ACHILLE
CANNAVALE

APPUNTI

ROBOTICA

2024

CIAO! QUESTI APPUNTI SONO FRUTTO DEL MIO STUDIO E DELLA MIA INTERPRETAZIONE, QUINDI POTREBBERO CONTENERE ERRORI, SVISTE O COSE MIGLIORABILI. BUONO STUDIO📚✍️

Riassunto Sistemi Robotici

Achille Cannavale

Corso di Laurea Magistrale in Ingegneria Informatica Intelligenza Artificiale e Robotica

Università degli Studi di Cassino e del Lazio Meridionale

Indice

Capitolo 1

Kinematics

La **cinematica** è la relazione che sussiste tra la posizione dei **giunti** e la posizione e orientamento dell'**organo terminale**.
Indichiamo la **terna di riferimento ortonormale** come:

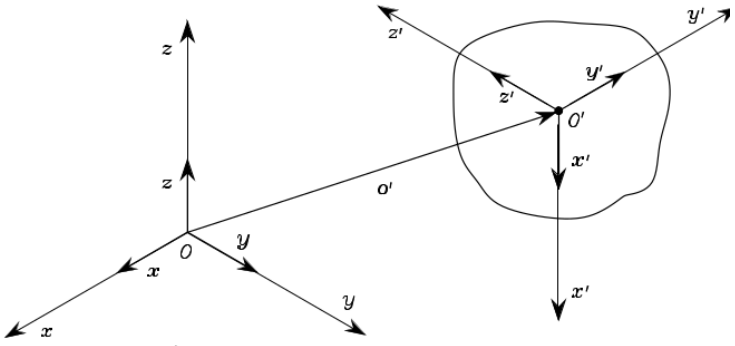
$$\Sigma, \mathbf{o}, \mathbf{x}, \mathbf{y}, \mathbf{z}$$

E un corpo fermo con una sua **terna solidale**:

$$\Sigma', \mathbf{o}', \mathbf{x}', \mathbf{y}', \mathbf{z}'$$

dove:

$$\mathbf{o} = \begin{bmatrix} \mathbf{o}'_x \\ \mathbf{o}'_y \\ \mathbf{o}'_z \end{bmatrix}$$



Possiamo scomporre i versori della terna Σ' **proiettandoli** sulla terna Σ:

$$\begin{cases} \mathbf{x}' = \mathbf{x}'_x\mathbf{x} + \mathbf{x}'_y\mathbf{y} + \mathbf{x}'_z\mathbf{z} \\ \mathbf{y}' = \mathbf{y}'_x\mathbf{x} + \mathbf{y}'_y\mathbf{y} + \mathbf{y}'_z\mathbf{z} \\ \mathbf{z}' = \mathbf{z}'_x\mathbf{x} + \mathbf{z}'_y\mathbf{y} + \mathbf{z}'_z\mathbf{z} \end{cases}$$

1.1 Matrice di Rotazione

$$\mathbf{R} = \begin{bmatrix} \mathbf{x}'^T\mathbf{x} & \mathbf{y}'^T\mathbf{x} & \mathbf{z}'^T\mathbf{x} \\ \mathbf{x}'^T\mathbf{y} & \mathbf{y}'^T\mathbf{y} & \mathbf{z}'^T\mathbf{y} \\ \mathbf{x}'^T\mathbf{z} & \mathbf{y}'^T\mathbf{z} & \mathbf{z}'^T\mathbf{z} \end{bmatrix} \in \mathcal{R}^{\ni \times \ni}$$

Vincoli:

- **Ortogonalità**: $\mathbf{x}'^T\mathbf{y}' = 0$ etc
- **Norma Unitaria**: $\mathbf{x}'^T\mathbf{x} = 1, ... \implies \mathbf{R}^T\mathbf{R} = \mathbf{I}_3, \mathbf{R}^T = \mathbf{R}^{-1}$
- $det(\mathbf{R}) = \pm 1$
- $\mathbf{R} \in so(m)$

Possiamo avere diverse interpretazioni della **Matrice di Rotazione**:

- **Operatore di Rotazione**: $\Sigma \longrightarrow \Sigma'$ ($\mathbf{R}$ rappresenta la rotazione che serve fare attorno ad un asse)
- **Vector Representation**
  - Se abbiamo due terne diverse, uno stesso punto $\mathbf{p}$ può essere rappresentato così:

$$\mathbf{p} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \mathbf{p'} = \begin{bmatrix} p'_x \\ p'_y \\ p'_z \end{bmatrix}$$

- **Vector Rotation**
  - Dati i vettori $\mathbf{p}$ e $\mathbf{p}' = \mathbf{R}\mathbf{p}$ espressi nella stessa terna, il vettore $\mathbf{p}$ avrà la stessa norma di $\mathbf{p}'$ e sarà ruotato rispetto a $\mathbf{p}'$ in base alla matrice $\mathbf{R}$,

1.2 Composizione Matrici di Rotazione

Supponiamo di avere tre terne con l'origine in comune: $\Sigma_0, \Sigma_1, \Sigma_2$ e dei vettori espressi in queste terne: $\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2$, definiamo $\mathbf{R}_i^j$ la matrice di rotazione che porta da Σ_i a Σ_j:[1]

$$\implies \mathbf{p}^0 = \mathbf{R}_1^0\mathbf{p}^1,\ \mathbf{p}^1 = \mathbf{R}_2^1\mathbf{p}^2,\ \mathbf{p}^0 = \mathbf{R}_1^0\mathbf{R}_2^1\mathbf{p}^2$$

Le matrici di rotazione sono una rappresentazione **ridondante**, avendo 9 parametri e 6 vincoli.

[1] Nelle **fixed frame**, tutte le operazioni sono invertite

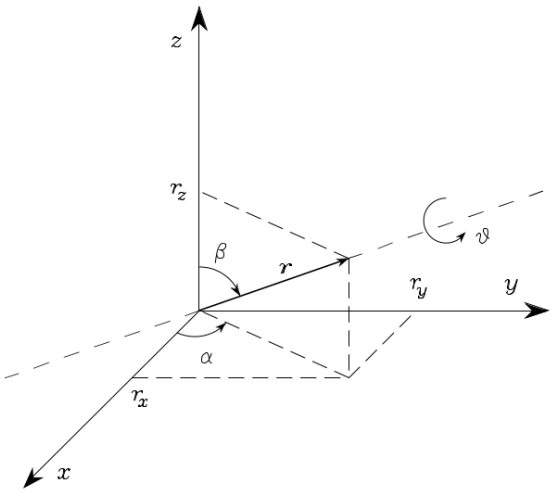
1.3 Angoli di Eulero

Ho 3 parametri indipendenti e quindi scrivo la matrice di rotazione come 3 rotazioni elementari, escludendo le combinazioni banali, avremo così 12 possibili angoli di eulero.
Esempi:

- **zyz** in terna corrente
  - per $\theta = 0$ *or* π non possiamo fare il problema inverso da **R** agli angoli di eulero (singolarità di rappresentazione)
- **Roll, pitch, yaw** in terna fissa

1.4 Asse-Angolo

Questa **rappresentazione non è minima**, in quanto ha 4 parametri. Per $\theta = 0$ c'è **singolarità di rappresentazione**.



