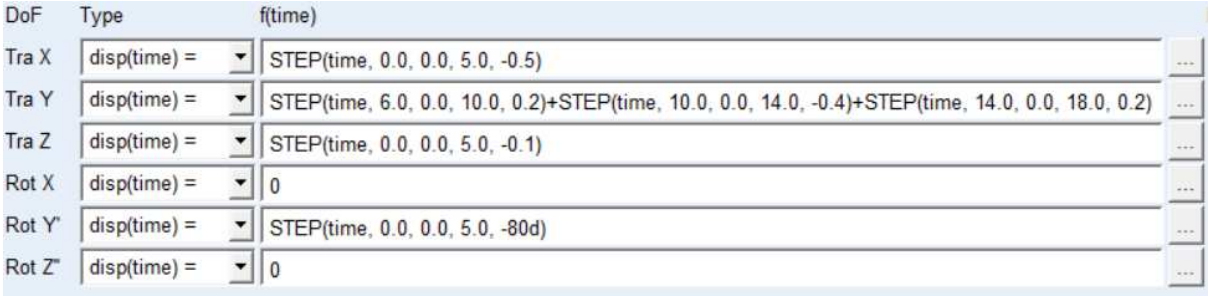


**Figura 3.7** *Movimentazione scelta per il robot.*

Attraverso l'assegnazione della precedente movimentazione al robot è stata realizzata la cinematica inversa del meccanismo, ovvero è possibile visualizzare all'interno del *Postprocessor* le rotazioni sui giunti necessarie all'esecuzione della traiettoria. Quindi, per poter applicare queste ultime ai *Motions* propri di ogni giunto, è necessario scaricare i dati relativi alla movimentazione su delle curve, in questo caso *spline*, le quali creano quindi delle funzioni continue da un insieme di punti dati. Per fare questo è sufficiente selezionare dal postprocessatore le curve relative alle variabili di stato SPOST_1, SPOST_2, SPOST_3, SPOST_4, SPOST_5 e SPOST_6 e successivamente eseguire il comando apposito per la generazione delle *spline*, assegnando il nome

opportuno alle stesse, in questo caso q1, q2, q3, q4, q5 e q6. A questo punto, disattivando il *General Point Motion* precedentemente creato, è possibile assegnare ai *Motions* relativi ai sei giunti (nuovamente attivati) le *spline* generate, attraverso la funzione CUBSPL (*Cubic Fitting Method*), la quale consente di effettuare un'interpolazione cubica dei dati salvati nelle *spline*. Infine è quindi possibile avviare la simulazione, potendo così visualizzare la movimentazione del robot scelta per la verifica dei modelli dinamici implementati.

Per poter realizzare la co-simulazione tra i software ADAMS e Simulink è necessario esportare i dati della simulazione completata attraverso il comando *Plant Export*, impostando tra gli *input signal(s)* la variabile INPUT e tra gli *output signal(s)* le rimanenti variabili di stato precedentemente create. Eseguendo così il file MATLAB generato da questa esportazione e successivamente il comando *adams_sys* nella *Command Window* si ottiene il sistema rappresentato nella Figura 3.8, in cui il blocco arancione rappresenta la simulazione ADAMS precedentemente effettuata.

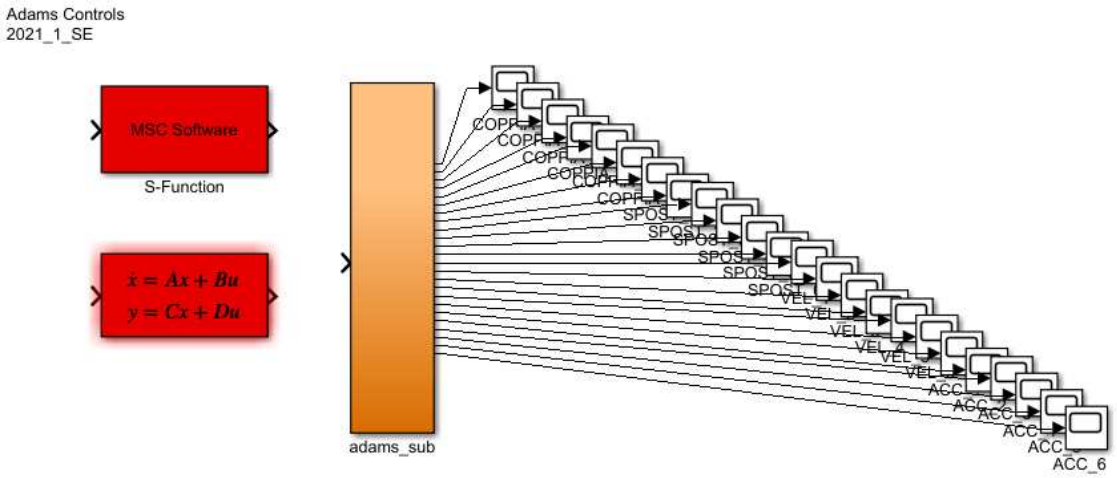


**Figura 3.8** *Esportazione dei dati della simulazione da ADAMS a Simulink.*

### 3.3.1 Verifica del modello dinamico di Newton-Eulero

La verifica del modello dinamico implementato secondo la formulazione di Newton-Eulero è stata effettuata attraverso la realizzazione in Simulink dello schema riportato nella Figura 3.9 in cui, a partire da sinistra, sono rappresentati:

- la simulazione eseguita in ADAMS esportata in Simulink (blocco arancione);
- gli *Scope* che consentono la visualizzazione grafica delle variabili di stato contenenti spostamenti, velocità e accelerazioni angolari;
- la *MATLAB Function* contenente la procedura di confronto tra i risultati ottenuti in ADAMS e quelli ottenuti dal modello implementato in MATLAB, la quale ha come input i valori del punto precedente e come output le coppie calcolate attraverso il modello realizzato in MATLAB (il contenuto della *MATLAB Function* è riportato nell'appendice A.6);
- gli *Scope* che consentono la visualizzazione grafica del confronto tra le coppie sui giunti valutate attraverso la simulazione in ADAMS e quelle ottenute attraverso il modello dinamico implementato in MATLAB.

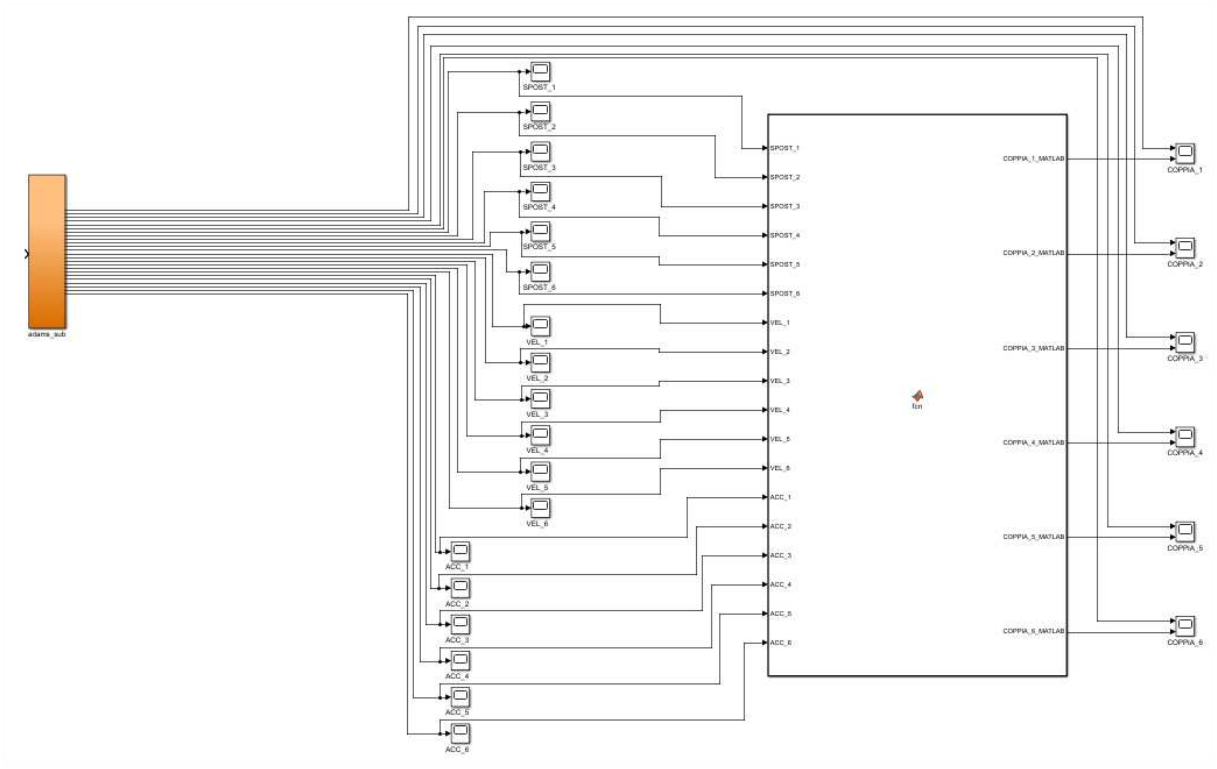

**Figura 3.9** *Schema per la verifica del modello dinamico di Newton-Eulero in Simulink.*

Di seguito, nelle Figure 3.10, 3.11, 3.12, 3.13, 3.14 e 3.15, sono riportati i grafici che rappresentano gli errori tra le misurazioni di coppia ottenute in ADAMS e MATLAB, valutati come differenza in valore assoluto tra le curve ottenute tramite i due software normalizzata rispetto al massimo registrato. Dai valori ridotti di questi ultimi è possibile quindi dedurre la correttezza del modello implementato.