1.5 Quaternioni

Il **Quaternione** è una rappresentazione a 4 parametri:

$$Q = \{\mu, \boldsymbol{\epsilon}\}$$

Con questa rappresentazione, posso:

- Convertire partendo da **Asse-Angolo** e da **Matrice di Rotazione**
- **Non ci sono singolarità di rappresentazione**

1.6 Trasformazioni Omogenee

Le **trasformazioni omogenee** sono una combinazione di traslazione e rotazione. Si utilizza la **Matrice di Trasformazione Omogenea**:

$$\mathbf{A} = \begin{bmatrix} \mathbf{R}_1^0 & \mathbf{o}_1^0 \\ \mathbf{0}^T & \mathbf{1} \end{bmatrix} \in \mathcal{R}^{4\times 4}$$

Tuttavia, per questioni matematiche, dobbiamo definire i vettori in questo modo:

$$\tilde{\mathbf{p}} = \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} \in \mathcal{R}^4 \implies \tilde{\mathbf{p}}^0 = \mathbf{A}_1^0 \mathbf{A}_2^1 ... \mathbf{A}_n^{n-1} \tilde{p}^n$$

1.7 Direct Kinematics

La **cinematica diretta** è il legame che unisce la **posizione e orientamento dell'organo terminale** e le **variabili di giunto**.
Definiamo quindi il **vettore delle variabili di giunto**:

$$\mathbf{q} \longrightarrow \mathbf{T}_e^b(\mathbf{q})$$

1.8 Denavit-Hartenberg Convention

Questa **convenzione** funge da strumento condiviso per definire le variabili di giunto.
Sono 4 numeri, di cui 1 sarà la variabile di giunto, mentre le altre rimangono costanti:

- a_i
- θ_i, per giunti **rotoidali**
- α_i
- d_i per giunti **prismatici**

1.9 Spazio dei Giunti e Spazio Operativo

$$\mathbf{q} = \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_n \end{bmatrix} \in \mathcal{R}^n, \quad q_i = \begin{cases} d_i, & Prismatico \\ \theta_i, & Rotoidale \end{cases}$$

Definiamo lo **Spazio Operativo** come:

$$\boldsymbol{\chi_e} = \begin{bmatrix} \mathbf{p}_e \\ \boldsymbol{\phi_e} \end{bmatrix} \in \mathcal{R}^m$$

con $\mathbf{p}_e$ la posizione e $\boldsymbol{\phi_e}$ l'orientamento dell'organo terminale. Quindi **l'equazione di cinematica diretta** è la seguente:

$$\boldsymbol{x_e} = \mathbf{K}(\mathbf{q})$$

1.10 Inverse Kinematics

In questo caso, conosco **posizione e orientamento dell'organo terminale** e voglio conoscere la **posizione e l'orientamento dei giunti**.

$$\boldsymbol{\chi_e} \implies \mathbf{q}$$

Sfruttando una soluzione algebrica ottengo un sistema di **eq. non lineari**.

# DoF	# IK sol. for arbitrary **pos**
< 3	0
3	> 0, finite in some part of the ws
> 3	∞ in some part of the ws

# DoF	# IK sol. for arbitrary **pos+or**
< 6	0
6	> 0, finite in some part of the ws
> 6	∞ in some part of the ws

Capitolo 2

Differential Kinematics

Per cinematica differenziale intendiamo il legame tra la velocità dei giunti e la velocità lineare e angolare dell'organo terminale.

2.1 Jacobiano Geometrico

Abbiamo visto come la cinematica diretta, partendo dal vettore $\mathbf{q}$ ci calcola la matrice di Trasformazione Omogenea:

$$\boldsymbol{T_e(q)} = \begin{bmatrix} \mathbf{R}_e(\mathbf{q}) & \mathbf{p}_e(\mathbf{q}) \\ \mathbf{0}^T & \mathbf{1} \end{bmatrix} \in \mathcal{R}^{4\times 4}$$

A questo punto è necessario calcolare lo Jacobiano Geometrico $\mathbf{J}(\mathbf{q}) \in \mathcal{R}^{6\times n}$:

$$\begin{cases} \dot{\mathbf{p}}_e = \mathbf{J}_p(\mathbf{q})\dot{\mathbf{q}} \in \mathcal{R}^3 \\ \boldsymbol{\omega_e} = \mathbf{J}_o(\mathbf{q})\dot{\mathbf{q}} \in \mathcal{R}^3 \end{cases} \implies$$

$$\implies \mathbf{v}_e = \begin{bmatrix} \dot{\mathbf{p}}_e \\ \boldsymbol{\omega}_e \end{bmatrix} = \begin{bmatrix} \mathbf{J}_p(\mathbf{q}) \\ \mathbf{J}_o(\mathbf{q}) \end{bmatrix} \cdot \dot{\mathbf{q}} = \mathbf{J}(\mathbf{q}) \cdot \dot{\mathbf{q}}$$

Tuttavia ci servono prima alcuni strumenti matematici per calcolare lo Jacobiano:

2.1.1 Derivata temporale di una matrice di rotazione

$$\begin{cases} \text{Velocità di un punto} \quad \dot{\mathbf{p}} = \dfrac{dp(t)}{dt} \\ \text{Velocità di un punto in un moto rotoidale} \quad \dot{\mathbf{p}} = \boldsymbol{\omega} \times \mathbf{p} \end{cases}$$

Dato che $\mathbf{R}(t)\mathbf{R}^T(t) = \mathbf{I}$ derivo:

$$\dot{\mathbf{R}}(t)\mathbf{R}^T(t) + \mathbf{R}(t)\dot{\mathbf{R}}^T(t) = 0$$

Chiamo:

$$\mathbf{S}(t) = \dot{\mathbf{R}}\mathbf{R}^T(t) \implies \mathbf{S}(t) + \mathbf{S}^T(t) = 0^1$$