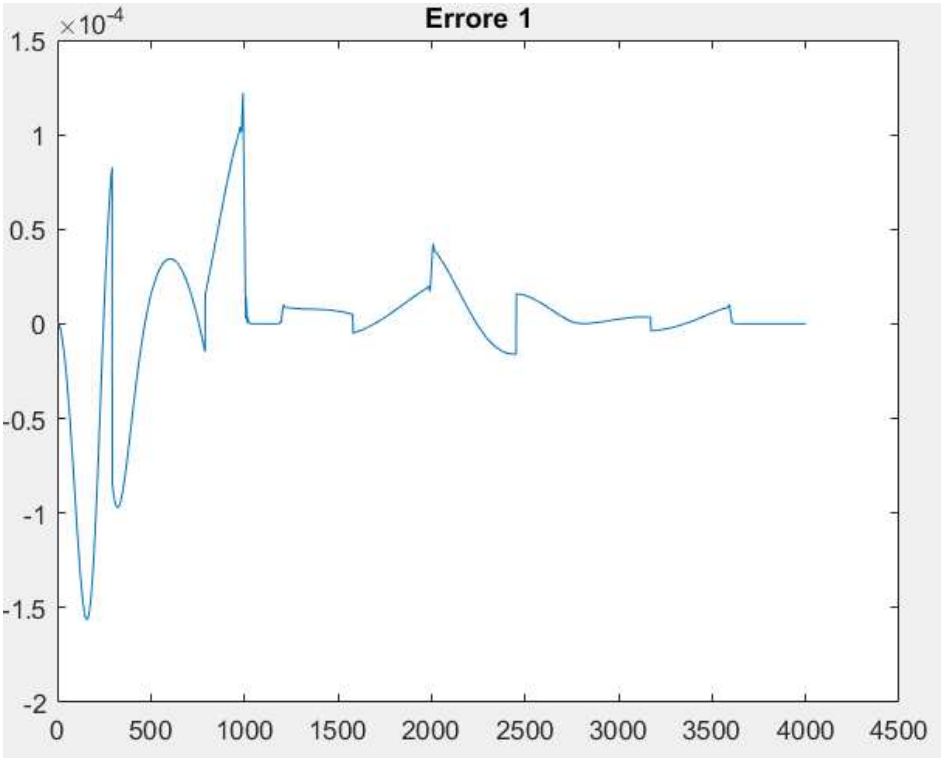


**Figura 3.10** *Errore tra le coppie misurate al giunto 1.*

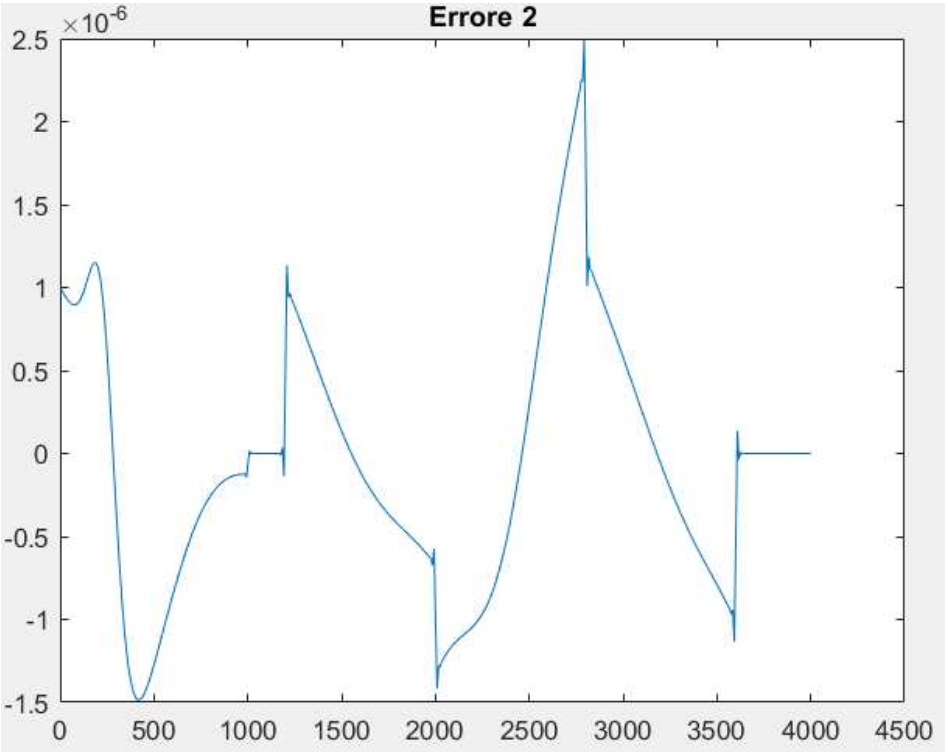


**Figura 3.11** *Errore tra le coppie misurate al giunto 2.*

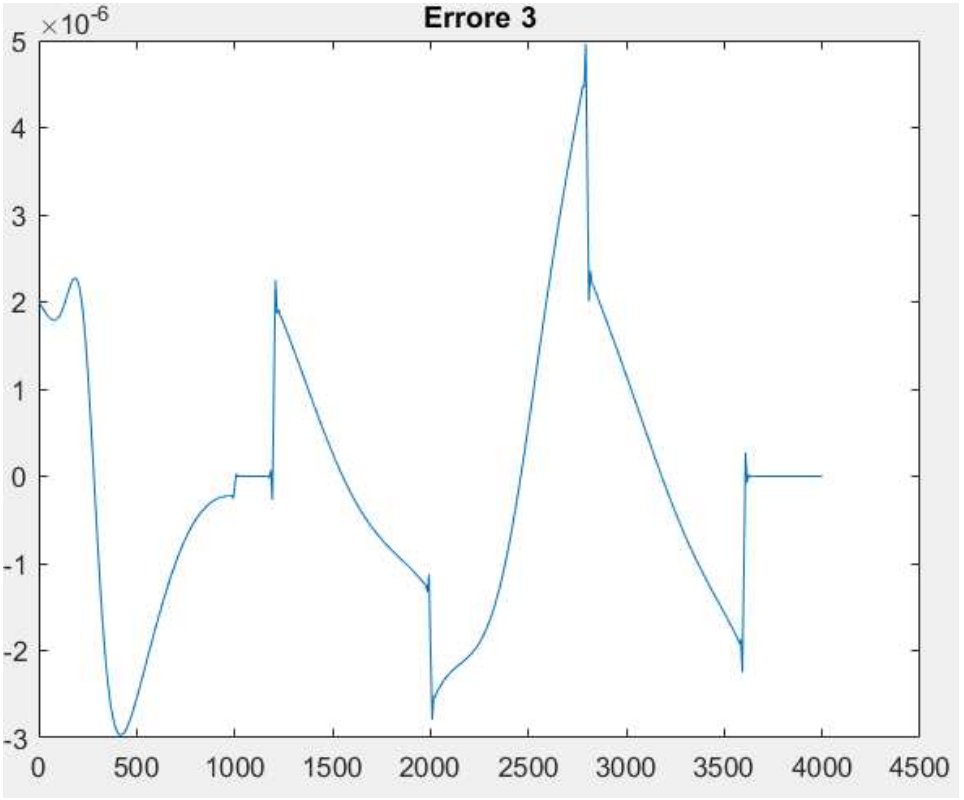


**Figura 3.12** *Errore tra le coppie misurate al giunto 3.*

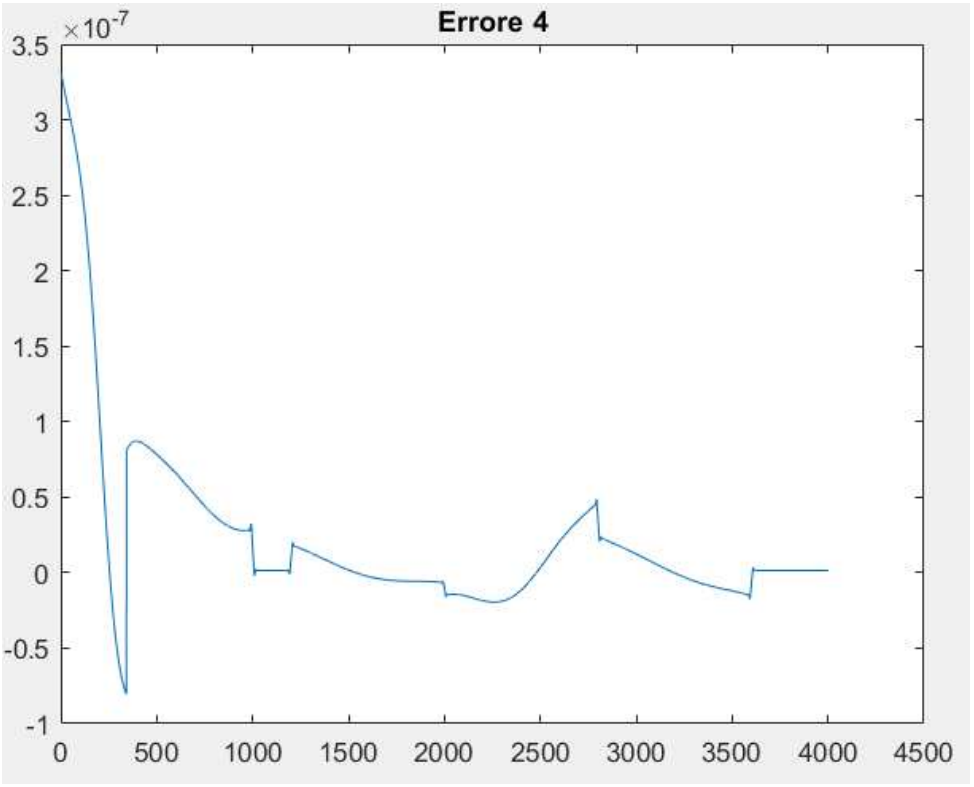


**Figura 3.13** *Errore tra le coppie misurate al giunto 4.*

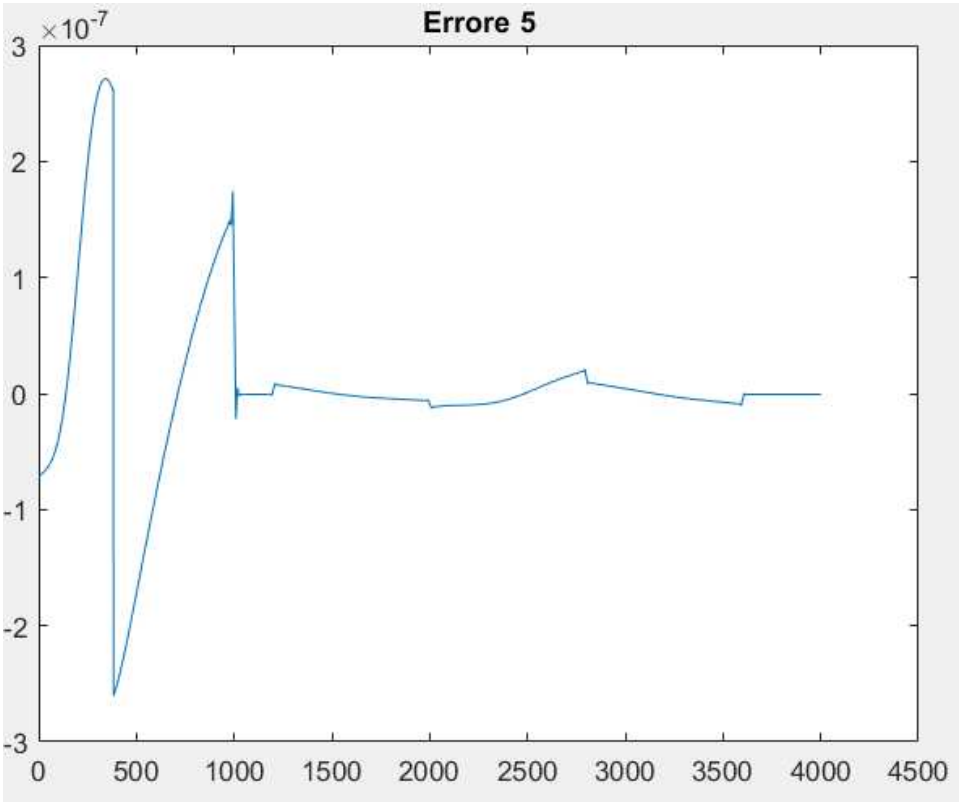


**Figura 3.14** *Errore tra le coppie misurate al giunto 5.*

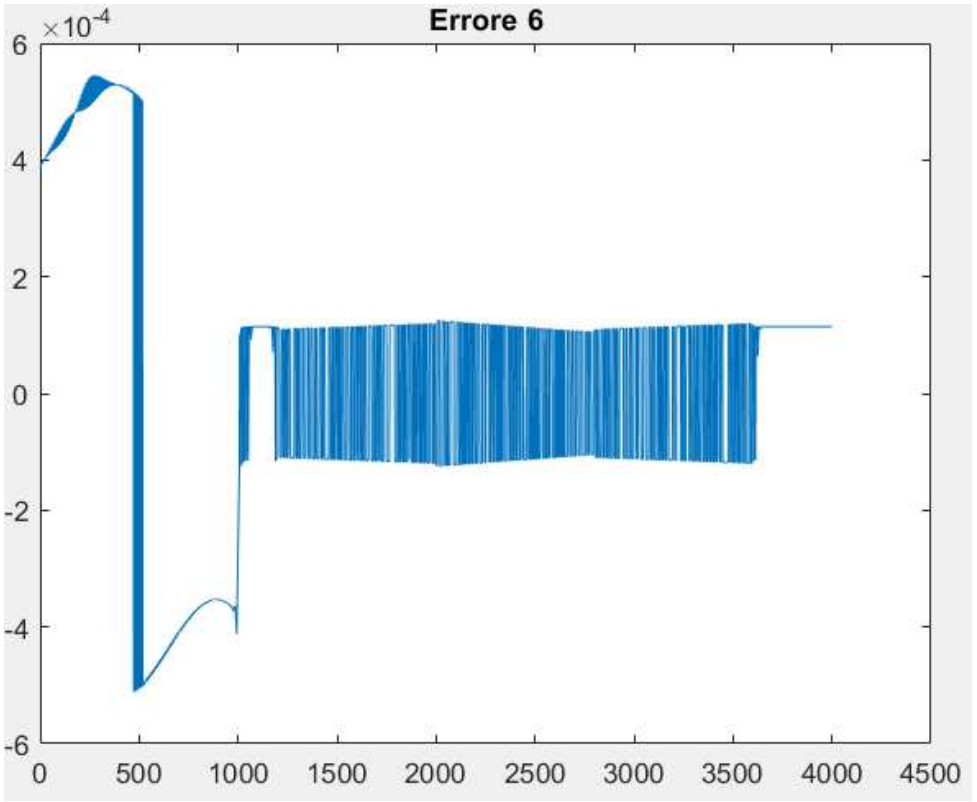


**Figura 3.15** *Errore tra le coppie misurate al giunto 6.*

### 3.3.2 Verifica del modello dinamico di Lagrange

La verifica del modello dinamico implementato attraverso la formulazione di Lagrange è stata effettuata attraverso la scrittura di uno script, riportato nell'appendice A.7, il quale presenta la sostituzione di valori numerici ai valori simbolici introdotti nel modello implementato, estrapolati dal modello multibody realizzato in ADAMS, attraverso l'ausilio dell'esportazione della simulazione in ambiente Simulink. I dati relativi a quest'ultima sono quindi riportati in MATLAB attraverso lo schema Simulink rappresentato nella Figura 3.16, il quale consente il salvataggio dei valori delle variabili di stato *To Workspace*, ovvero all'interno dell'ambiente MATLAB.

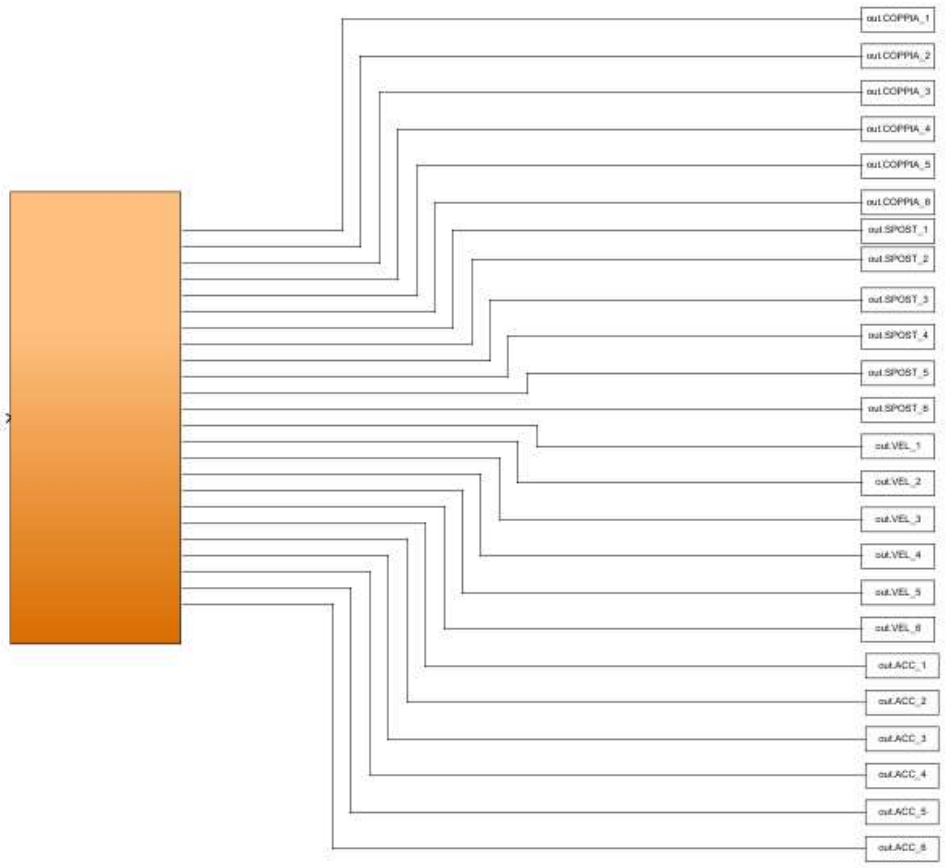


**Figura 3.16** *Schema Simulink per l'esportazione in MATLAB dei dati della simulazione.*

Di seguito, nelle Figure 3.17, 3.18, 3.19, 3.20, 3.21 e 3.22, sono riportati i grafici che rappresentano gli errori tra le misurazioni di coppia ottenute in ADAMS e MATLAB, valutati in maniera analoga a quanto descritto nella verifica del modello di Newton-Eulero. Dai valori ridotti di questi ultimi è possibile quindi dedurre la correttezza del modello implementato.

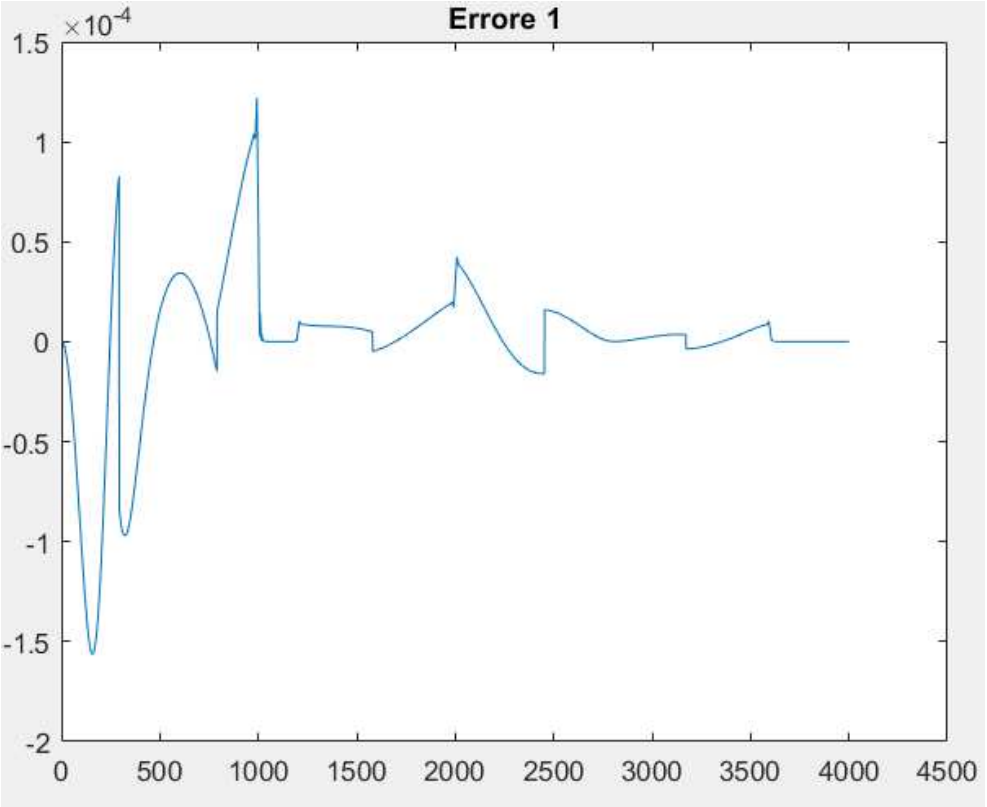


**Figura 3.17** *Errore tra le coppie misurate al giunto 1.*

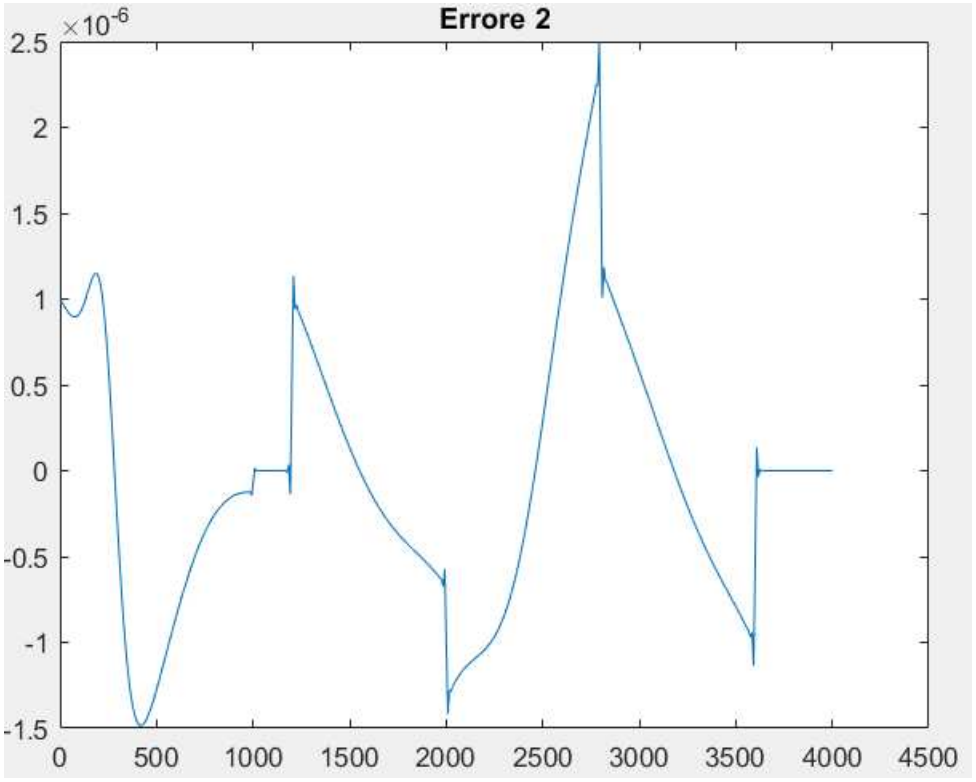


**Figura 3.18** *Errore tra le coppie misurate al giunto 2.*

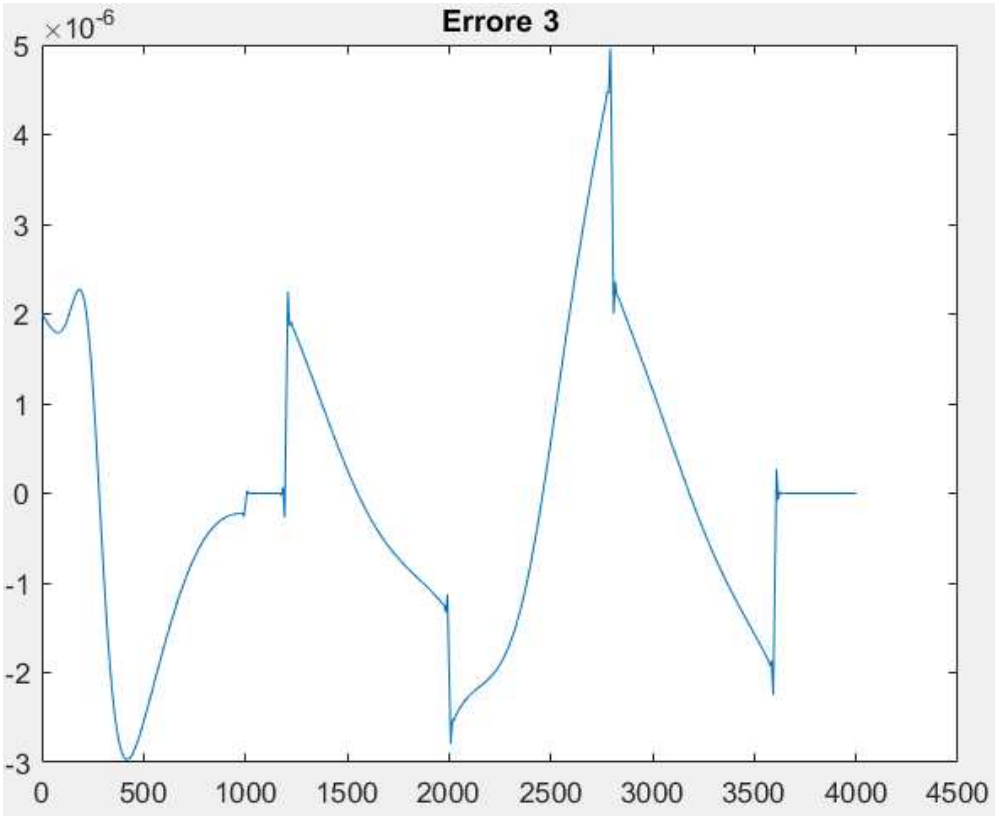


**Figura 3.19** *Errore tra le coppie misurate al giunto 3.*

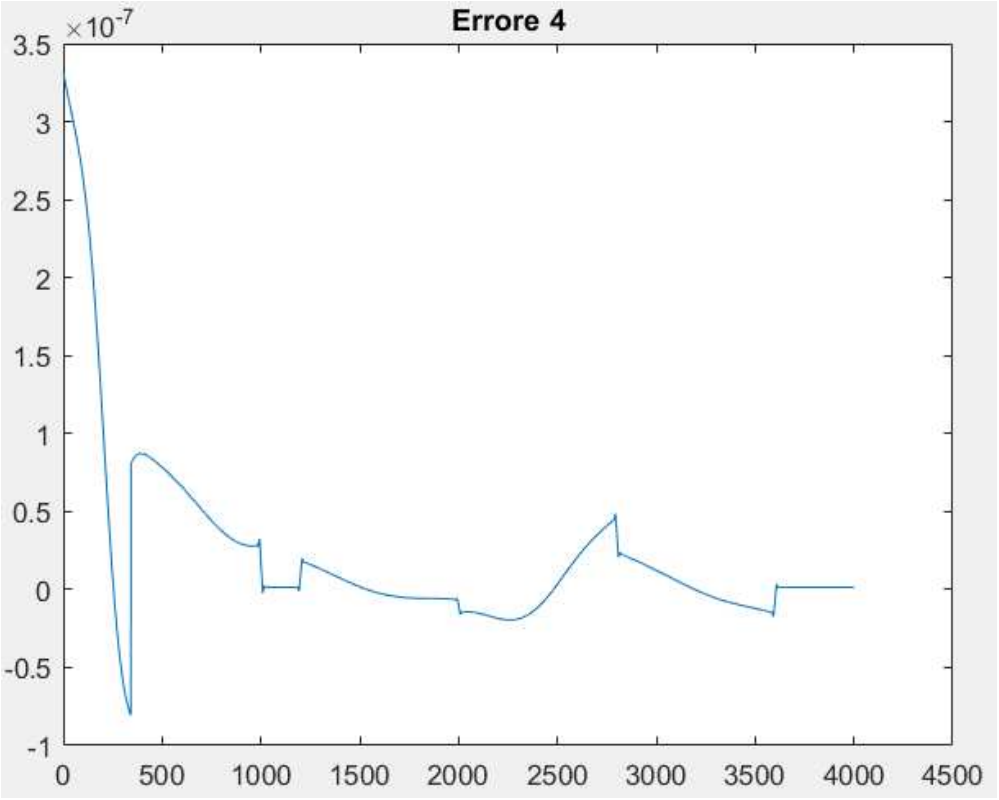


**Figura 3.20** *Errore tra le coppie misurate al giunto 4.*

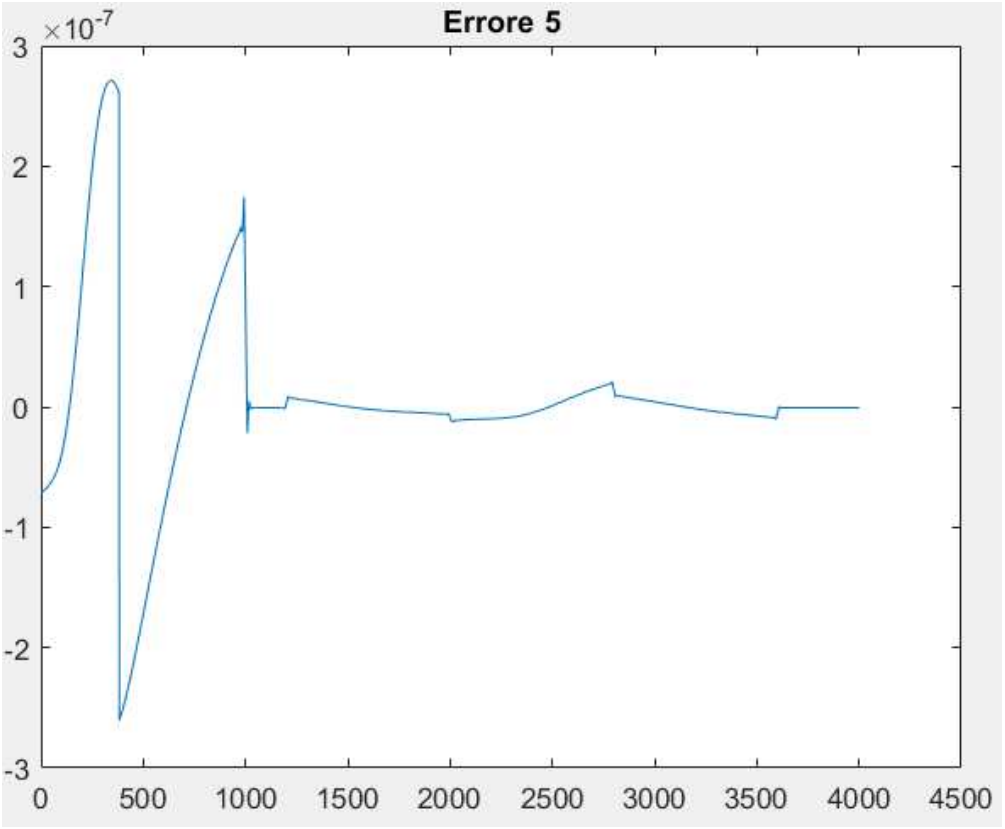


**Figura 3.21** *Errore tra le coppie misurate al giunto 5.*

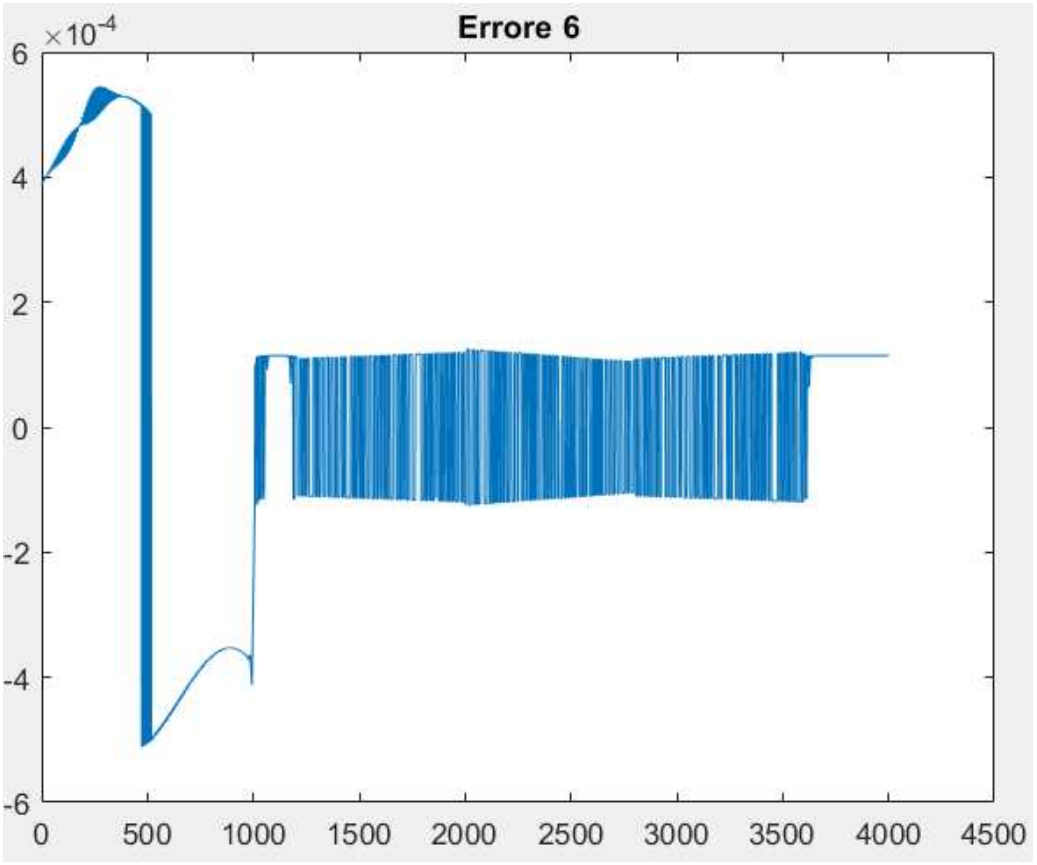


**Figura 3.22** *Errore tra le coppie misurate al giunto 6.*

# Capitolo 4

# Identificazione dei parametri dinamici

Il fine ultimo di questa trattazione consiste nell'identificazione dei parametri dinamici del robot FANUC CRX-10iA/L. Questa procedura sperimentale riveste un ruolo fondamentale all'interno del contesto della robotica collaborativa ed industriale, in cui l'affidabilità delle simulazioni è un requisito necessario per la previsione dei guasti nonché per la valutazione dei rischi associati alle applicazioni. A differenza dei parametri cinematici, i dati dinamici dei manipolatori spesso non sono forniti dai costruttori, motivo per cui è di solito richiesta una procedura di identificazione parametrica per la misurazione degli stessi [14].

I passi seguiti ai fini dell'identificazione parametrica del manipolatore sono i seguenti:

- costruzione del regressore;
- riduzione del regressore;
- identificazione dei parametri;
- verifica dei parametri.

## 4.1 Costruzione del regressore

Come riportato nei capitoli precedenti, la dinamica di un robot industriale può essere derivata attraverso le formulazioni di Newton-Eulero o Lagrange, dalle quali è sempre possibile ricondursi ad un'equazione vettoriale del tipo:

$$M(q)\ddot{\mathbf{q}} + C\left(q, \dot{\mathbf{q}}\right)\dot{\mathbf{q}} + F\left(\dot{\mathbf{q}}\right) + G(q) = \tau \tag{4.1}$$

dove

- **q** [n x 1] è il vettore degli spostamenti angolari dei giunti;
- $\dot{\mathbf{q}}$ [n x 1] è il vettore delle velocità angolari dei giunti;
- $\ddot{\mathbf{q}}$ [n x 1] è il vettore delle accelerazioni angolari dei giunti;
- **τ** [n x 1] è il vettore delle forze/coppie di attuazione;
- **M**(**q**) [n x n] è la matrice di massa (o di inerzia) del manipolatore;
- **C**(**q**, $\dot{\mathbf{q}}$) [n x 1] è il vettore dei termini centrifughi e di Coriolis;
- **G**(**q**) [n x 1] è il vettore dei termini gravitazionali;
- **F**($\dot{\mathbf{q}}$) [n x 1] è il vettore dei termini di attrito (viscoso e secco).

Tuttavia, ai fini dello scopo della presente trattazione, è necessario riscrivere l'equazione precedente in una forma lineare rispetto ad un insieme di parametri dinamici ***p*** [r x 1] opportunamente definito come:

$$Yp = \tau \tag{4.2}$$

dove ***Y*** (**q**, $\dot{\mathbf{q}}$, $\ddot{\mathbf{q}}$) [n x r] rappresenta la matrice dei coefficienti delle equazioni dinamiche, chiamato *regressore* o *matrice di regressione* [14].

Nel presente elaborato, con l'obiettivo di semplificare la procedura, si è scelto di considerare noti i vettori rappresentanti le posizioni dei baricentri, in quanto presenti in maniera non lineare all'interno delle equazioni dinamiche del manipolatore. Il vettore dei parametri inerziali da identificare può essere quindi così definito: ***p*** = [I11_1 I22_1 I33_1 I12_1 I13_1 I23_1 I11_2 I22_2 I33_2 I12_2 I13_2 I23_2 I11_3 I22_3 I33_3 I12_3 I13_3 I23_3 I11_4 I22_4 I33_4 I12_4 I13_4

I23_4 I11_5 I22_5 I33_5 I12_5 I13_5 I23_5 I11_6 I22_6 I33_6 I12_6 I13_6 I23_6 m1 m2 m3 m4 m5 m6]$^T$. Quest'ultimo è un vettore [42 x 1] contenente le componenti delle matrici di inerzia baricentriche dei membri espresse nelle terne locali e le masse degli stessi.

Il regressore è stato ottenuto attraverso l'utilizzo della funzione *built-in* `jacobian` inclusa in MATLAB, come riportato in appendice A.8, e successivamente è stata salvata in memoria la funzione che consente il calcolo dello stesso. Quest'ultima è indispensabile per la valutazione numerica del regressore, la quale è stata effettuata attraverso l'implementazione di molteplici test in ADAMS tali da eccitare la dinamica del manipolatore in maniera completa, attraverso l'assegnazione di valori randomici alle variabili di giunto. Le traiettorie da utilizzare devono essere sufficientemente ricche ma tali da non eccitare dinamiche non modellate (tipicamente vengono impiegate traiettorie di tipo polinomiale) [21].

Il codice implementato per l'ottenimento del regressore relativo al singolo test è riportato nell'appendice A.9.

Dopo aver effettuato i test e dopo aver riordinato adeguatamente le righe delle matrici di regressione ottenute per renderle compatibili con la struttura dei vettori contenenti le coppie sui giunti, è possibile "impilare" i regressori risultanti in modo da ottenere il regressore finale, caratterizzato da un numero di righe molto superiore rispetto al numero di colonne.

## 4.2 Riduzione del regressore

A causa della presenza di vincoli cinematici, non tutti i parametri dinamici influenzano la risposta dinamica del manipolatore. In generale questi possono essere distinti in: unicamente identificabili, identificabili in combinazione

lineare e non identificabili. Il processo di riduzione del regressore consente di distinguere i parametri nelle precedenti tre categorie, rappresentando quindi un passaggio fondamentale ai fini della corretta implementazione della procedura di identificazione parametrica.

Per prima cosa è necessario individuare ed eliminare dalla matrice di regressione le colonne nulle, i cui parametri corrispondenti rappresentano quindi quelli non identificabili, ovvero quelli che non intervengono nella dinamica del manipolatore. In questo caso i parametri appartenenti a questa categoria sono: I11_1, I33_1, I12_1, I13_1, I23_1 e m1.

Il passo successivo consiste nel distinguere i parametri tra unicamente identificabili e identificabili in combinazione lineare. Per fare questo tipicamente vengono utilizzati due approcci numerici: la *QR decomposition* e la *SVD* (*Singular Value Decomposition*). Nella presente trattazione si è scelto di utilizzare la funzione *built-in* `svd` inclusa in MATLAB, la quale consente di riscrivere la matrice di regressione Y come $Y=U\Sigma V^T$, dove $\Sigma=\text{diag}(\sigma_i)$ è una matrice diagonale i cui elementi sono i valori singolari di Y [14]. L'utilizzo di tale funzione consente quindi di distinguere, tra i 42 parametri dinamici individuati, 6 parametri non identificabili, 5 identificabili in combinazione lineare e 31 univocamente identificabili.

## 4.3 Identificazione dei parametri

A seguito dell'individuazione dei parametri univocamente identificabili, è possibile implementare il calcolo dei parametri inerziali attraverso la stima ai minimi quadrati:

$$p=\left(Y^{\mathrm{T}}Y\right)^{-1}Y^{\mathrm{T}}\tau \tag{4.3}$$

dove $\boldsymbol{p}$ è il vettore dei parametri identificati, $\boldsymbol{Y}$ è il regressore numerico ridotto e $\boldsymbol{\tau}$ è il vettore delle coppie valutato attraverso test implementati in ADAMS i cui dati sono stati esportati in MATLAB.