Ora moltiplico $\mathbf{S}$ per $\mathbf{R}$:

$$\dot{\mathbf{R}}(t) = \mathbf{S}(\omega(t))\mathbf{R}(t)$$

Prendiamo ora ad esempio un punto $\mathbf{p}$:

$$\mathbf{p}^0 = \mathbf{o}_1^0 + \mathbf{R}_1^0\mathbf{p}^1$$

derivo:

$$\begin{aligned}\dot{\mathbf{p}}^0 &= \dot{\mathbf{o}}_1^0 + \dot{\mathbf{R}}_1^0\mathbf{p}^1 + \mathbf{R}_1^0\dot{\mathbf{p}}^1 = \\ &= \dot{\mathbf{o}}_1^0 + \mathbf{S}(\omega_1^0)\mathbf{R}_1^0\mathbf{p}^1 + \underbrace{\cancel{\mathbf{R}_1^0\dot{\mathbf{p}}^1}}_{p^1 \text{ è solidale con } \Sigma^1} = \\ &= \dot{\mathbf{o}}_1^0 + \underbrace{\boldsymbol{\omega}_\mathbf{1}^\mathbf{0} \times \mathbf{r}_1^0}_{\text{effetto giradischi}}\end{aligned}$$

ora possiamo ricavare le velocità lineari:

$$\mathbf{p}_i = \mathbf{p}_{i-1} + \mathbf{R}_{i-1}\mathbf{r}_{i-1,i}^{i-1}$$

derivo:

$$\begin{aligned}\dot{\mathbf{p}} &= \dot{\mathbf{p}}_{i-1} + \mathbf{R}_{i-1}\dot{\mathbf{r}}_{i-1,i}^{i-1} + \boldsymbol{\omega}_{i-1} \times \mathbf{R}_{i-1}\mathbf{r}_{i-1,i}^{i-1} \\ &= \dot{\mathbf{p}}_{i-1} + \mathbf{v}_{i-1,i} + \boldsymbol{\omega}_{i-1} \times \mathbf{r}_{i-1,i}\end{aligned}$$

Per le velocità angolari invece (salto i calcoli):

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

Ora è possibile calcolare lo Jacobiano Geometrico:

$$\mathrm{J} = \begin{bmatrix}\mathbf{J}_{p_1} & \cdots & \mathbf{J}_{p_n} \\ \mathbf{J}_{o_1} & \cdots & \mathbf{J}_{o_n}\end{bmatrix}$$

In particolare, è importante notare come al Cinematica Diretta fosse un legame non lineare, in quanto aveva il dominio in $\mathbf{q}$ con seni e coseni e il codominio in $\begin{bmatrix}\mathbf{p}_e \\ \boldsymbol{\psi}_e\end{bmatrix} = \mathbf{K}(\mathbf{q})$.
Mentre il legame che crea lo Jacobiano Geometrico è lineare, in quanto ha dominio in $\dot{\mathbf{q}}$ ed è confiugrazione dipendente.

2.2 Jacobiano Analitico

$$\begin{cases} \dot{\mathbf{p}}_e = \mathbf{J}_p(\mathbf{q}) \cdot \dot{\mathbf{q}} \\ \dot{\boldsymbol{\phi}}_e = \dfrac{\partial \boldsymbol{\phi}}{\partial \mathbf{q}} = \mathbf{J}_\phi(\mathbf{q}) \cdot \dot{\mathbf{q}} \end{cases}$$

$$\dot{\boldsymbol{\chi}} = \begin{bmatrix} \dot{\mathbf{p}}_e \\ \dot{\boldsymbol{\phi}}_e \end{bmatrix} = \begin{bmatrix} \mathbf{J}_p(\mathbf{q}) \\ \mathbf{J}_\phi(\mathbf{q}) \end{bmatrix} \cdot \dot{\mathbf{q}} = \underbrace{J_A(\mathbf{q})}_{\frac{\partial \mathbf{K}(\mathbf{q})}{\partial \mathbf{q}}} \cdot \dot{\mathbf{q}}$$

IMPORTANTE: $\dot{\boldsymbol{\phi}}_e \neq \boldsymbol{\omega_e}$

Tuttavia risulta facile calcolare una serie di matrici 3×3, diverse per ogni rappresentazione, che mi permettono di stabilire un legame tra $\dot{\boldsymbol{\phi}}_e$ e $\boldsymbol{\omega_e}$:

$$\mathbf{J} = \mathbf{T}_A(\boldsymbol{\phi})\mathbf{J}_A$$

2.3 Singolarità Cinematiche

Data la relazione:

$$\mathbf{v}_e = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}$$

Se $\mathbf{J}$ perde rango, siamo di fronte a delle singolarità cinematiche, che producono:

- Perdità di mobilità
- Ho infinite soluzioni al problema cinematico
- Quando arrivo vicino alla singolarità comincio ad avere già problemi

Esempio:

$$\dot{\mathbf{q}} = \mathbf{J}^{-1}\mathbf{v}_e = \frac{AGG(\mathbf{J})}{DET(\mathbf{J})}\mathbf{v}_e$$

con $DET(\mathbf{J}) << 1$ dato che è sceso di rango.
Le singolarità si trovano ai confini dello spazio di lavoro principalmente, ma alle volte anche all'interno.
Un altro indizio di singolarità sono i vettori $\mathbf{J}_p$ paralleli tra di loro $\implies$ scesa di rango.

2.4 Pseudoinversa

Quando ho più equazioni che ingognite, dobbiamo formulare una soluzione a minimizzazione dell'errore:

$$min_{\mathbf{x}} \frac{1}{2} ||\mathbf{y} - \mathbf{A}\mathbf{x}||^2$$

$$\mathbf{x} = \mathbf{A}^{\dagger}\mathbf{y} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{y}$$

2.5 SVD

Data una matrice $\mathbf{M} \in \mathcal{R}^{m \times n}$, la decomposizione a valor singolare è la seguente:

$$\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$$

con:

- $\mathbf{U}$ base ortonormale di $\mathcal{R}^m$
- $\mathbf{V}$ base ortonormale di $\mathcal{R}^n$
- $\mathbf{\Sigma}$ contiene i valori singolari di $\mathbf{M}$ sulla diagonale

2.6 Forma Quadratica

Una forma quadratica è una funzione scalare:

$$f(\mathbf{x}) = \mathbf{x}^T\mathbf{A} \quad \mathbf{x} \in \mathcal{R}$$

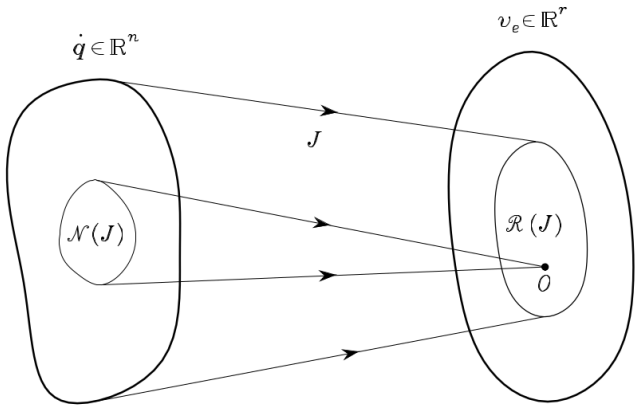
2.7 Inversione della Cinematica Differenziale