La precedente espressione è stata valutata in MATLAB attraverso la funzione *built-in* `pinv` del software, la quale consente di estrarre la matrice pseudoinversa $\boldsymbol{Y}^{\dagger}=(\boldsymbol{Y}^{T}\boldsymbol{Y})^{-1}\boldsymbol{Y}^{T}$ [r x n] della matrice $\boldsymbol{Y}$ (pseudoinversa di Moore-Penrose), ovvero una matrice che può fungere da sostituta parziale della matrice inversa nei casi in cui quest'ultima non esista. Questa funzione consente di risolvere un sistema di equazioni lineari quando lo stesso presenta molte soluzioni o una soluzione non unica.

La matrice pseudoinversa di Moore-Penrose gode delle seguenti proprietà:

- se n=r, $\boldsymbol{Y}^{\dagger}=\boldsymbol{Y}^{-1}$ se $\boldsymbol{Y}$ è a rango pieno;
- se n>r, che corrisponde al caso in cui il numero di equazioni sia maggiore rispetto alle variabili da identificare, la matrice pseudoinversa restituisce la soluzione tale che la norma del vettore residuo $\boldsymbol{Y}\cdot\boldsymbol{p}-\boldsymbol{\tau}$ sia minima;
- se n<r, ovvero il caso in cui le equazioni a disposizione sono in numero inferiore rispetto alle variabili da identificare, il problema ha generalmente infinite soluzioni e la soluzione ottenuta con la matrice pseudoinversa è la soluzione particolare nella quale la norma del vettore dei parametri è minima.

Se la matrice $\boldsymbol{Y}$ è a rango pieno allora per i tre casi appena presentati il calcolo della matrice pseudoinversa di Moore-Penrose è:

- se n=r, $\boldsymbol{Y}^{\dagger}=\boldsymbol{Y}^{-1}$;
- se n>r, $\boldsymbol{Y}^{\dagger}=(\boldsymbol{Y}^{T}\boldsymbol{Y})^{-1}\boldsymbol{Y}^{T}$;
- se n<r, $\boldsymbol{Y}^{\dagger}=\boldsymbol{Y}^{T}(\boldsymbol{Y}\boldsymbol{Y}^{T})^{-1}$.

Nella presente trattazione si è scelto di fornire un numero di equazioni n maggiore del numero delle incognite da identificare r. Così facendo la matrice $\boldsymbol{Y}$ rientra nel caso di un sistema sovradeterminato in cui il numero di righe è maggiore del numero di colonne. Quindi l'equazione di risoluzione che consente l'identificazione dei parametri dinamici è:

$$p = Y^{\dagger} \tau \tag{4.4}$$

Il calcolo della pseudoinversa può dare luogo ad errori numerici nel caso in cui la matrice $\boldsymbol{Y}^T\boldsymbol{Y}$ risulti mal condizionata numericamente. In questi casi è utile utilizzare algoritmi ottimizzati che consentono di risolvere il sistema lineare senza il calcolo dell'inversa di $\boldsymbol{Y}^T\boldsymbol{Y}$. La funzione `pinv` di MATLAB utilizza infatti la *Singular Value Decomposition* per formare la matrice pseudoinversa, uno degli strumenti migliori per l'analisi dei problemi mal posti. La documentazione dettagliata relativa a questo aspetto è riportata in [22].

## 4.4 Verifica dei parametri

Per verificare la procedura di identificazione proposta in questo capitolo, è stato utilizzato lo stesso codice implementato per la verifica del modello dinamico realizzato attraverso la formulazione di Lagrange riportato nell'appendice A.7, all'interno del quale sono stati introdotti i parametri dinamici identificati al paragrafo precedente. Inoltre le traiettorie utilizzate per effettuare il confronto tra le coppie misurate in ADAMS e quelle calcolate attraverso lo script MATLAB aggiornato con i nuovi parametri sono le stesse riportate nel paragrafo 3.3. Da quest'ultima procedura di confronto è possibile valutare gli errori tra le misurazioni di coppia ottenute in ADAMS e MATLAB in

maniera analoga a quanto riportato nei paragrafi 3.3.1 e 3.3.2. Di seguito, nelle Figure 4.1, 4.2, 4.3, 4.4, 4.5 e 4.6, sono riportati i grafici relativi a questi ultimi i cui valori ridotti dimostrano la correttezza dei parametri identificati.

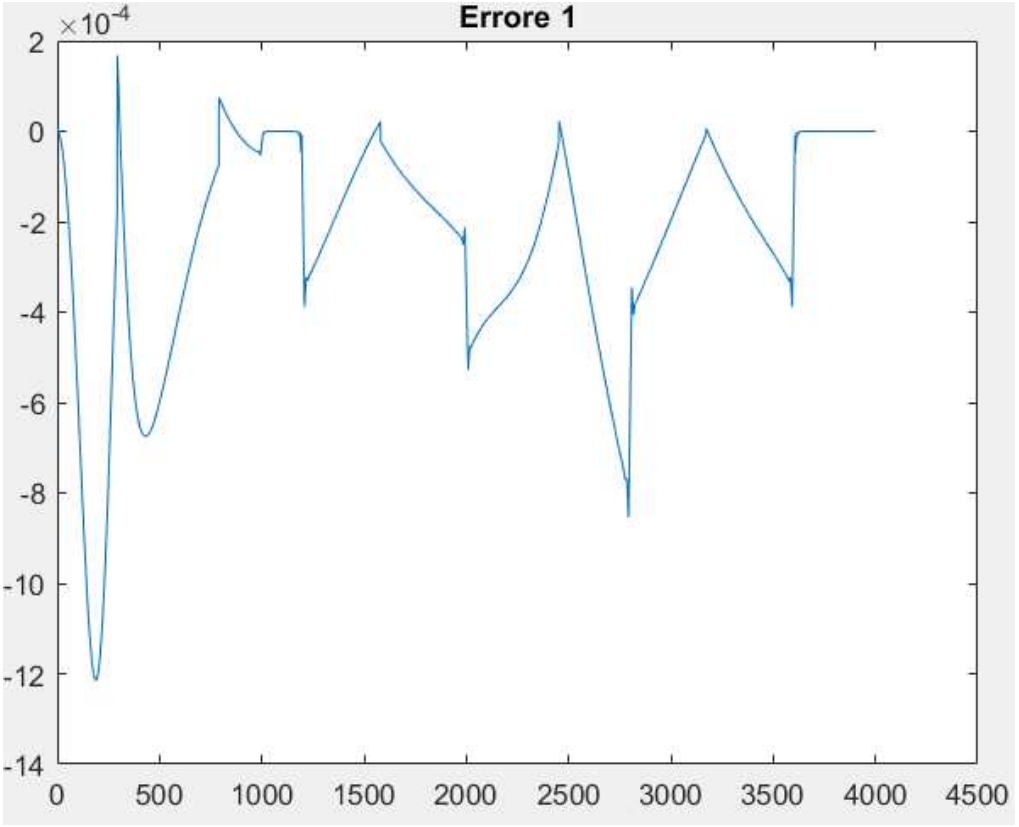


**Figura 4.1** *Errore tra le coppie misurate al giunto 1.*

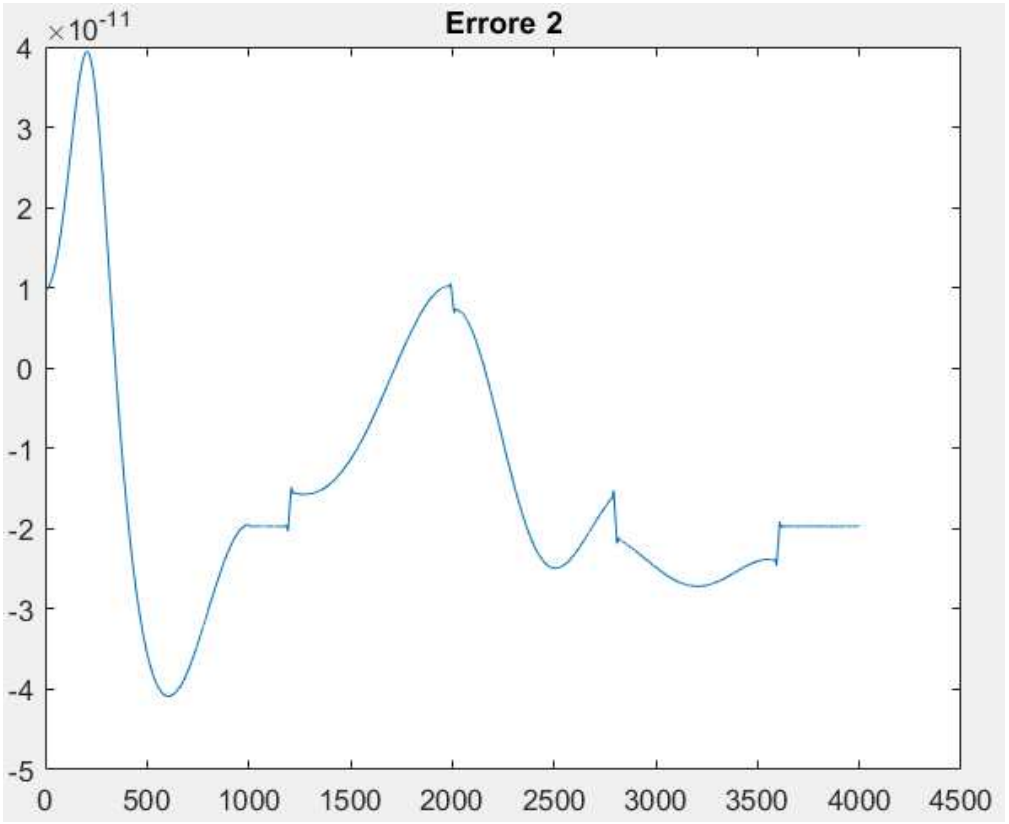


**Figura 4.2** *Errore tra le coppie misurate al giunto 2.*

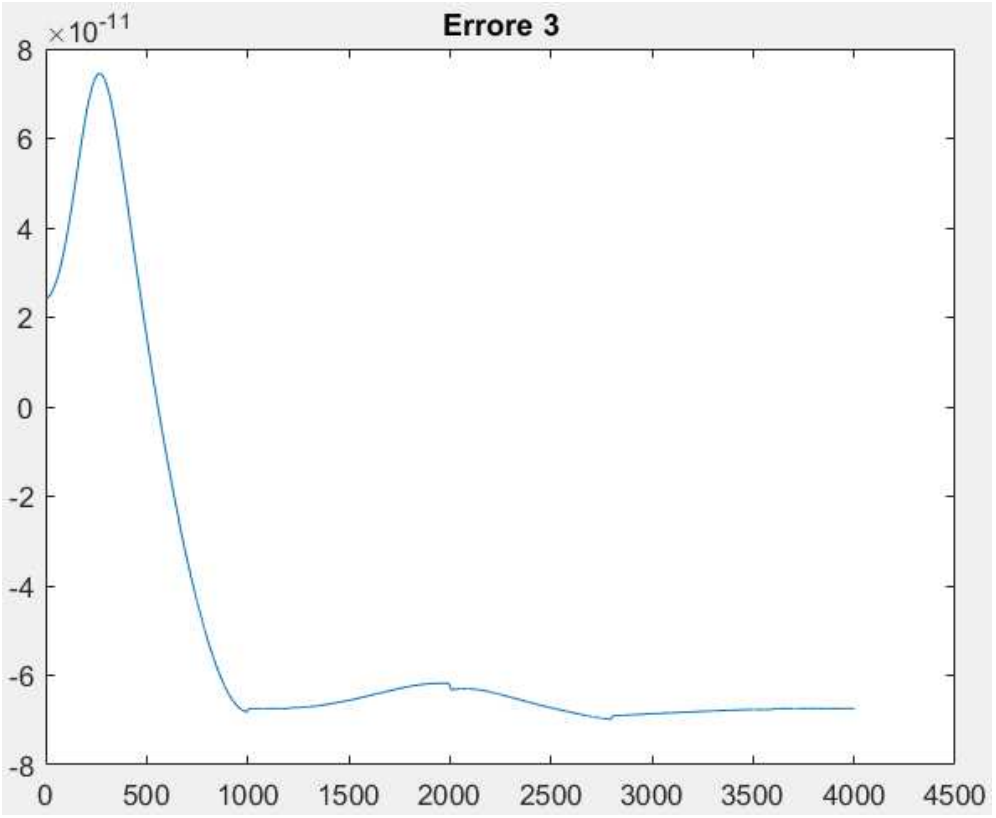


**Figura 4.3** *Errore tra le coppie misurate al giunto 3.*

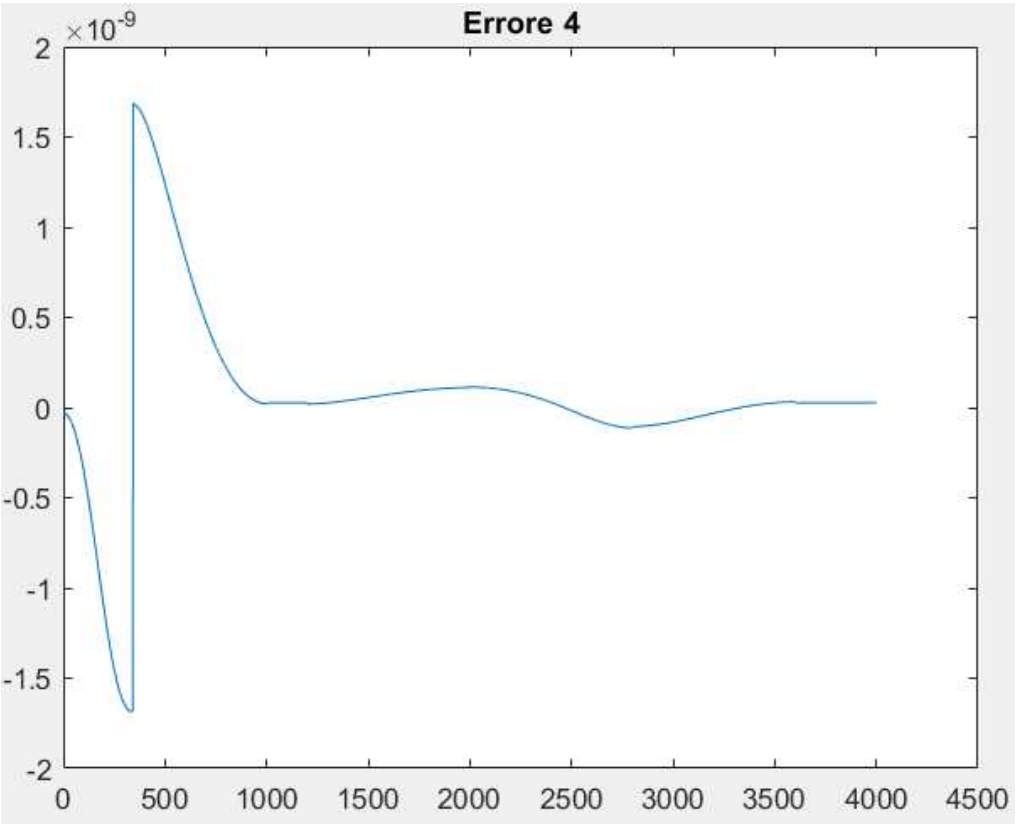


**Figura 4.4** *Errore tra le coppie misurate al giunto 4.*

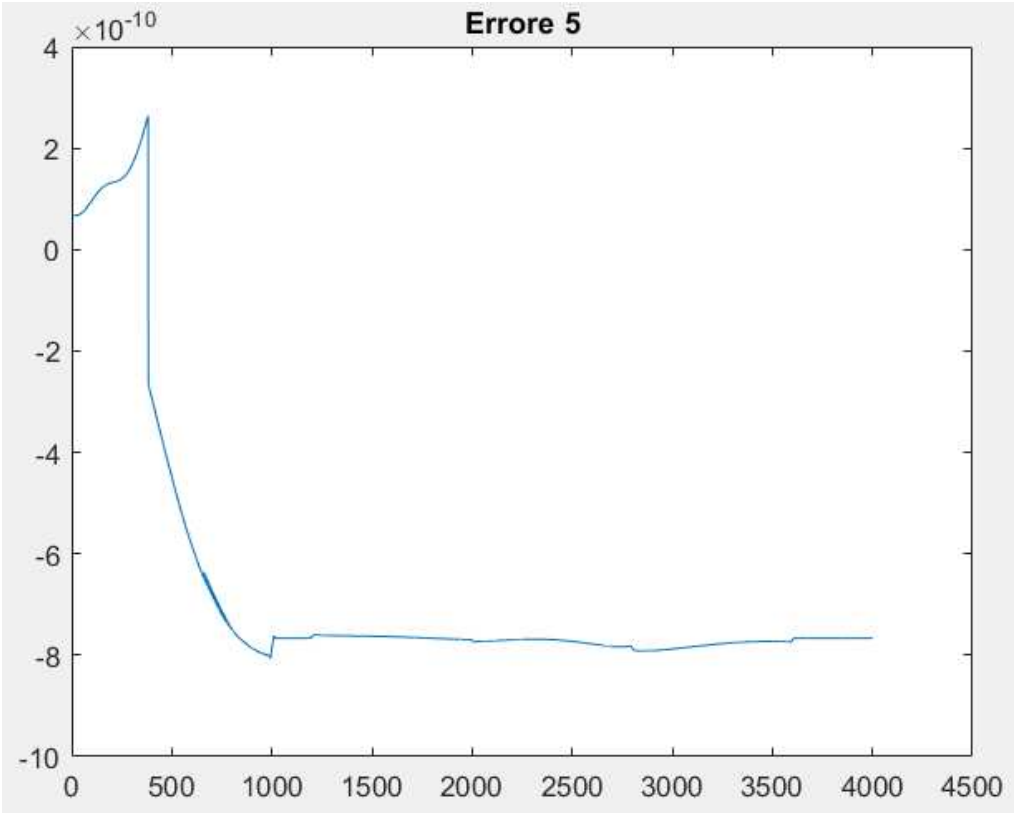


**Figura 4.5** *Errore tra le coppie misurate al giunto 5.*

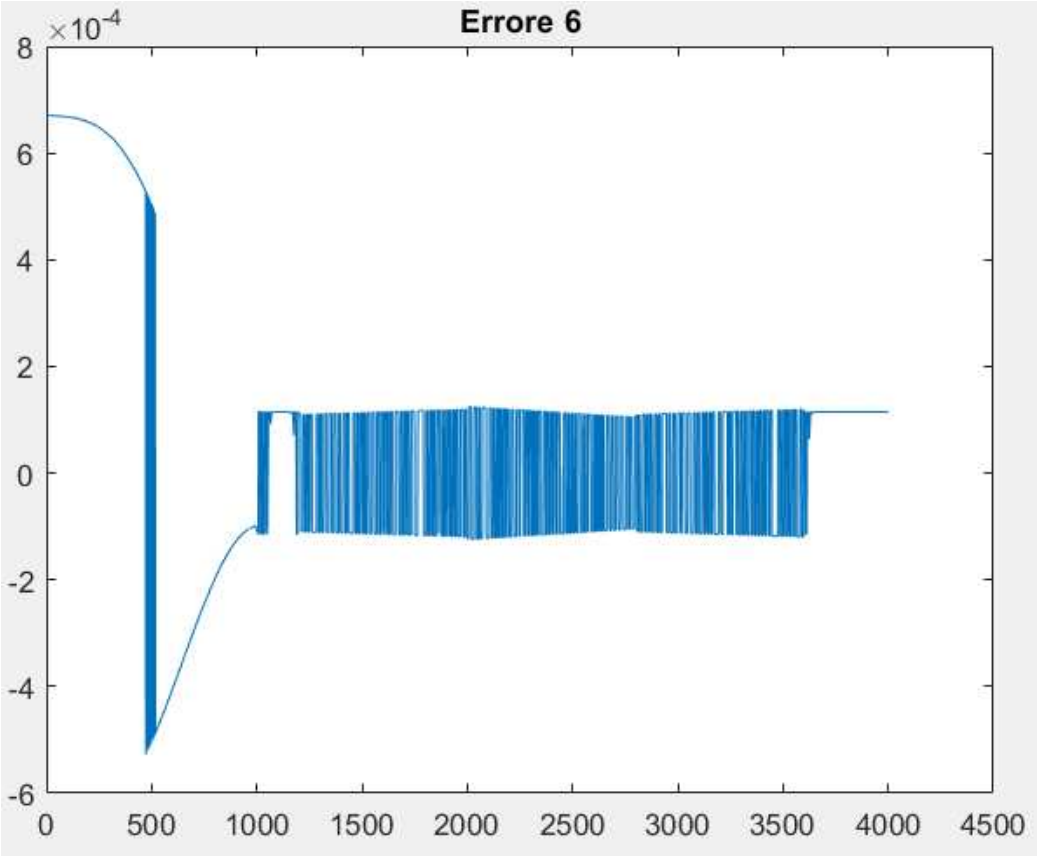


**Figura 4.6** *Errore tra le coppie misurate al giunto 6.*

Per ottenere dei risultati più accurati rispetto alla semplice ispezione visiva dei grafici ottenuti è stato introdotto l'indice di prestazione NRMSE (*Normalized Root Mean Square Error*) così definito:

$$NRMSE = 1 - \left| x - \frac{x_{ref}}{x - mean\left( x_{ref} \right)} \right|^2 \tag{4.5}$$

dove $x$ è la coppia calcolata attraverso i parametri identificati, $x_{ref}$ è la coppia di riferimento calcolata attraverso la simulazione in ADAMS e $mean(x_{ref})$ è la media della coppia di riferimento. Dalla formula precedente è possibile dedurre che tanto più il valore NRMSE è vicino all'unità quanto più i parametri stimati sono prossimi a quelli reali, quindi il modello implementato gode di un buon *fitting* [23]. Il calcolo del precedente indice di prestazione è stato implementato

in MATLAB attraverso l'ausilio della funzione *built-in* `goodnessOfFit`, la quale restituisce la norma dell'errore tra un set di dati di test e uno di riferimento. Questa funzione consente infatti di specificare la tipologia di funzione di costo da utilizzare nel confronto, in questo caso appunto NRMSE. Il valore restituito dalla precedente funzione rappresenta un *fitting* ottimale quanto più è vicino allo zero.

Gli indici di performance valutati per ogni giunto sono riportati nella seguente tabella 4.1:

| Giunto | goodnessOfFit |
|---|---|
| 1 | 0.0011 |
| 2 | 1.5570e-10 |
| 3 | 8.7096e-10 |
| 4 | 1.0981e-09 |
| 5 | 2.6027e-09 |
| 6 | 0.0010 |

**Tabella 4.1** *Valutazione dell'indice di performance a seguito dell'identificazione parametrica.*

Dalla precedente tabella è quindi possibile confermare la correttezza dei parametri dinamici identificati attraverso la procedura implementata.

# Capitolo 5

# Conclusioni

Nel presente elaborato è stata sviluppata una procedura di identificazione dei parametri dinamici di un robot collaborativo industriale, strumento necessario ai fini dell'implementazione di algoritmi di controllo e pianificazione della traiettoria, non essendo tali parametri forniti dalle case produttrici. L'accuratezza di suddetti parametri determina la precisione, le prestazioni, la sicurezza e la robustezza di tali algoritmi, consentendo la realizzazione di modelli del manipolatore nonché la simulazione affidabile dello stesso. Partendo dall'implementazione del modello cinematico del robot CRX-10iA/L di produzione FANUC, attraverso la convenzione di Denavit-Hartenberg, si è passati alla modellazione dinamica dello stesso, realizzata attraverso le formulazioni di Newton-Eulero e Lagrange; tali modelli sono poi stati verificati attraverso la tecnica della co-simulazione, ovvero collaborazione tra i software ADAMS e Simulink. Di seguito è stata implementata la procedura di identificazione parametrica, articolata nei suoi passi fondamentali: costruzione e riduzione del regressore, calcolo dei parametri e verifica del metodo.

Una possibile evoluzione del presente elaborato può essere ottenuta attraverso l'implementazione di test sperimentali eseguiti sul robot per la verifica degli algoritmi presentati, non essendo stato possibile per impossibilità tecniche. In questo modo sarà possibile ottenere dati provenienti dal manipolatore, dati che nella presente trattazione sono stati ricavati per mezzo di simulazioni implementate attraverso il software MSC ADAMS.

# Appendice A

# Codici MATLAB

Di seguito sono riportati gli script MATLAB implementati nello sviluppo della presente trattazione.

**Function MATLAB A.1:** calcolo della matrice di trasformazione omogenea con rotazione attorno all'asse x e traslazione lungo l'asse x.

```
% output: Tx (matrice di trasformazione omogenea)
% input:  alpha [rad] (angolo di rotazione attorno
          all'asse x)
%         a [m] (distanza di traslazione lungo l'asse x)

function output = Rot_Tras_x(alpha,a)

% matrice di rotazione attorno all'asse x
Rx=[1     0            0;
     0    cos(alpha)   -sin(alpha);
     0    sin(alpha)   cos(alpha)];

% vettore di traslazione lungo l'asse x
Lx=[a 0 0]';
```

```
% matrice di trasformazione omogenea
Tx=[Rx    Lx;
     0 0 0 1];

output=Tx;
```

**Function MATLAB A.2:** calcolo della matrice di trasformazione omogenea con rotazione attorno all'asse z e traslazione lungo l'asse z.

```
% output: Tz (matrice di trasformazione omogenea)
% input:  theta [rad] (angolo di rotazione attorno
          all'asse z)
%         d [m] (distanza di traslazione lungo l'asse z)

function output = Rot_Tras_z(theta,d)

% matrice di rotazione attorno all'asse z
Rz=[cos(theta)  -sin(theta)  0;
     sin(theta)  cos(theta)   0;
     0          0          1];

% vettore di traslazione lungo l'asse z
Lz=[0 0 d]';

% matrice di trasformazione omogenea
Tz=[Rz    Lz;
     0 0 0 1];
```

```
output=Tz;
```

**Script MATLAB A.3:** modello cinematico del robot FANUC CRX-10iA/L con rappresentazione grafica della configurazione impostata dall'utente.

```
% assegnazione dei valori alle variabili di giunto
arbitrarie [rad]


theta_1=deg2rad(90);

theta_2=deg2rad(90);

theta_3=deg2rad(0);

theta_4=deg2rad(0);

theta_5=deg2rad(0);

theta_6=deg2rad(0);


% dimensioni del robot [m]


A=0.2604;

B=0.710;

C=A;

D=0.540;

E=0.150;

F=0.160;

G=0.245;

%% Tabella Denavit-Hartenberg

% membro 1
```

```
alpha_1 = deg2rad(90);  % rad
a_1 = 0;                % m
d_1 = G;                % m

% membro 2

alpha_2 = deg2rad(-180);  % rad
a_2 = B;                % m
d_2 = A;                % m

% membro 3

alpha_3 = deg2rad(-90);   % rad
a_3 = 0;                % m
d_3 = C;                % m

% membro 4

alpha_4 = deg2rad(-90);   % rad
a_4 = 0;                % m
d_4 = D;                % m

% membro 5

alpha_5 = deg2rad(90);  % rad
```

```
a_5 = 0;                    % m
d_5 = E;                       % m

% membro 6

alpha_6 = deg2rad(0);    % rad
a_6 = 0;                    % m
d_6 = F;                    % m
%% Matrici di trasformazione omogenea
% da {0} a {1}

Tz1=Rot_Tras_z(theta_1,d_1);
Tx1=Rot_Tras_x(alpha_1,a_1);

T01=Tz1*Tx1;

% da {1} a {2}

Tz2=Rot_Tras_z(theta_2,d_2);
Tx2=Rot_Tras_x(alpha_2,a_2);

T21=Tz2*Tx2;

% da {2} a {3}

Tz3=Rot_Tras_z(theta_3,d_3);
```

```
Tx3=Rot_Tras_x(alpha_3,a_3);

T32=Tz3*Tx3;

% da {3} a {4}

Tz4=Rot_Tras_z(theta_4,d_4);
Tx4=Rot_Tras_x(alpha_4,a_4);

T43=Tz4*Tx4;

% da {4} a {5}

Tz5=Rot_Tras_z(theta_5,d_5);
Tx5=Rot_Tras_x(alpha_5,a_5);

T54=Tz5*Tx5;

% da {5} a {6}

Tz6=Rot_Tras_z(theta_6,d_6);
Tx6=Rot_Tras_x(alpha_6,a_6);

T65=Tz6*Tx6;

% da {6} a {0}
```

```
T60=T01*T21*T32*T43*T54*T65;

%% Verifica grafica

p1=T01*[0 0 0 1]';

p2=T01*T21*[0 0 0 1]';

p3=T01*T21*T32*[0 0 0 1]';

p4=T01*T21*T32*T43*[0 0 0 1]';

p5=T01*T21*T32*T43*T54*[0 0 0 1]';

p6=T01*T21*T32*T43*T54*T65*[0 0 0 1]';


% punto intermedio aggiuntivo


Tz1_int=Rot_Tras_z(theta_1,A);

Tx1_int=Rot_Tras_x(alpha_1,0);

T1_int1=Tz1_int*Tx1_int;

p1_int=T01*T1_int1*[0 0 0 1]';


% punti notevoli del robot


P1=p1(1:3);

P1_int=p1_int(1:3);

P2=p2(1:3);

P3=p3(1:3);

P4=p4(1:3);

P5=p5(1:3);

P6=p6(1:3);
```

```
O=[0 0 0]';


dati=[O P1 P1_int P2 P3 P4 P5 P6];


% rappresentazione grafica del robot


figure

plot3(dati(1,:),dati(2,:),dati(3,:),'-o')

title('Configurazione meccanica del robot Fanuc
CRX-10iA/L')

xlabel('x [m]')

ylabel('y [m]')

zlabel('z [m]')

hold on

grid on

axis equal


% posizione del terminale rispetto all'origine


[P]=T60(1:3,4);
```