L'equazione della cinematica differenziale è la seguente:

$$\mathbf{v}_e = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}$$

Dobbiamo quindi invertire questa relazione, in quanto, conosco la velocità che voglio dare al mio end-effector ($\mathbf{v}_e$), conosco la confiugrazione dei giunti ($\mathbf{J}(\mathbf{q})$), che velocità devo dare ai giunti? ($\dot{\mathbf{q}}$)

2.8 Ridondanza



Quindi ogni vettore nello spazio giunti, può essere rappresentato come:

$$\dot{\mathbf{q}} = \underbrace{\dot{\mathbf{q}}^*}_{\text{Qualsiasi soluzione}} + \underbrace{\mathbf{p}\dot{\mathbf{q}}_a}_{\text{qualsiasi cosa nello spazio nullo}}$$

A questo punto possiamo procedere all'inversione, dato un robot ridondante:

$$\mathbf{v}_e = \mathbf{J}\dot{\mathbf{q}} \quad con\ \mathbf{J} \in \mathcal{R}^{r \times n},\ r < n$$

$$\implies \min_{\mathbf{q}} g(\dot{\mathbf{q}}) = \frac{1}{2}\dot{\mathbf{q}}^T\dot{\mathbf{q}} > 0 \quad s.t.\ \mathbf{v}_e = \mathbf{J}\dot{\mathbf{q}}$$

La cui soluzione è la Pseudoinversa:

$$\dot{\mathbf{q}} = \underbrace{\mathbf{J}^T(\mathbf{J}\mathbf{J}^T)\mathbf{J}^{-1}}_{\mathbf{J}^\dagger}\mathbf{v}_e$$

La ridondanza mi è utile anche a evitare Singolarità cinematiche.

$$g''(\dot{\mathbf{q}}) = ||\mathbf{v}_e - \mathbf{J}\dot{\mathbf{q}}||^2 + \mathbf{K}^2||\dot{\mathbf{q}}||^2$$

Ovvero: puoi commettere un piccolo errore nell'inversione della ginematica, a patto che le velocità dei giunti rimangano basse.

$$\implies \mathbf{J}^* = \mathbf{J}^T(\mathbf{J}\mathbf{J}^T + \mathbf{K}^2\mathbf{I})^{-1}$$

2.9 Algoritmi per l'Inversione Cinematica: Jacobian Inversion

Caso $r = n$

$$\dot{\mathbf{q}} = \mathbf{J}_A^{-1}(\mathbf{q})(\dot{\mathbf{x}}_d + \mathbf{K}\mathbf{e}) \implies \dot{\mathbf{e}} + \mathbf{K}\mathbf{e} = \mathbf{0}$$

Abbiamo così linearizzato la dinamica dell'errore.
Caso $n > r$

$$\dot{\mathbf{q}} = \mathbf{J}_A^{\dagger}(\dot{\mathbf{x}}_d + \mathbf{K}\mathbf{e}) + (\mathbf{I} - \mathbf{J}_A^{\dagger}\mathbf{J}_A)\dot{\mathbf{q}}_a$$

2.10 Algoritmi per l'Inversione Cinematica: Jacobian Transpose

Costruisco una funzione di Lyapunov:

$$V(\mathbf{e}) = \frac{1}{2}\mathbf{e}^T\mathbf{K}\mathbf{e}$$

dove:

$$V(\mathbf{e}) > 0 \quad \forall \mathbf{e} \neq \mathbf{0}$$

$$\begin{aligned}\dot{V}(\mathbf{e}) &= \mathbf{e}^T\mathbf{K}\dot{\mathbf{x}}_d - \mathbf{e}^T\mathbf{K}\dot{\mathbf{x}}_e = \\ &= \mathbf{e}^T\mathbf{K}\dot{\mathbf{x}}_d - \mathbf{e}^T\mathbf{K}\mathbf{J}_A(\mathbf{q})\dot{\mathbf{q}}\end{aligned}$$

Scegliendo $\dot{\mathbf{q}} = \mathbf{J}_A^T(\mathbf{q})\mathbf{K}\mathbf{e}$ ottengo:

$$= \mathbf{e}^T\mathbf{K}\dot{\mathbf{x}}_d - \mathbf{e}^T\mathbf{K}\mathbf{J}_A(\mathbf{q})\mathbf{J}_A^T(\mathbf{q})\mathbf{K}\mathbf{e}$$

Nel caso in cui fossimo in presenza di una Singolarità Cinematica e volessimo dare al robot una velocità che non può darci, rimarrà bloccato.

$$\mathcal{N}(\mathbf{J}_A^T) \quad \text{è la direzione proibita}$$

2.11 Inversione della Cinematica: Orientamento

Possiamo dimostrare che, definendo l'errore di orientamento in questo modo:

$$\mathbf{e}_o = \delta\epsilon = \eta_e(\mathbf{q})\epsilon_d - \eta_d\epsilon_e(\mathbf{q}) - \mathbf{S}(\epsilon_d)\epsilon_e(\mathbf{q}) \in \mathcal{R}^3$$

Posso inserirlo nella formula di Inversione della Cinematica:

$$\dot{\mathbf{q}} = \mathbf{J}^{-1}(\mathbf{q}) \begin{bmatrix} \dot{\mathbf{p}}_d + \mathbf{K}_p\mathbf{e}_p \\ \boldsymbol{\omega}_d + \mathbf{K}_o\mathbf{e}_o \end{bmatrix}$$

La cosa positiva è che non devo passare per la Matrice di Rotazione. Quindi riassumendo, avremo:

$$\dot{\mathbf{q}} = \mathbf{J}_A^{\dagger}(\mathbf{q}) \begin{bmatrix} \dot{\mathbf{p}}_d + \mathbf{K}_p \mathbf{e}_p \\ \boldsymbol{\phi}_d + \mathbf{K}_o \mathbf{e}_o \end{bmatrix} \quad \mathbf{e}_o = \boldsymbol{\phi}_d - \boldsymbol{\phi}_e$$

$$\dot{\mathbf{q}} = \mathbf{J}^{\dagger}(\mathbf{q}) \begin{bmatrix} \dot{\mathbf{p}}_d + \mathbf{K}_p \mathbf{e}_p \\ \boldsymbol{\omega}_d + \mathbf{K}_o \mathbf{e}_o \end{bmatrix} \quad \mathbf{e}_o = \delta\boldsymbol{\epsilon}$$