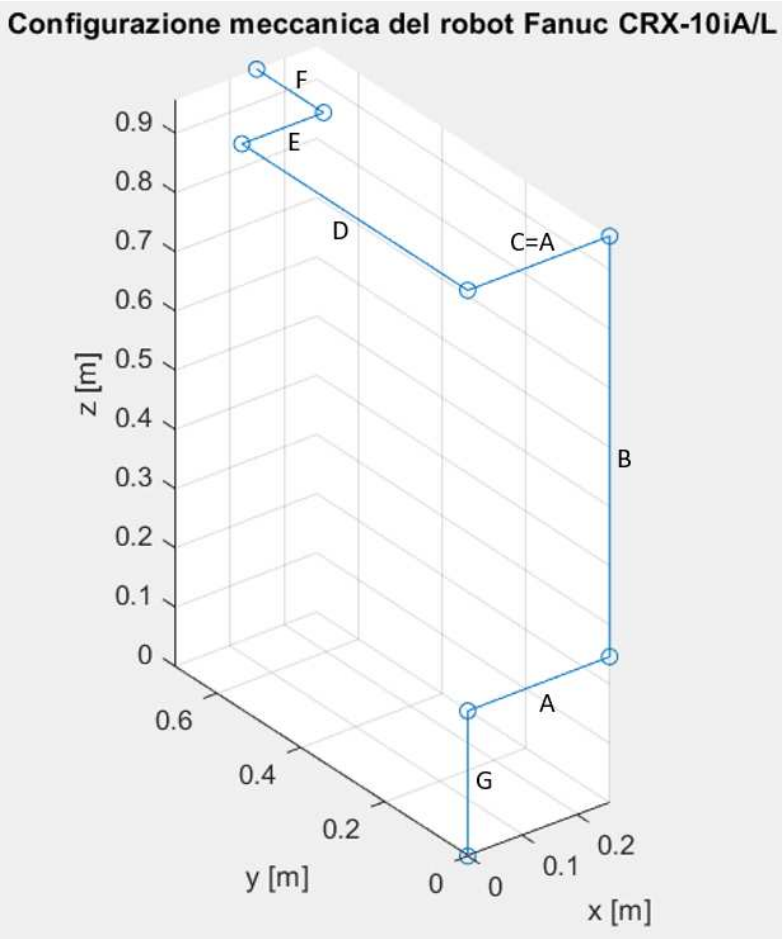


**Figura A.1** *Configurazione del robot impostata dall'utente.*

**Script MATLAB A.4:** modellazione dinamica del robot FANUC CRX-10iA/L attraverso il metodo di Newton-Eulero.

```
syms alpha_1 d_1 a_1 alpha_2 d_2 a_2 alpha_3 d_3 a_3
alpha_4 d_4 a_4 alpha_5 d_5 a_5 alpha_6 d_6 a_6 real %
parametri DH

syms c1x c1y c1z c2x c2y c2z c3x c3y c3z c4x c4y c4z c5x
c5y c5z c6x c6y c6z real % coordinate baricentri membro i
rispetto terna i

syms g m1 m2 m3 m4 m5 m6 Ic1 Ic2 Ic3 Ic4 Ic5 Ic6 real %
gravità, masse, inerzie baricentriche

syms theta_1 theta_2 theta_3 theta_4 theta_5 theta_6
theta1px theta1py theta1pz theta2px theta2py theta2pz
theta3px theta3py theta3pz theta4px theta4py theta4pz
```

```
theta5px theta5py theta5pz theta6px theta6py theta6pz
theta1ppx theta1ppy theta1ppz theta2ppx theta2ppy
theta2ppz theta3ppx theta3ppy theta3ppz theta4ppx
theta4ppy theta4ppz theta5ppx theta5ppy theta5ppz
theta6ppx theta6ppy theta6ppz real % angoli, velocità
angolari, accelerazioni angolari

syms theta1 theta2 theta3 theta4 theta5 theta6 phi1 phi2
phi3 phi4 phi5 phi6 psi1 psi2 psi3 psi4 psi5 psi6 real %
angoli di eulero tra terna centrata in cm e terna locale i


% velocità angolare membro i rispetto terna i-1

theta_1_p=[0 0 theta1pz]';

theta_2_p=[0 0 theta2pz]';

theta_3_p=[0 0 theta3pz]';

theta_4_p=[0 0 theta4pz]';

theta_5_p=[0 0 theta5pz]';

theta_6_p=[0 0 theta6pz]';


% accelerazione angolare membro i rispetto terna i-1

theta_1_pp=[0 0 theta1ppz]';

theta_2_pp=[0 0 theta2ppz]';

theta_3_pp=[0 0 theta3ppz]';

theta_4_pp=[0 0 theta4ppz]';

theta_5_pp=[0 0 theta5ppz]';

theta_6_pp=[0 0 theta6ppz]';


theta_pp = [theta_1_pp; theta_2_pp; theta_3_pp;
theta_4_pp; theta_5_pp; theta_6_pp];


%% Forward computation
```

```
% matrici di trasformazione e rotazione terne DH
% da {0} a {1}
Tz1=Rot_Tras_z(theta_1,d_1);
Tx1=Rot_Tras_x(alpha_1,a_1);
T01=Tz1*Tx1;
R01=T01(1:3,1:3);
R01=simplify(R01);
T01=simplify(T01);

% da {1} a {2}
Tz2=Rot_Tras_z(theta_2,d_2);
Tx2=Rot_Tras_x(alpha_2,a_2);
T12=Tz2*Tx2;
T02=T01*T12;
R12=T12(1:3,1:3);
R02=R01*R12;
R02=simplify(R02);
T02=simplify(T02);

% da {2} a {3}
Tz3=Rot_Tras_z(theta_3,d_3);
Tx3=Rot_Tras_x(alpha_3,a_3);
T23=Tz3*Tx3;
T03=T01*T12*T23;
R23=T23(1:3,1:3);
```

```
R03=R01*R12*R23;
R03=simplify(R03);
T03=simplify(T03);

% da {3} a {4}
Tz4=Rot_Tras_z(theta_4,d_4);
Tx4=Rot_Tras_x(alpha_4,a_4);
T34=Tz4*Tx4;
T04=T01*T12*T23*T34;
R34=T34(1:3,1:3);
R04=R01*R12*R23*R34;
R04=simplify(R04);
T04=simplify(T04);

% da {4} a {5}
Tz5=Rot_Tras_z(theta_5,d_5);
Tx5=Rot_Tras_x(alpha_5,a_5);
T45=Tz5*Tx5;
T05=T01*T12*T23*T34*T45;
R45=T45(1:3,1:3);
R05=R01*R12*R23*R34*R45;
R05=simplify(R05);
T05=simplify(T05);

% da {5} a {6}
Tz6=Rot_Tras_z(theta_6,d_6);
```

```
Tx6=Rot_Tras_x(alpha_6,a_6);
T56=Tz6*Tx6;
R56=T56(1:3,1:3);

% da {6} a {0}
T06=T01*T12*T23*T34*T45*T56;
R06=R01*R12*R23*R34*R45*R56;
R06=simplify(R06);
T06=simplify(T06);

% posizione terna locale i rispetto a terna locale i-1
r01 = T01(1:3,4);
r12 = T12(1:3,4);
r23 = T23(1:3,4);
r34 = T34(1:3,4);
r45 = T45(1:3,4);
r56 = T56(1:3,4);

% posizione baricentro membro i rispetto terna i
rc1 = [c1x c1y c1z]';
rc2 = [c2x c2y c2z]';
rc3 = [c3x c3y c3z]';
rc4 = [c4x c4y c4z]';
rc5 = [c5x c5y c5z]';
rc6 = [c6x c6y c6z]';
```

```
% parametri telaio

om0 = [0 0 0]';

om0_p = [0 0 0]';

ac0 = [0 0 0]';

ae0 = [0 0 0]';


% calcolo velocità angolari, accelerazioni angolari e
accelerazioni baricentri

om1 = om0+theta_1_p; % velocità angolare

om1_p = om0_p+cross(om0,theta_1_p)+theta_1_pp; %
accelerazione angolare

ae1 = ae0+cross(om1_p,r01)+cross(om1,cross(om1,r01)); %
accelerazione lineare centro terna locale

ac1=
ae1+cross(om1_p,R01*rc1)+cross(om1,cross(om1,R01*rc1)); %
accelerazione lineare baricentro


om2 = om1+R01*theta_2_p;

om2_p = om1_p+cross(om1,R01*theta_2_p)+R01*theta_2_pp;

ae2 =
ae1+cross(om2_p,R01*r12)+cross(om2,cross(om2,R01*r12));

ac2 =
ae2+cross(om2_p,R02*rc2)+cross(om2,cross(om2,R02*rc2));


om3 = om2+R02*theta_3_p;

om3_p = om2_p+cross(om2,R02*theta_3_p)+R02*theta_3_pp;

ae3 =
ae2+cross(om3_p,R02*r23)+cross(om3,cross(om3,R02*r23));

ac3 =
ae3+cross(om3_p,R03*rc3)+cross(om3,cross(om3,R03*rc3));
```

```
om4 = om3+R03*theta_4_p;

om4_p = om3_p+cross(om3,R03*theta_4_p)+R03*theta_4_pp;

ae4 =
ae3+cross(om4_p,R03*r34)+cross(om4,cross(om4,R03*r34));

ac4 =
ae4+cross(om4_p,R04*rc4)+cross(om4,cross(om4,R04*rc4));


om5 = om4+R04*theta_5_p;

om5_p = om4_p+cross(om4,R04*theta_5_p)+R04*theta_5_pp;

ae5 =
ae4+cross(om5_p,R04*r45)+cross(om5,cross(om5,R04*r45));

ac5 =
ae5+cross(om5_p,R05*rc5)+cross(om5,cross(om5,R05*rc5));


om6 = om5+R05*theta_6_p;

om6_p = om5_p+cross(om5,R05*theta_6_p)+R05*theta_6_pp;

ae6 =
ae5+cross(om6_p,R05*r56)+cross(om6,cross(om6,R05*r56));

ac6
=ae6+cross(om6_p,R06*rc6)+cross(om6,cross(om6,R06*rc6));


ac1 = simplify(ac1);

ac2 = simplify(ac2);

ac3 = simplify(ac3);

ac4 = simplify(ac4);

ac5 = simplify(ac5);

ac6 = simplify(ac6);
```

```
%% Backward computation

% parametri telaio

g0 = [0 0 -g]'; % gravità nel sistema globale

f7 = [0 0 0]'; % forza agente sull'ultimo membro

n7 = [0 0 0]'; % coppia agente sull'ultimo membro



% matrici rotazione (angoli eulero) da terna locale i a cm
i

Rc1 =
[-sin(phi1)*cos(theta1)*sin(psi1)+cos(phi1)*cos(psi1)
-sin(phi1)*cos(theta1)*cos(psi1)-cos(phi1)*sin(psi1)
sin(phi1)*sin(theta1);
cos(phi1)*cos(theta1)*sin(psi1)+sin(phi1)*cos(psi1)
-cos(phi1)*cos(theta1)*cos(psi1)-sin(phi1)*sin(psi1)
-cos(phi1)*sin(theta1); sin(phi1)*sin(psi1)
sin(theta1)*cos(psi1) cos(theta1)];

Rc2 =
[-sin(phi2)*cos(theta2)*sin(psi2)+cos(phi2)*cos(psi2)
-sin(phi2)*cos(theta2)*cos(psi2)-cos(phi2)*sin(psi2)
sin(phi2)*sin(theta2);
cos(phi2)*cos(theta2)*sin(psi2)+sin(phi2)*cos(psi2)
-cos(phi2)*cos(theta2)*cos(psi2)-sin(phi2)*sin(psi2)
-cos(phi2)*sin(theta2); sin(phi2)*sin(psi2)
sin(theta2)*cos(psi2) cos(theta2)];

Rc3 =
[-sin(phi3)*cos(theta3)*sin(psi3)+cos(phi3)*cos(psi3)
-sin(phi3)*cos(theta3)*cos(psi3)-cos(phi3)*sin(psi3)
sin(phi3)*sin(theta3);
cos(phi3)*cos(theta3)*sin(psi3)+sin(phi3)*cos(psi3)
-cos(phi3)*cos(theta3)*cos(psi3)-sin(phi3)*sin(psi3)
-cos(phi3)*sin(theta3); sin(phi3)*sin(psi3)
sin(theta3)*cos(psi3) cos(theta3)];

Rc4 =
[-sin(phi4)*cos(theta4)*sin(psi4)+cos(phi4)*cos(psi4)
-sin(phi4)*cos(theta4)*cos(psi4)-cos(phi4)*sin(psi4)
sin(phi4)*sin(theta4);
cos(phi4)*cos(theta4)*sin(psi4)+sin(phi4)*cos(psi4)
-cos(phi4)*cos(theta4)*cos(psi4)-sin(phi4)*sin(psi4)
```

```
-cos(phi4)*sin(theta4); sin(phi4)*sin(psi4)
sin(theta4)*cos(psi4) cos(theta4)];

Rc5 =
[-sin(phi5)*cos(theta5)*sin(psi5)+cos(phi5)*cos(psi5)
-sin(phi5)*cos(theta5)*cos(psi5)-cos(phi5)*sin(psi5)
sin(phi5)*sin(theta5);
cos(phi5)*cos(theta5)*sin(psi5)+sin(phi5)*cos(psi5)
-cos(phi5)*cos(theta5)*cos(psi5)-sin(phi5)*sin(psi5)
-cos(phi5)*sin(theta5); sin(phi5)*sin(psi5)
sin(theta5)*cos(psi5) cos(theta5)];

Rc6 =
[-sin(phi6)*cos(theta6)*sin(psi6)+cos(phi6)*cos(psi6)
-sin(phi6)*cos(theta6)*cos(psi6)-cos(phi6)*sin(psi6)
sin(phi6)*sin(theta6);
cos(phi6)*cos(theta6)*sin(psi6)+sin(phi6)*cos(psi6)
-cos(phi6)*cos(theta6)*cos(psi6)-sin(phi6)*sin(psi6)
-cos(phi6)*sin(theta6); sin(phi6)*sin(psi6)
sin(theta6)*cos(psi6) cos(theta6)];



Rc1 = simplify(Rc1);

Rc2 = simplify(Rc2);

Rc3 = simplify(Rc3);

Rc4 = simplify(Rc4);

Rc5 = simplify(Rc5);

Rc6 = simplify(Rc6);



% matrice inerzia trasformata da cm a terna locale i

I6 = Rc6*Ic6*Rc6';

I5 = Rc5*Ic5*Rc5';

I4 = Rc4*Ic4*Rc4';

I3 = Rc3*Ic3*Rc3';

I2 = Rc2*Ic2*Rc2';

I1 = Rc1*Ic1*Rc1';
```

```
% calcolo forze, inerzia membro i e coppie rispetto alla
terna 0

f6 = f7+m6*(ac6-g0); % fi-1,i

I60 = R06*I6*R06'; % inerzia nel sistema globale

n6=-cross(f6,R05*r56+R06*rc6)+n7+cross(f7,R06*rc6)+I60*om6
_p+cross(om6,I60*om6); % Ni-1,i



f5 = f6+m5*(ac5-g0);

I50 = R05*I5*R05';

n5 =
-cross(f5,R04*r45+R05*rc5)+n6+cross(f6,R05*rc5)+I50*om5_p+
cross(om5,I50*om5);



f4 = f5+m4*(ac4-g0);

I40 = R04*I4*R04';

n4 =
-cross(f4,R03*r34+R04*rc4)+n5+cross(f5,R04*rc4)+I40*om4_p+
cross(om4,I40*om4);



f3 = f4+m3*(ac3-g0);

I30 = R03*I3*R03';

n3 =
-cross(f3,R02*r23+R03*rc3)+n4+cross(f4,R03*rc3)+I30*om3_p+
cross(om3,I30*om3);



f2 = f3+m2*(ac2-g0);

I20 = R02*I2*R02';

n2 =
-cross(f2,R01*r12+R02*rc2)+n3+cross(f3,R02*rc2)+I20*om2_p+
cross(om2,I20*om2);
```

```
f1 = f2+m1*(ac1-g0);

I10 = R01*I1*R01';

n1 =
-cross(f1,r01+R01*rc1)+n2+cross(f2,R01*rc1)+I10*om1_p+cros
s(om1,I10*om1);


% coppia membro i espressa in terna i-1

n1l = n1;

n2l = R01'*n2;

n3l = R02'*n3;

n4l = R03'*n4;

n5l = R04'*n5;

n6l = R05'*n6;


% componenti delle coppie lungo asse z della terna i-1

n1l = n1l(3,1);

n2l = n2l(3,1);

n3l = n3l(3,1);

n4l = n4l(3,1);

n5l = n5l(3,1);

n6l = n6l(3,1);
```

**Script MATLAB A.5:** modellazione dinamica del robot FANUC CRX-10iA/L attraverso il metodo di Lagrange.

```
syms alpha_1 d_1 a_1 alpha_2 d_2 a_2 alpha_3 d_3 a_3
alpha_4 d_4 a_4 alpha_5 d_5 a_5 alpha_6 d_6 a_6 real %
parametri DH

syms c1x c1y c1z c2x c2y c2z c3x c3y c3z c4x c4y c4z c5x
c5y c5z c6x c6y c6z real % coordinate baricentri membro i
rispetto terna i

syms g m1 m2 m3 m4 m5 m6 Ic1 Ic2 Ic3 Ic4 Ic5 Ic6 real %
gravità, masse, inerzie baricentriche

syms theta_1 theta_2 theta_3 theta_4 theta_5 theta_6
theta1px theta1py theta1pz theta2px theta2py theta2pz
theta3px theta3py theta3pz theta4px theta4py theta4pz
theta5px theta5py theta5pz theta6px theta6py theta6pz
theta1ppx theta1ppy theta1ppz theta2ppx theta2ppy
theta2ppz theta3ppx theta3ppy theta3ppz theta4ppx
theta4ppy theta4ppz theta5ppx theta5ppy theta5ppz
theta6ppx theta6ppy theta6ppz real % angoli, velocità
angolari, accelerazioni angolari

syms theta1 theta2 theta3 theta4 theta5 theta6 phi1 phi2
phi3 phi4 phi5 phi6 psi1 psi2 psi3 psi4 psi5 psi6 real %
angoli eulero tra terna centrata in cm e terna locale i

syms I11_1 I22_1 I33_1 I12_1 I13_1 I23_1 I11_2 I22_2 I33_2
I12_2 I13_2 I23_2 I11_3 I22_3 I33_3 I12_3 I13_3 I23_3
I11_4 I22_4 I33_4 I12_4 I13_4 I23_4 I11_5 I22_5 I33_5
I12_5 I13_5 I23_5 I11_6 I22_6 I33_6 I12_6 I13_6 I23_6 real
% componenti matrici inerzia espressa nella terna locale i



theta = [theta_1 theta_2 theta_3 theta_4 theta_5
theta_6]';

thetap = [theta1pz theta2pz theta3pz theta4pz theta5pz
theta6pz]';

thetapp = [theta1ppz theta2ppz theta3ppz theta4ppz
theta5ppz theta6ppz]';

rc1 = [c1x c1y c1z]';

rc2 = [c2x c2y c2z]';

rc3 = [c3x c3y c3z]';

rc4 = [c4x c4y c4z]';
```

```
rc5 = [c5x c5y c5z]';

rc6 = [c6x c6y c6z]';

theta1p = [0 0 theta1pz]'; % [rad/s]

theta2p = [0 0 theta2pz]'; % [rad/s]

theta3p = [0 0 theta3pz]'; % [rad/s]

theta4p = [0 0 theta4pz]'; % [rad/s]

theta5p = [0 0 theta5pz]'; % [rad/s]

theta6p = [0 0 theta6pz]'; % [rad/s]


% corpo 1

Tz1=Rot_Tras_z(pi/2 + theta_1,d_1);

Tx1=Rot_Tras_x(alpha_1,a_1);

T01=Tz1*Tx1;

% sostituzione valori cos(alpha) e sin(alpha) per ridurre
il carico

% computazionale

for i=1:16

    T01(i) = subs(T01(i), cos(alpha_1), 0);

    T01(i) = subs(T01(i), sin(alpha_1), 1);

    T01(i) = subs(T01(i), a_1, 0);

end

R01 = T01(1:3,1:3);

r01 = T01(1:3,4);

om1 = theta1p;

v1 = cross(om1,r01);

vc1 = cross(om1,R01*rc1) + v1;
```

```
I10 = simplify(R01*[I11_1 I12_1 I13_1; I12_1 I22_1 I23_1;
I13_1 I23_1 I33_1]*R01');

% T = Ttrasl + Trot

Ecin1 = 1/2*m1*vc1'*vc1 + 1/2*om1'*I10*om1;

% energia potenziale

rc10 = T01*[rc1; 1];

Epot1 = m1*g*rc10(3);


% corpo 2

Tz2=Rot_Tras_z(pi/2 + theta_2,d_2);

Tx2=Rot_Tras_x(alpha_2,a_2);

T12=Tz2*Tx2;

for i=1:16

    T12(i) = subs(T12(i), cos(alpha_2), -1);

    T12(i) = subs(T12(i), sin(alpha_2), 0);

end

T02=T01*T12;

R12=T12(1:3,1:3);

R02=R01*R12;

r12 = T12(1:3,4);

om2 = om1+R01*theta2p;

v2 = v1 + cross(om2,R01*r12);

vc2 = cross(om2,R02*rc2) + v2;

I20 = simplify(R02*[I11_2 I12_2 I13_2; I12_2 I22_2 I23_2;
I13_2 I23_2 I33_2]*R02');

Ecin2 = 1/2*m2*vc2'*vc2 + 1/2*om2'*I20*om2;

rc20 = T02*[rc2; 1];
```

```
Epot2 = m2*g*rc20(3);


% corpo 3

Tz3=Rot_Tras_z(theta_3,d_3);

Tx3=Rot_Tras_x(alpha_3,a_3);

T23=Tz3*Tx3;

for i=1:16

    T23(i) = subs(T23(i), cos(alpha_3), 0);

    T23(i) = subs(T23(i), sin(alpha_3), -1);

    T23(i) = subs(T23(i), a_3, 0);

end

T03=T01*T12*T23;

R23=T23(1:3,1:3);

R03=R01*R12*R23;

r23 = T23(1:3,4);

om3 = om2+R02*theta3p;

v3 = v2 + cross(om3,R02*r23);

vc3 = cross(om3,R03*rc3) + v3;

I30 = simplify(R03*[I11_3 I12_3 I13_3; I12_3 I22_3 I23_3;
I13_3 I23_3 I33_3]*R03');

Ecin3 = 1/2*m3*vc3'*vc3 + 1/2*om3'*I30*om3;

rc30 = T03*[rc3; 1];

Epot3 = m3*g*rc30(3);


% corpo 4

Tz4=Rot_Tras_z(theta_4,d_4);

Tx4=Rot_Tras_x(alpha_4,a_4);
```

```
T34=Tz4*Tx4;

for i=1:16

    T34(i) = subs(T34(i), cos(alpha_4), 0);

    T34(i) = subs(T34(i), sin(alpha_4), -1);

    T34(i) = subs(T34(i), a_4, 0);

end

T04=T01*T12*T23*T34;

R34=T34(1:3,1:3);

R04=R01*R12*R23*R34;

r34 = T34(1:3,4);

om4 = om3+R03*theta4p;

v4 = v3 + cross(om4,R03*r34);

vc4 = cross(om4,R04*rc4) + v4;

I40 = simplify(R04*[I11_4 I12_4 I13_4; I12_4 I22_4 I23_4;
I13_4 I23_4 I33_4]*R04');

Ecin4 = 1/2*m4*vc4'*vc4 + 1/2*om4'*I40*om4;

rc40 = T04*[rc4; 1];

Epot4 = m4*g*rc40(3);


% corpo 5

Tz5=Rot_Tras_z(theta_5,d_5);

Tx5=Rot_Tras_x(alpha_5,a_5);

T45=Tz5*Tx5;

for i=1:16

    T45(i) = subs(T45(i), cos(alpha_5), 0);

    T45(i) = subs(T45(i), sin(alpha_5), 1);

    T45(i) = subs(T45(i), a_5, 0);
```

```
end

T05=T01*T12*T23*T34*T45;

R45=T45(1:3,1:3);

R05=R01*R12*R23*R34*R45;

r45 = T45(1:3,4);

om5 = om4+R04*theta5p;

v5 = v4 + cross(om5,R04*r45);

vc5 = cross(om5,R05*rc5) + v5;

I50 = simplify(R05*[I11_5 I12_5 I13_5; I12_5 I22_5 I23_5;
I13_5 I23_5 I33_5]*R05');

Ecin5 = 1/2*m5*vc5'*vc5 + 1/2*om5'*I50*om5;

rc50 = T05*[rc5; 1];

Epot5 = m5*g*rc50(3);


% corpo 6

Tz6=Rot_Tras_z(theta_6,d_6);

Tx6=Rot_Tras_x(alpha_6,a_6);

T56=Tz6*Tx6;

for i=1:16

    T56(i) = subs(T56(i), cos(alpha_6), 1);

    T56(i) = subs(T56(i), sin(alpha_6), 0);

    T56(i) = subs(T56(i), a_6, 0);

end

T06=T01*T12*T23*T34*T45*T56;

R56=T56(1:3,1:3);

R06=R01*R12*R23*R34*R45*R56;

r56 = T56(1:3,4);
```

```
om6 = om5+R05*theta6p;

v6 = v5 + cross(om6,R05*r56);

vc6 = cross(om6,R06*rc6) + v6;

I60 = simplify(R06*[I11_6 I12_6 I13_6; I12_6 I22_6 I23_6;
I13_6 I23_6 I33_6]*R06');

Ecin6 = 1/2*m6*vc6'*vc6 + 1/2*om6'*I60*om6;

rc60 = T06*[rc6; 1];

Epot6 = m6*g*rc60(3);



Ecin = Ecin1 + Ecin2 + Ecin3 + Ecin4 + Ecin5 + Ecin6;

Epot = Epot1 + Epot2 + Epot3 + Epot4 + Epot5 + Epot6;

L = Ecin - Epot;

dLdthetap = jacobian(L, thetap);

n   =   jacobian(dLdthetap,   [theta'   thetap']')*[thetap'
thetapp']' - jacobian(L, theta)';
```

**MATLAB Function A.6:** Verifica del modello dinamico implementato in MATLAB attraverso la formulazione di Newton-Eulero.

```
function
[COPPIA_1_MATLAB,COPPIA_2_MATLAB,COPPIA_3_MATLAB,COPPIA_4_
MATLAB,COPPIA_5_MATLAB,COPPIA_6_MATLAB] =
fcn(SPOST_1,SPOST_2,SPOST_3,SPOST_4,SPOST_5,SPOST_6,VEL_1,
VEL_2,VEL_3,VEL_4,VEL_5,VEL_6,ACC_1,ACC_2,ACC_3,ACC_4,ACC_
5,ACC_6)



% masse ipotizzando densità 1000 kg/m^3 (ADAMS) [kg]

m1 = 5.5063351928;

m2 = 12.1234867663;

m3 = 2.8442715858;
```

```
m4 = 3.4171857808;

m5 = 1.4829756859;

m6 = 0.365751772;


g = 9.8; % [m/s^2]


% velocità angolare membro i rispetto terna i-1
(componente lungo z i-1)

% [rad/s]

theta1p = [0 0 VEL_1]';

theta2p = [0 0 VEL_2]';

theta3p = [0 0 VEL_3]';

theta4p = [0 0 VEL_4]';

theta5p = [0 0 VEL_5]';

theta6p = [0 0 VEL_6]';


% accelerazione angolare membro i rispetto terna i-1
(componente lungo z

% i-1) [rad/s^2]

theta1pp = [0 0 ACC_1]';

theta2pp = [0 0 ACC_2]';

theta3pp = [0 0 ACC_3]';

theta4pp = [0 0 ACC_4]';

theta5pp = [0 0 ACC_5]';

theta6pp = [0 0 ACC_6]';


%% Forward computation
```

```
% matrici di trasformazione e rotazione terne DH
% da {0} a {1}
Tz1=Rot_Tras_z(pi/2+SPOST_1,0.2503);
Tx1=Rot_Tras_x(pi/2,0);
T01=Tz1*Tx1;
R01=T01(1:3,1:3);

% da {1} a {2}
Tz2=Rot_Tras_z(pi/2+SPOST_2,0.2604);
Tx2=Rot_Tras_x(-pi,0.71);
T12=Tz2*Tx2;
T02=T01*T12;
R12=T12(1:3,1:3);
R02=R01*R12;

% da {2} a {3}
Tz3=Rot_Tras_z(0+SPOST_3,0.2604);
Tx3=Rot_Tras_x(-pi/2,0);
T23=Tz3*Tx3;
T03=T01*T12*T23;
R23=T23(1:3,1:3);
R03=R01*R12*R23;

% da {3} a {4}
Tz4=Rot_Tras_z(0+SPOST_4,0.540);
```

```
Tx4=Rot_Tras_x(-pi/2,0);
T34=Tz4*Tx4;
T04=T01*T12*T23*T34;
R34=T34(1:3,1:3);
R04=R01*R12*R23*R34;

% da {4} a {5}
Tz5=Rot_Tras_z(0+SPOST_5,0.150);
Tx5=Rot_Tras_x(pi/2,0);
T45=Tz5*Tx5;
T05=T01*T12*T23*T34*T45;
R45=T45(1:3,1:3);
R05=R01*R12*R23*R34*R45;

% da {5} a {6}
Tz6=Rot_Tras_z(0+SPOST_6,0.160);
Tx6=Rot_Tras_x(0,0);
T56=Tz6*Tx6;
R56=T56(1:3,1:3);

% da {6} a {0}
T06=T01*T12*T23*T34*T45*T56;
R06=R01*R12*R23*R34*R45*R56;

% posizione terna locale i rispetto a terna locale i-1
r01 = T01(1:3,4);
```

```
r12 = T12(1:3,4);

r23 = T23(1:3,4);

r34 = T34(1:3,4);

r45 = T45(1:3,4);

r56 = T56(1:3,4);


% posizione baricentro membro i rispetto terna i (ADAMS)

rc1 = [-4.55e-06 -0.04 1.77e-02]';

rc2 = [-0.37 1.83e-06 0.06]';

rc3 = [0 6.47e-03 0.04]';

rc4 = [0 0.16 1.68e-03]';

rc5 = [0 -0.01 0.02]';

rc6 = [0 0 -0.02]';


% parametri telaio

om0 = [0 0 0]';

om0_p = [0 0 0]';

ac0 = [0 0 0]';

ae0 = [0 0 0]';


% calcolo velocità angolari, accelerazioni angolari e
accelerazioni baricentri

om1 = om0+theta1p; % velocità angolare

om1_p = om0_p+cross(om0,theta1p)+theta1pp; % accelerazione
angolare

ae1 = ae0+cross(om1_p,r01)+cross(om1,cross(om1,r01)); %
accelerazione lineare centro terna locale
```

```
ac1 =
ae1+cross(om1_p,R01*rc1)+cross(om1,cross(om1,R01*rc1)); %
accelerazione lineare baricentro



om2 = om1+R01*theta2p;

om2_p = om1_p+cross(om1,R01*theta2p)+R01*theta2pp;

ae2 =
ae1+cross(om2_p,R01*r12)+cross(om2,cross(om2,R01*r12));

ac2 =
ae2+cross(om2_p,R02*rc2)+cross(om2,cross(om2,R02*rc2));



om3 = om2+R02*theta3p;

om3_p = om2_p+cross(om2,R02*theta3p)+R02*theta3pp;

ae3 =
ae2+cross(om3_p,R02*r23)+cross(om3,cross(om3,R02*r23));

ac3 =
ae3+cross(om3_p,R03*rc3)+cross(om3,cross(om3,R03*rc3));



om4 = om3+R03*theta4p;

om4_p = om3_p+cross(om3,R03*theta4p)+R03*theta4pp;

ae4 =
ae3+cross(om4_p,R03*r34)+cross(om4,cross(om4,R03*r34));

ac4 =
ae4+cross(om4_p,R04*rc4)+cross(om4,cross(om4,R04*rc4));



om5 = om4+R04*theta5p;

om5_p = om4_p+cross(om4,R04*theta5p)+R04*theta5pp;

ae5 =
ae4+cross(om5_p,R04*r45)+cross(om5,cross(om5,R04*r45));

ac5 =
ae5+cross(om5_p,R05*rc5)+cross(om5,cross(om5,R05*rc5));
```

```
om6 = om5+R05*theta6p;

om6_p = om5_p+cross(om5,R05*theta6p)+R05*theta6pp;

ae6 =
ae5+cross(om6_p,R05*r56)+cross(om6,cross(om6,R05*r56));

ac6 =
ae6+cross(om6_p,R06*rc6)+cross(om6,cross(om6,R06*rc6));


%% Backward computation


% parametri telaio

g0 = [0 0 -g]'; % gravità nel sistema globale

f7 = [0 0 0]'; % non agisce forza sull'ultimo membro

n7 = [0 0 0]'; % non agisce coppia sull'ultimo membro


% matrici d'inerzia baricentriche (ADAMS) [kg*m^2]

Ic6 = [3.8324992395E-04 0 0; 0 2.8780936509E-04 0; 0 0
2.8109242471E-04];

Ic5 = [4.0118412581E-03 0 0; 0 3.4152247178E-03 0; 0 0
2.521599251E-03];

Ic4 = [5.9410078458E-02 0 0; 0 5.9157896523E-02 0; 0 0
4.4861263316E-03];

Ic3 = [1.2854461399E-02 0 0; 0 1.2658132505E-02 0; 0 0
6.3701192822E-03];

Ic2 = [0.9589779179 0 0; 0 0.9481281818 0; 0 0
3.7316898245E-02];

Ic1 = [3.6640276653E-02 0 0; 0 3.259035102E-02 0; 0 0
2.1644758586E-02];
```

```
% Rgdhi matrice di rotazione di dhi rispetto a ground
(ADAMS)

% Rgcmi matrice di rotazione di cmi rispetto a ground
(ADAMS)

% Rci matrice di rotazione di cmi rispetto a dhi

Rgdh1 = [0.9978 3.469e-16 -0.06633; -0.6633 0 -0.9978;
-3.542e-16 1 0];

Rgcm1 = [0.9978 0.06205 -0.02312; -0.06622 0.9359 -0.3460;
0.0001675 0.3468 0.9379];

Rc1 = Rgdh1'*Rgcm1;

Rgdh2 = [6.702e-16 0.9978 0.06632; -5.889e-16 -0.06632
0.9978; 1 -7.076e-16 5.464e-16];

Rgcm2 = [0.9978 0.06651 0.001845; -0.06653 0.9974 0.02772;
3.112E-06 -0.02778 0.9996];

Rc2 = Rgdh2'*Rgcm2;

Rgdh3 = [3.986E-16 -0.06632 0.9978; -1.323E-16 -0.9978
-0.06632; 1 -1.055E-16 -4.064E-16];

Rgcm3 = [0.0001789 0.05465 -0.9985; -0.001475 0.9985
0.05465; 1 0.001482 9.801E-05];

Rc3 = Rgdh3'*Rgcm3;

Rgdh4 = [5.977E-16 -0.9978 -0.06632; -5.644E-16 0.06632
-0.9978; 1 6.339E-16 -5.238E-16];

Rgcm4 = [0.0002207 0.08337 -0.9965; 0.002913 0.9965
0.08337; 1 -0.002921 -2.297E-05];

Rc4 = Rgdh4'*Rgcm4;

Rgdh5 = [1.012E-15 -0.06632 0.9978; 1.585E-16 -0.9978
-0.06632; 1 2.252E-16 -9.993E-16];

Rgcm5 = [4.318E-05 0.5518 -0.8339; 3.251E-06 0.8339
0.5518; 1 -2.654E-05 3.422E-05];

Rc5 = Rgdh5'*Rgcm5;

Rgdh6 = [1.312E-15 -0.06632 0.9978; 1.896E-16 -0.9978
-0.06632; 1 2.762E-16 -1.296E-15];
```

```
Rgcm6 = [0.9975 0.0001359 -0.07024; -0.07024 1.457E-05
-0.9975; -0.0001345 1 2.408E-05];

Rc6 = Rgdh6'*Rgcm6;



% matrice inerzia trasformata da cm a terna locale i

I6 = Rc6*Ic6*Rc6';

I5 = Rc5*Ic5*Rc5';

I4 = Rc4*Ic4*Rc4';

I3 = Rc3*Ic3*Rc3';

I2 = Rc2*Ic2*Rc2';

I1 = Rc1*Ic1*Rc1';



% calcolo forze, inerzia membro i e coppie rispetto alla
terna 0

f6 = f7+m6*(ac6-g0); % fi-1,i

I60 = R06*I6*R06'; % inerzia nel sistema globale

n6 =
-cross(f6,R05*r56+R06*rc6)+n7+cross(f7,R06*rc6)+I60*om6_p+
cross(om6,I60*om6); % Ni-1,i



f5 = f6+m5*(ac5-g0);

I50 = R05*I5*R05';

n5 =
-cross(f5,R04*r45+R05*rc5)+n6+cross(f6,R05*rc5)+I50*om5_p+
cross(om5,I50*om5);



f4 = f5+m4*(ac4-g0);

I40 = R04*I4*R04';
```

```
n4 =
-cross(f4,R03*r34+R04*rc4)+n5+cross(f5,R04*rc4)+I40*om4_p+
cross(om4,I40*om4);



f3 = f4+m3*(ac3-g0);

I30 = R03*I3*R03';

n3 =
-cross(f3,R02*r23+R03*rc3)+n4+cross(f4,R03*rc3)+I30*om3_p+
cross(om3,I30*om3);



f2 = f3+m2*(ac2-g0);

I20 = R02*I2*R02';

n2 =
-cross(f2,R01*r12+R02*rc2)+n3+cross(f3,R02*rc2)+I20*om2_p+
cross(om2,I20*om2);



f1 = f2+m1*(ac1-g0);

I10 = R01*I1*R01';

n1 =
-cross(f1,r01+R01*rc1)+n2+cross(f2,R01*rc1)+I10*om1_p+cros
s(om1,I10*om1);



% coppia membro i espressa in terna i-1

n1l = n1;

n2l = R01'*n2;

n3l = R02'*n3;

n4l = R03'*n4;

n5l = R04'*n5;

n6l = R05'*n6;
```

```
% componenti delle coppie lungo asse z della terna i-1
COPPIA_1_MATLAB=n1l(3,1);
COPPIA_2_MATLAB=n2l(3,1);
COPPIA_3_MATLAB=n3l(3,1);
COPPIA_4_MATLAB=n4l(3,1);
COPPIA_5_MATLAB=n5l(3,1);
COPPIA_6_MATLAB=n6l(3,1);
```