Capitolo 3

Statica

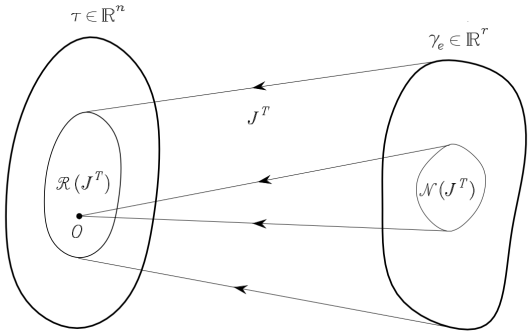
La Statica consiste nel trovare un legame tra le forze e i momenti che agiscono all'organo terminale e le coppie ai giunti.
Definiamo:

$$\boldsymbol{\gamma}_e = \begin{bmatrix} \mathbf{f}_e \\ \boldsymbol{\mu}_e \end{bmatrix} \in \mathcal{R}^6 \qquad \text{Forze e momenti all'organo terminale}$$

$$\boldsymbol{\tau} \in \mathcal{R}^n \qquad \text{Forze e coppie ai giunti}$$

Il nostro robot sarà in equilibrio statico se e solo se il lavoro elementare dell'organo terminale sia uguale al lavoro elementare dei giunti:

$$dW_\tau = dW_\gamma \quad \forall \delta \mathbf{q}$$
$$\downarrow$$
$$\boldsymbol{\tau} = \mathbf{J}^T(\mathbf{q})\boldsymbol{\gamma}_e$$



$$\mathcal{N}(\boldsymbol{J}) \equiv \mathcal{R}^{\perp}(\boldsymbol{J}^{\mathrm{T}}) \qquad \mathcal{R}(\boldsymbol{J}) \equiv \mathcal{N}^{\perp}(\boldsymbol{J}^{\mathrm{T}})$$

3.1 Dualità Cinetostatica

$$\mathbf{v}_e = \mathbf{J}\dot{\mathbf{q}}$$
$$\boldsymbol{\tau} = \mathbf{J}^T\boldsymbol{\gamma}_e$$

Caso $n = m$ non ridondante

$$\implies \mathbf{J} \quad \text{a rango pieno}$$
$$\implies \mathcal{N}(\mathbf{J}) = \mathbf{0}, \quad \mathcal{N}(\mathbf{J}^T) = \mathbf{0}$$

Caso $n > m$ ridondante

$$\implies \mathbf{J} \quad \text{a rango pieno}$$
$$\implies \mathcal{N}(\mathbf{J}) \neq \mathbf{0}, \quad \mathcal{N}(\mathbf{J}^T) = \mathbf{0}$$

Caso singolarità

$$\implies \mathcal{N}(\mathbf{J}) \neq \mathbf{0}, \quad \mathcal{N}(\mathbf{J}^T) \neq \mathbf{0}$$

La dualità cinetostatica ci dice quindi:

$$\boldsymbol{\tau} = \mathbf{J}^T \boldsymbol{\gamma}_e$$
$$\dot{\mathbf{q}} = \mathbf{J}^T(\mathbf{q})\mathbf{K}\mathbf{e}$$

Come possiamo vedere, le forze e i momenti all'organo terminale, sono quiparati all'errore di posizione.
Moltiplicando l'errore per un guadagno $\mathbf{K}$ ottengo una molla virtuale che applica una forza all'end effector verso la posizione desiderata. Questa forza viene proiettata ai giunti con $\mathbf{J}^T \implies \boldsymbol{\tau} = \dot{\mathbf{q}}$.

3.2 Ellissoidi di Manipolabilità

Dato un vettore $\mathbf{x} \in \mathcal{R}^n \implies \mathbf{x}^T\mathbf{W}^{-1}\mathbf{x} = 1$ è un ellisse in $\mathcal{R}^n$ con:

- Autovettori di $\mathbf{W}\mathbf{W}^T$ come assi principali
- Autovalori $\sigma_i = \sqrt{\lambda_i(\mathbf{W}\mathbf{W}^T)}$ come lunghezza degli assi

Considerando una sfera nello spazio giunti:

$$\dot{\mathbf{q}}^T\dot{\mathbf{q}} = 1$$
$$\implies \dot{\mathbf{q}} = \mathbf{J}^\dagger(\mathbf{q})\mathbf{v}_e$$
$$\implies \mathbf{v}_e^T(\mathbf{J}(\mathbf{q})\mathbf{J}^T(\mathbf{q}))^{-1}\mathbf{v}_e = 1$$

Quindi una sfera nello spazio giunti è un'ellisse nello spazio cartesiano:

- Autovettori di $\mathbf{J}\mathbf{J}^T$ come assi principali
- Autovalori $\sigma_i = \sqrt{\lambda_i(\mathbf{J}\mathbf{J}^T)}$ come lunghezza degli assi

Possiamo ora fare un ragionamento analogo con le forze, ottenendo:

$$\boldsymbol{\gamma}_e^T(\mathbf{J}(\mathbf{q})\mathbf{J}^T(\mathbf{q}))\boldsymbol{\gamma}_e = 1$$

La geometria ci dice che $(\mathbf{J}(\mathbf{q})\mathbf{J}^T(\mathbf{q}))^{-1}$ e $(\mathbf{J}(\mathbf{q})\mathbf{J}^T(\mathbf{q}))$ hanno gli stessi autovalori ma autovettori opposti.
Quindi la migliore direzione per usare il manipolatore come moltiplicatore di velocità è la peggiore per usarlo come moltiplicatore di forza e viceversa.

Capitolo 4

Dynamics

Il modello matematico della dinamica, nello spazio dei giunti è il seguente:

$$\mathbf{B}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{F}_v\dot{\mathbf{q}} + f_s(\dot{\mathbf{q}}) + \mathbf{g}(\mathbf{q}) = \boldsymbol{\tau} - \mathbf{J}^T(\mathbf{q})\mathbf{h}$$

- $\mathbf{B}(\mathbf{q})\ddot{\mathbf{q}} \implies$ Matrice delle masse per accelerazione dei giunti
- $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} \implies$ Termini di Coriolis e centripeti
- $\mathbf{F}_v\dot{\mathbf{q}} + f_s(\dot{\mathbf{q}}) \implies$ Attrito
- $\mathbf{g}(\mathbf{q}) \implies$ Gravità
- $\boldsymbol{\tau} \implies$ Attuazione
- $-\mathbf{J}^T(\mathbf{q})\mathbf{h} \implies$ Interazione con l'esterno

Useremo il modello in due momenti diversi:

- Dinamica Diretta; calcolo dell'accelerazione $\ddot{\mathbf{q}}$, utile per le simulazioni
- Dinamica Inversa: calcolo della $\boldsymbol{\tau}$, conoscendo $\ddot{\mathbf{q}}(t), \dot{\mathbf{q}}(t), \mathbf{q}(t), \mathbf{h}(t)$, utile nella pianificazine del moto e del controllo

Capitolo 5

Trajectory Planning

La pianificazione della traiettoria consiste nella generazione di input che saranno usati come reference per il controllo del moto. L'algoritmo di pianificazione ha come input:

- Definizione del percorso
- Vincoli del percorso
- Vincoli della dinamica del robot

In output avrò:

- Le traiettorie dei giunti o dell'organo terminale espresse come sequenza temporale della posizione, velocità e accelerazione

Le traiettorie possono essere nello spazio cartesiano o nello spazio giunti.