**Script MATLAB A.7:** Verifica del modello dinamico implementato in MATLAB attraverso la formulazione di Lagrange.

```
% variabili costanti (DH e gravità)
a_1 = 0;
a_2 = 0.71;
a_3 = 0;
a_4 = 0;
a_5 = 0;
a_6 = 0;
d_1 = 0.2503;
d_2 = 0.2604;
d_3 = 0.2604;
d_4 = 0.540;
d_5 = 0.150;
d_6 = 0.160;
alpha_1 = pi/2;
```

```
alpha_2 = -pi;

alpha_3 = -pi/2;

alpha_4 = -pi/2;

alpha_5 = pi/2;

alpha_6 = 0;

g = 9.8;


% masse ipotizzando densità 1000 kg/m^3 (ADAMS) [kg]

m1 = 5.5063351928;

m2 = 12.1234867663;

m3 = 2.8442715858;

m4 = 3.4171857808;

m5 = 1.4829756859;

m6 = 0.365751772;


% matrici d'inerzia baricentriche (ADAMS) [kg*m^2]

Ic6 = [3.8324992395E-04 0 0; 0 2.8780936509E-04 0; 0 0
2.8109242471E-04];

Ic5 = [4.0118412581E-03 0 0; 0 3.4152247178E-03 0; 0 0
2.521599251E-03];

Ic4 = [5.9410078458E-02 0 0; 0 5.9157896523E-02 0; 0 0
4.4861263316E-03];

Ic3 = [1.2854461399E-02 0 0; 0 1.2658132505E-02 0; 0 0
6.3701192822E-03];

Ic2 = [0.9589779179 0 0; 0 0.9481281818 0; 0 0
3.7316898245E-02];

Ic1 = [3.6640276653E-02 0 0; 0 3.259035102E-02 0; 0 0
2.1644758586E-02];
```

```
% Rgdhi matrice di rotazione di dhi rispetto a ground
(ADAMS)

% Rgcmi matrice di rotazione di cmi rispetto a ground
(ADAMS)

% Rci matrice di rotazione di cmi rispetto a dhi

Rgdh1 = [0.9978 3.469e-16 -0.06633; -0.6633 0 -0.9978;
-3.542e-16 1 0];

Rgcm1 = [0.9978 0.06205 -0.02312; -0.06622 0.9359 -0.3460;
0.0001675 0.3468 0.9379];

Rc1 = Rgdh1'*Rgcm1;

Rgdh2 = [6.702e-16 0.9978 0.06632; -5.889e-16 -0.06632
0.9978; 1 -7.076e-16 5.464e-16];

Rgcm2 = [0.9978 0.06651 0.001845; -0.06653 0.9974 0.02772;
3.112E-06 -0.02778 0.9996];

Rc2 = Rgdh2'*Rgcm2;

Rgdh3 = [3.986E-16 -0.06632 0.9978; -1.323E-16 -0.9978
-0.06632; 1 -1.055E-16 -4.064E-16];

Rgcm3 = [0.0001789 0.05465 -0.9985; -0.001475 0.9985
0.05465; 1 0.001482 9.801E-05];

Rc3 = Rgdh3'*Rgcm3;

Rgdh4 = [5.977E-16 -0.9978 -0.06632; -5.644E-16 0.06632
-0.9978; 1 6.339E-16 -5.238E-16];

Rgcm4 = [0.0002207 0.08337 -0.9965; 0.002913 0.9965
0.08337; 1 -0.002921 -2.297E-05];

Rc4 = Rgdh4'*Rgcm4;

Rgdh5 = [1.012E-15 -0.06632 0.9978; 1.585E-16 -0.9978
-0.06632; 1 2.252E-16 -9.993E-16];

Rgcm5 = [4.318E-05 0.5518 -0.8339; 3.251E-06 0.8339
0.5518; 1 -2.654E-05 3.422E-05];

Rc5 = Rgdh5'*Rgcm5;

Rgdh6 = [1.312E-15 -0.06632 0.9978; 1.896E-16 -0.9978
-0.06632; 1 2.762E-16 -1.296E-15];
```

```
Rgcm6 = [0.9975 0.0001359 -0.07024; -0.07024 1.457E-05
-0.9975; -0.0001345 1 2.408E-05];

Rc6 = Rgdh6'*Rgcm6;


% matrice inerzia trasformata da cm a terna locale i

I6 = Rc6*Ic6*Rc6';

I5 = Rc5*Ic5*Rc5';

I4 = Rc4*Ic4*Rc4';

I3 = Rc3*Ic3*Rc3';

I2 = Rc2*Ic2*Rc2';

I1 = Rc1*Ic1*Rc1';


% posizione baricentro membro i rispetto terna i (ADAMS)

c1x = -4.55e-06;

c1y = -4e-02;

c1z = 1.77e-02;

c2x = -0.37;

c2y = 1.83e-06;

c2z = 0.06;

c3x = 0;

c3y = 6.47e-03;

c3z = 0.04;

c4x = 0;

c4y = 0.16;

c4z = 1.68e-03;

c5x = 0;

c5y = -0.01;
```

```
c5z = 0.02;

c6x = 0;

c6y = 0;

c6z = -0.02;


rc1 = [c1x c1y c1z]';

rc2 = [c2x c2y c2z]';

rc3 = [c3x c3y c3z]';

rc4 = [c4x c4y c4z]';

rc5 = [c5x c5y c5z]';

rc6 = [c6x c6y c6z]';


% variabili prese da ADAMS (spostamenti, velocità e
accelerazioni angolari e coppie)

theta_1 = out.SPOST_1.Data;

theta_2 = out.SPOST_2.Data;

theta_3 = out.SPOST_3.Data;

theta_4 = out.SPOST_4.Data;

theta_5 = out.SPOST_5.Data;

theta_6 = out.SPOST_6.Data;

theta1pz = out.VEL_1.Data;

theta2pz = out.VEL_2.Data;

theta3pz = out.VEL_3.Data;

theta4pz = out.VEL_4.Data;

theta5pz = out.VEL_5.Data;

theta6pz = out.VEL_6.Data;

theta1ppz = out.ACC_1.Data;
```

```
theta2ppz = out.ACC_2.Data;
theta3ppz = out.ACC_3.Data;
theta4ppz = out.ACC_4.Data;
theta5ppz = out.ACC_5.Data;
theta6ppz = out.ACC_6.Data;

n1 = out.COPPIA_1.Data;
n2 = out.COPPIA_2.Data;
n3 = out.COPPIA_3.Data;
n4 = out.COPPIA_4.Data;
n5 = out.COPPIA_5.Data;
n6 = out.COPPIA_6.Data;

n1tot = [];
n2tot = [];
n3tot = [];
n4tot = [];
n5tot = [];
n6tot = [];

% sostituzione valori numerici
for cont = 1:numel(theta_1)
    t1 = theta_1(cont);
    t2 = theta_2(cont);
    t3 = theta_3(cont);
    t4 = theta_4(cont);
```

```
    t5 = theta_5(cont);

    t6 = theta_6(cont);

    t1p = theta1pz(cont);

    t2p = theta2pz(cont);

    t3p = theta3pz(cont);

    t4p = theta4pz(cont);

    t5p = theta5pz(cont);

    t6p = theta6pz(cont);

    t1pp = theta1ppz(cont);

    t2pp = theta2ppz(cont);

    t3pp = theta3ppz(cont);

    t4pp = theta4ppz(cont);

    t5pp = theta5ppz(cont);

    t6pp = theta6ppz(cont);


    % n_function: funzione che sostituisce valori
      numerici all'espressione simbolica della coppia
      ottenuta dal modello impementato in MATLAB

    n =
n_function(I2(1,1),I3(1,1),I4(1,1),I5(1,1),I6(1,1),I2(1,2)
,I3(1,2),I4(1,2),I5(1,2),I6(1,2),I2(1,3),I3(1,3),I4(1,3),I
5(1,3),I6(1,3),I1(2,2),I2(2,2),I3(2,2),I4(2,2),I5(2,2),I6(
2,2),I2(2,3),I3(2,3),I4(2,3),I5(2,3),I6(2,3),I2(3,3),I3(3,
3),I4(3,3),I5(3,3),I6(3,3),a_2,c1x,c2x,c3x,c4x,c5x,c6x,c2y
,c3y,c4y,c5y,c6y,c1z,c2z,c3z,c4z,c5z,c6z,d_2,d_3,d_4,d_5,d
_6,g,m1,m2,m3,m4,m5,m6,t1,t2,t3,t4,t5,t6,t1p,t2p,t3p,t4p,t
5p,t6p,t1pp,t2pp,t3pp,t4pp,t5pp,t6pp);

    n1tot = [n1tot; n(1)];

    n2tot = [n2tot; n(2)];

    n3tot = [n3tot; n(3)];
```

```
    n4tot = [n4tot; n(4)];
    n5tot = [n5tot; n(5)];
    n6tot = [n6tot; n(6)];
end
% confronto grafico coppie ADAMS  e modello MATLAB
plot(n1);
hold on;
plot(n1tot);

plot(n2);
hold on;
plot(n2tot);

plot(n3);
hold on;
plot(n3tot);

plot(n4);
hold on;
plot(n4tot);

plot(n5);
hold on;
plot(n5tot);

plot(n6);
```

```
hold on;

plot(n6tot);
```

**Script MATLAB A.8:** calcolo della matrice di regressione simbolica, e salvataggio della funzione corrispondente, dai risultati ottenuti dalla modellazione dinamica del manipolatore attraverso la formulazione di Lagrange.

```
% vettore simbolico dei parametri da identificare

inc_inerz = [I11_1 I22_1 I33_1 I12_1 I13_1 I23_1 I11_2
I22_2 I33_2 I12_2 I13_2 I23_2 I11_3 I22_3 I33_3 I12_3
I13_3 I23_3 I11_4 I22_4 I33_4 I12_4 I13_4 I23_4 I11_5
I22_5 I33_5 I12_5 I13_5 I23_5 I11_6 I22_6 I33_6 I12_6
I13_6 I23_6 m1 m2 m3 m4 m5 m6]';


% regressore

J = jacobian(n, inc_inerz);


% funzione

matlabFunction(J,'file','J.m')
```

**Function MATLAB A.9:** valutazione numerica della matrice di regressione relativa al singolo test realizzato.

```
% variabili costanti

a_1 = 0;

a_2 = 0.71;

a_3 = 0;
```

```
a_4 = 0;
a_5 = 0;
a_6 = 0;
d_1 = 0.2503;
d_2 = 0.2604;
d_3 = 0.2604;
d_4 = 0.540;
d_5 = 0.150;
d_6 = 0.160;
alpha_1 = pi/2;
alpha_2 = -pi;
alpha_3 = -pi/2;
alpha_4 = -pi/2;
alpha_5 = pi/2;
alpha_6 = 0;
g = 9.8;

% posizioni baricentri supposte note (ADAMS)
c1x = -4.55e-06;
c1y = -0.04;
c1z = 1.77e-02;
c2x = -0.37;
c2y = 1.83e-06;
c2z = 0.06;
c3x = 0;
c3y = 6.47e-03;
```

```
c3z = 0.04;

c4x = 0;

c4y = 0.16;

c4z = 1.68e-03;

c5x = 0;

c5y = -0.01;

c5z = 0.02;

c6x = 0;

c6y = 0;

c6z = -0.02;


% variabili prese da ADAMS (spostamenti, velocità e
accelerazioni angolari e coppie)

theta_1 = out.SPOST_1.Data;

theta_2 = out.SPOST_2.Data;

theta_3 = out.SPOST_3.Data;

theta_4 = out.SPOST_4.Data;

theta_5 = out.SPOST_5.Data;

theta_6 = out.SPOST_6.Data;

theta1pz = out.VEL_1.Data;

theta2pz = out.VEL_2.Data;

theta3pz = out.VEL_3.Data;

theta4pz = out.VEL_4.Data;

theta5pz = out.VEL_5.Data;

theta6pz = out.VEL_6.Data;

theta1ppz = out.ACC_1.Data;

theta2ppz = out.ACC_2.Data;
```

```
theta3ppz = out.ACC_3.Data;
theta4ppz = out.ACC_4.Data;
theta5ppz = out.ACC_5.Data;
theta6ppz = out.ACC_6.Data;

n1 = out.COPPIA_1.Data;
n2 = out.COPPIA_2.Data;
n3 = out.COPPIA_3.Data;
n4 = out.COPPIA_4.Data;
n5 = out.COPPIA_5.Data;
n6 = out.COPPIA_6.Data;

Jtest = [];

for cont = 1:numel(theta_1)

        t1 = theta_1(cont);
        t2 = theta_2(cont);
        t3 = theta_3(cont);
        t4 = theta_4(cont);
        t5 = theta_5(cont);
        t6 = theta_6(cont);
        t1p = theta1pz(cont);
        t2p = theta2pz(cont);
        t3p = theta3pz(cont);
        t4p = theta4pz(cont);
```

```
        t5p = theta5pz(cont);

        t6p = theta6pz(cont);

        t1pp = theta1ppz(cont);

        t2pp = theta2ppz(cont);

        t3pp = theta3ppz(cont);

        t4pp = theta4ppz(cont);

        t5pp = theta5ppz(cont);

        t6pp = theta6ppz(cont);


        Jtest = [Jtest;
J(a_2,c1x,c2x,c3x,c4x,c5x,c6x,c2y,c3y,c4y,c5y,c6y,c1z,c2z,
c3z,c4z,c5z,c6z,d_2,d_3,d_4,d_5,d_6,g,t1,t2,t3,t4,t5,t6,t1
p,t2p,t3p,t4p,t5p,t6p,t1pp,t2pp,t3pp,t4pp,t5pp,t6pp)];

end


ntest = [n1; n2; n3; n4; n5; n6];
```

# Bibliografia


[1] FANUC, “CRX series flyer”, 2022.

[2] FANUC, Scheda tecnica “MDS-04019”, 2022.

[3] FANUC, “CRX-10iA - Operator's Manual”, 2020.

[4] S. El Zaatari, M. Marei, W. Li, and Z. Usman, “Cobot programming for collaborative industrial tasks: An overview”, *Robotics and Autonomous Systems*, vol. 116, pp. 162– 180, 2019.

[5] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: modelling, planning and control*. Springer Science & Business Media, 2010.

[6] M. Dev Anad, T. Selvaraj, and S. Kumanan, “Parameter identification of robot arm using artificial neural networks”, *National Conference on Computational Intelligence*, 2007.

[7] G. Michalos, S. Makris, P. Tsarouchi, T. Guasch, D. Kontovrakis, and G. Chryssolouris, “Design considerations for safe human-robot collaborative workplaces”, *Procedia CIrP*, vol. 37, pp. 248–253, 2015.

[8] DIRETTIVA 2006/42/CE DEL PARLAMENTO EUROPEO E DEL CONSIGLIO, *Gazzetta ufficiale dell’Unione europea,* 2006.

[9] I. O. for Standardization (ISO), “Safety of machinery—general principles for design—risk assessment and risk reduction”, 2010.

[10] I. 10218-1, “Robots and robotic devices–safety requirements for industrial robots–part 1 robots”, 2011.

[11] I. 10218-2, “Robots and robotic devices–safety requirements for industrial robots–part 2: Robot systems and integration”, 2011.

[12] I. ISO, “Ts 15066 (2016): robots and robotic devices–collaborative robots”, Geneva: International Organization for Standardization, 2016.

[13] I. ISO, “ISO 8373: robots and robotic devices–vocabulary”, 2021.

[14] A. Raviola, A. De Martin, R. Guida, S. Pastorelli, S. Mauro, M. Sorli, “Identification of a UR5 collaborative robot dynamic parameters”, *The 30th International Conference on Robotics*, Alpe-Adria-Danube Region, pp. 69-77, 2021.

[15] A. Kinsheel and Z. Taha, “Identification of the parameters of robot manipulators dynamics about an operating point using perturbed dynamics,” *11th Int. Conf. Control. Autom. Robot. Vision*, Singapore, pp. 144–148, 2010.

[16] Azeddien Kinsheel, “Robust least square estimation of the CRS A465 robot arm’s dynamic model parameters,” *J. Mech. Eng. Res.*, vol. 4, no. 3, pp. 89–99, 2012.

[17] V. Villani, F. Pini, F. Leali and C. Secchi, “Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications”, *Mechatronics*, pp. 248–266, 2018.

[18] B. Siciliano, O. Khatib, *Springer Handbook of Robotics,* seconda edizione, Springer International Publishing, 2016.

[19] J. J. Craig, *Introduction to robotics: Mechanics and Control*, terza edizione, Pearson Higher Education, 2014.

[20] T. Brezina, Z. Hadas, and J. Vetiska, “Using of co-simulation ADAMS-SIMULINK for development of mechatronic systems”, *14th International Conference Mechatronika*, pp. 59–64, 2011.

[21] L. Biagiotti, R. Zanasi, *Identificazione di sistemi dinamici*, 2011.

[22] C. Hansen, “Regularization Tools: A Matlab Package for Analysis and Solution of Discrete Ill-Posed Problems”, *Numer. Algo. 46*, pp. 189–194, 2007.

[23] L. Angel, J. Viola, “Parametric identification of a delta type parallel robot”, *Colombian Conference on Robotics and Automation (CCRA)*, 2016.