5.1 Joint Space Trajectories

Devo generare $\mathbf{q}(t) \in \mathcal{R}^n$ che interpola $\mathbf{q}_i$ e $\mathbf{q}_f$. Noi studieremo solo il caso di un movimento punto-punto.
Si dimostra che se vogliamo minimizzare il quadrato della coppia di forze:

$$min \int_0^{t_f} \boldsymbol{\tau}^2(t)dt$$

La cui soluzione è un polinomio ordine 2 nella velocità.
La traiettoria che viene utilizzata è quella con profilo trapezoidale.

5.2 Operational Space Trajectories

Per quanto riguarda la traiettoria della posizione, si genera una curva parametrica, alla quale sarà possibile applicare una determinata legge oraria, come ad esempio un profilo trapezoidale.

Per quanto riguarda l'orientamento il discorso è più complesso data una rotazione iniziale $\mathbf{R}_i$ e una finale $\mathbf{R}_f$ ottengo la rotazione delta così:

$$\mathbf{R}_f^i = \mathbf{R}_i^T \mathbf{R}_f$$

Da questa matrice estraggo la coppia Asse-Angolo:

$$\begin{cases} \theta_f \\ \mathbf{r} \end{cases}$$

così potremo dare a $\theta(t)$ una legge oraria in modo tale da avere:

$$\begin{cases} \theta(0) = 0 \\ \theta(t_f) = \theta_f \end{cases}$$

$$\mathbf{R}(t) = \mathbf{R}_i \mathbf{R}_f^i(\theta(t))$$
$$\boldsymbol{\omega}^i = \dot{\theta}_i \mathbf{t}$$

Capitolo 6

Motion Control

Consideriamo il modello di un pendolo:

$$\mathbf{I}\ddot{\boldsymbol{\theta}} + mgl\ sin(\boldsymbol{\theta}) = \boldsymbol{\tau}$$

6.1 PD

Il $\boldsymbol{\tau}$ è il mio grado di libertà. Quale possiamo scegliere?
$\boldsymbol{\tau} = \mathbf{K}_p\tilde{\theta} + \mathbf{K}_v\dot{\tilde{\theta}}$ è semplice, ma l'errore a regime è diverso da zero ed è guidato dall'errore. In aggiunta, è configurazione dipendente dato che c'è $sin(\theta)$.

6.2 PID

$$\boldsymbol{\tau} = \mathbf{K}_p\tilde{\theta} + \mathbf{K}_v\dot{\tilde{\theta}} + \mathbf{K}_i\int\tilde{\theta}$$

Qui abbiamo introdotto un'azione integrale, che fa tendere l'errore a regime a zero. TUttavia è sempre guidato dall'errore e sempre configurazione dipendente. In aggiunta, l'azione integrale da problemi in cambi repentini di configurazione.

6.3 Open-Loop Model Based

$$\boldsymbol{\tau} = \hat{\mathbf{I}}\ddot{\boldsymbol{\theta}}_d + \hat{m}gl\ sin(\theta)$$

In questa configurazione, in teoria, avrei errore a regime nullo, tuttavia dovrei calcolare il modello e i parametri.

6.4 Feedback Linearization

$$\boldsymbol{\tau} = \hat{\mathbf{I}}\ddot{\boldsymbol{\theta}}_d + \hat{m}gl\ sin(\theta) + \mathbf{K}_p\tilde{\theta} + \mathbf{K}_v\hat{\theta} + \mathbf{K}_i \int \tilde{\theta}$$

Si dice che questo metodo "cancella la dinamica". Ora vado a sostituire:

$$\cancel{\mathbf{I}\ddot{\boldsymbol{\theta}} + mgl\ sin(\boldsymbol{\theta})} = \cancel{\hat{\mathbf{I}}\ddot{\boldsymbol{\theta}}_d + \hat{m}gl\ sin(\theta)} + \mathbf{K}_p\tilde{\theta} + \mathbf{K}_v\hat{\theta} + \mathbf{K}_i \int \tilde{\theta}$$

Tuttavia non conosciamo alla perfezione i parametri:

$$\implies \mathbf{K}_i = \tilde{m}gl\ sin(\theta)$$

6.5 Adaptive

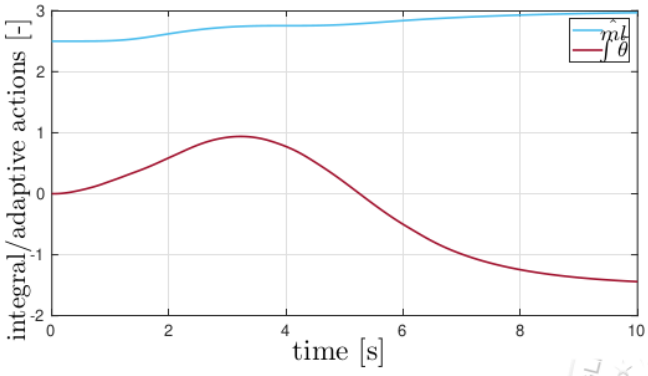
$$\tau = \hat{I}\ddot{\theta} + \hat{m}gl\ sin(\theta) + K_p\tilde{\theta} + K_v\dot{\tilde{\theta}}$$

Adattiamo i parametri del modello stavolta:

$$\dot{\hat{m}}l = \text{non ci interessa}$$

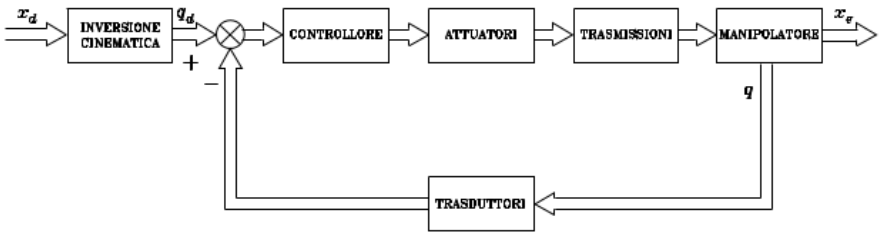
Se facciamo questo, non sto compensando l'errore del controllo, ma sto stimando meglio il modello matematico.

further insight on the adaptive/integral term
the adaptive term tries to estimate the physical nature of the mismatching while the integral term *blindly* considers everything as disturbance

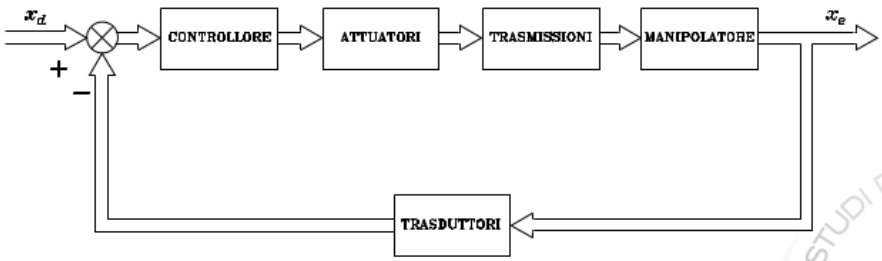


6.6 Due Paradigmi Di Controllo di un Robot

Joint space control



Operational space control



6.7 Joint Space Control

Quello che segue è il nostro modello dinamico:

$$\mathbf{B}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{F}_v\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \boldsymbol{\tau}$$

Per controllo intendiamo calcolare $\boldsymbol{\tau}$ in modo tale che $\mathbf{q}$ tenda a $\mathbf{q}_d$.
I nostri robot hanno solitamente delle trasmissioni con rapporti di riduzione, che implicano:

$$\mathbf{K}_r\mathbf{q} = \mathbf{q}_m \implies \boldsymbol{\tau}_m = \mathbf{K}_r^{-1}\boldsymbol{\tau}$$

6.8 Controllo Decentralizzato

Mi riscrivo il modello dinamico esplicitando τ_m al posto di τ. Sarà configurazione dipendente è non lineare. A questo punto calcolo un'inerzia media:

$$B(q) = \tilde{B} + \Delta B(q)$$

Quindi sostituendo nel modello matematico posso separare le due parti: un controllo con l'inerzia media e tutto il resto lo considero un disturbo e lo trascuro.
Questo disturbo sarà piccolo se $\ddot{q}_m$ e $\dot{q}_m$ sono piccoli. Quindi il mio controllo sarà il più semplice possibile.

6.9 Controllo Centralizzato: PD + Compensazione gravità

Assegno $\dot{\mathbf{q}}_d$ e $\tilde{\mathbf{q}}_d = 0$.
Quindi il mio scopo sarà quello di mandare a zero lo stato formato da: $[\tilde{\mathbf{q}}^T \dot{\mathbf{q}}^T]$ con $\tilde{\mathbf{q}} = \mathbf{q}_d - \mathbf{q}$.
Costruisco una funzione di Lyapunov partendo da questo stato:

$$V(\dot{\mathbf{q}}, \tilde{\mathbf{q}}) = \frac{1}{2}\dot{\mathbf{q}}^T \mathbf{B}(\mathbf{q})\dot{\mathbf{q}} + \frac{1}{2}\tilde{\mathbf{q}}^T \mathbf{K}_p \tilde{\mathbf{q}} > 0$$

Derivandola ottengo una funzione dipendente dall'ingresso u. Tuttavia per Lyapunov, $\dot{V} < 0 \implies$ devo imporre una u che mi rende negativa la $\dot{V}$:

$$u = g(\mathbf{q}) + \mathbf{K}_p \tilde{\mathbf{q}} \mathbf{K}_D \dot{\mathbf{q}} \implies \dot{V} \leq 0$$

Quindi quando questo sistema evolverà, il controllo si fermerà solo quando $\dot{\mathbf{q}} = 0 \ \forall \tilde{q}$, quindi devo cercare l'equilibrio, e questo si fa mettendo il controllo nel modello:

$$\underbrace{\mathbf{B}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{F}\dot{\mathbf{q}} + g(\mathbf{q})}_{Modello} = \underbrace{g(\mathbf{q}) + \mathbf{K}_p \tilde{\mathbf{q}} - \mathbf{K}_D \dot{\mathbf{q}}}_{\mathbf{u}}$$

La teoria mi dice che $\dot{\mathbf{q}} = \ddot{\mathbf{q}} = \mathbf{0}$ allora:

$$0 = \mathbf{K}_p \tilde{\mathbf{q}} \implies \tilde{\mathbf{q}} = \mathbf{q}_d - \mathbf{q} = 0$$

Ruolo dei guadagni:

- $\mathbf{K}_p$ rigidità della molla virtuale sull'errore
- $\mathbf{K}_D$ attrito virtuale

La compensazione della gravità non sarà mai perfetta, non rendendoci possibile l'eliminazione della $g(\mathbf{q})$ algebricamente. Tuttavia la letteratura ci dice che se $\mathbf{K}_p >> 1$ funziona lo stesso, ma non arrivo alla configurazione desiderata. Per questo si introduce una "piccola azione integrale" detti anche "Meccanismi di Anti Windup".

6.10 Controllo a Dinamica Inversa

Per semplicità impongo:

$$n(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{F}\dot{\mathbf{q}} + g(\mathbf{q})$$

Quindi il modello sarà:

$$\mathbf{B}(\mathbf{q})\ddot{\mathbf{q}} + n(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{u}$$

Posso pensare di linearizzare il modello, ovvero compensare tutti i termini:

$$\mathbf{u} = \mathbf{B}(\mathbf{q})\mathbf{y} + \mathbf{n}(\mathbf{q}, \dot{\mathbf{q}})$$
$$\implies \ddot{\mathbf{q}} = \mathbf{y}$$

Dove $\mathbf{y}$ è un segnale ausiliario, ovvero un mio grado di libertà.
Una possibile scelta di $\mathbf{y}$ per stabilizare è:

$$\mathbf{y} = \ddot{\mathbf{q}}_d + \mathbf{K}_d\dot{\tilde{\mathbf{q}}} + \mathbf{K}_p\tilde{\mathbf{q}} \implies \ddot{\tilde{\mathbf{q}}} + \mathbf{K}_d\dot{\tilde{\mathbf{q}}} + \mathbf{K}_p\tilde{\mathbf{q}} = 0$$

Che è un'equazione differenziale lineare con coefficienti che sono parametri di progetto. Tuttavia la compensazione della dinamica non è mai perfetta:

$$\mathbf{u} = \hat{\mathbf{B}}(\mathbf{q})(\ddot{\mathbf{q}}_d + \mathbf{K}_D\dot{\tilde{\mathbf{q}}} + \mathbf{K}_p\tilde{\mathbf{q}}) + \hat{\mathbf{n}}(\mathbf{q}, \dot{\mathbf{q}})$$

6.11 Centralized Control Adaptive

Abbiamo sempre lo stesso modello:

$$\mathbf{B}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, dq)\dot{\mathbf{q}} + \mathbf{F}\dot{\mathbf{q}} + g(\mathbf{q}) = \mathbf{Y}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})\boldsymbol{\pi} = \mathbf{u}$$

Dove:

- $\mathbf{Y}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$ è La Matrice Regressore
- $\boldsymbol{\pi}$ è il vettore dei parametri dinamici

Posso scrivere il modello come Matrice Regressore per il Vettore dei Parametri Dinamici grazie alla linearità dei parametri.
A questo punto costruisco una legge di controllo con cui compenso la dinamica:

$$\mathbf{u} = \mathbf{B}(\mathbf{q})\ddot{\mathbf{q}}_r + \mathbf{C}(\mathbf{q}, dq)\dot{\mathbf{q}}_r + \mathbf{F}\dot{\mathbf{q}}_r + g(\mathbf{q}) + \mathbf{K}_D\boldsymbol{\sigma}$$

Esplicitiamo i parametri:

$$\begin{cases} \ddot{\mathbf{q}}_r = \ddot{\mathbf{q}}_d + \mathbf{\Lambda}\tilde{\mathbf{q}} \\ \dot{\mathbf{q}}_r = \dot{\mathbf{q}}_d + \mathbf{\Lambda}\tilde{\mathbf{q}} \\ \boldsymbol{\sigma} = \dot{\mathbf{q}}_r - \dot{\mathbf{q}} = \dot{\tilde{\mathbf{q}}} + \mathbf{\Lambda}\tilde{\mathbf{q}} \end{cases}$$

Ottenendo così:

$$\mathbf{B}(\mathbf{q})\dot{\boldsymbol{\sigma}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\boldsymbol{\sigma} + \mathbf{F}\boldsymbol{\sigma} + \mathbf{K}_D\boldsymbol{\sigma} = \mathbf{0}$$

Per capire se questa legge di controllo funziona, possiamo usare Lyapunov, usando come stato $[\boldsymbol{\sigma}, \tilde{\mathbf{q}}]$:

$$V(\boldsymbol{\sigma}, \tilde{\mathbf{q}}) = \frac{1}{2}\boldsymbol{\sigma}^T\mathbf{B}(\mathbf{q})\boldsymbol{\sigma} + \frac{1}{2}\tilde{\mathbf{q}}^T\tilde{\mathbf{q}} > 0 \quad \forall \boldsymbol{\sigma}\tilde{\mathbf{q}} \neq \mathbf{0}$$

$$\implies \dot{V} = -\boldsymbol{\sigma}^T\mathbf{F}\boldsymbol{\sigma} - \dot{\tilde{\mathbf{q}}}^T\mathbf{K}_D\dot{\tilde{\mathbf{q}}}\mathbf{\Lambda}\mathbf{K}_D\mathbf{\Lambda}\tilde{\mathbf{q}} < 0$$

Però ancora una volta non ho la conoscenza perfetta del modello, quindi riscrivo il mio controllo come:

$$\begin{aligned} \mathbf{u} &= \hat{\mathbf{B}}(\mathbf{q})\ddot{\mathbf{q}}_r + \hat{\mathbf{C}}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}_r + \hat{\mathbf{F}}\dot{\mathbf{q}}_r + \hat{g} + \mathbf{K}_D\boldsymbol{\sigma} = \\ &= \mathbf{Y}(\mathbf{q}, \dot{\mathbf{q}}, \dot{\mathbf{q}}_r, \ddot{\mathbf{q}}_r)\hat{\boldsymbol{\pi}} + \mathbf{K}_D\boldsymbol{\sigma} \end{aligned}$$

A questo punto prendo l'errore del vettore dei parametri dinamici e lo pongo come elemento dello "stato esteso":

$$V(\boldsymbol{\sigma}, \tilde{\mathbf{q}}, \tilde{\boldsymbol{\pi}}) = \frac{1}{2}\boldsymbol{\sigma}^T\mathbf{B}(\mathbf{q})\boldsymbol{\sigma} + \tilde{\mathbf{q}}^T\mathbf{\Lambda}\mathbf{K}_D\tilde{\mathbf{q}} + \frac{1}{2}\tilde{\boldsymbol{\pi}}^T\mathbf{K}_\pi\tilde{\boldsymbol{\pi}} > 0 \quad \forall \boldsymbol{\sigma}, \tilde{\mathbf{q}}, \tilde{\boldsymbol{\pi}} \neq \mathbf{0}$$

Alla fine se scelgo un'opportuna dinamica per il vettore dei parametri dinamici stimato:

$$\dot{\hat{\boldsymbol{\pi}}} = \mathbf{K}_\pi^{-1}\mathbf{Y}^T(\mathbf{q}, \dot{\mathbf{q}}, \dot{\mathbf{q}}_r, \ddot{\mathbf{q}}_r)\boldsymbol{\sigma}$$

Importante da notare che:

$$\begin{cases} \tilde{\mathbf{q}} \longrightarrow \mathbf{0} \\ \mathbf{Y}(\mathbf{q}, \dot{\mathbf{q}}, \dot{\mathbf{q}}_r, \ddot{\mathbf{q}}_r)(\dot{\boldsymbol{\pi}} - \boldsymbol{\pi}) \longrightarrow \mathbf{0} \end{cases}$$

Quindi se il Regressore ha un nullo è possibile che anche se il vettore dei parametri dinamici sia sbagliato, io abbia lo stesso errore nullo.

6.12 Operational Space Control: PD + Gravity Compensation

Usiamo Lyapunov per ricavarci il nostro controllo:

$$V(\dot{\mathbf{q}}, \tilde{\mathbf{x}}) = \frac{1}{2}\dot{\mathbf{q}}^T\mathbf{B}(\mathbf{q})\dot{\mathbf{q}} + \frac{1}{2}\tilde{\mathbf{x}}^T\mathbf{K}_p\tilde{\mathbf{x}} > 0 \quad \forall \dot{\mathbf{q}}, \tilde{\mathbf{x}} \neq \mathbf{0}$$

$$\implies \dot{V} = -\dot{\mathbf{q}}^T\mathbf{F}\dot{\mathbf{q}} + \dot{\mathbf{q}}^T(\mathbf{u} - g(\mathbf{q}) - \mathbf{J}_A^T(\mathbf{q})\mathbf{K}_p\tilde{\mathbf{x}})$$

Esplicitiamo la $\mathbf{u}$:

$$\mathbf{u} = g(\mathbf{q}) + \mathbf{J}_A^T(\mathbf{q})\mathbf{K}_P\tilde{\mathbf{x}} - \underbrace{\mathbf{J}_A^T(\mathbf{q})\mathbf{K}_D\mathbf{J}_A(\mathbf{q})\dot{\mathbf{q}}}_{\text{attrito arbitrario}}$$

$$\implies \dot{V} = -\dot{\mathbf{q}}^T\mathbf{F}\dot{\mathbf{q}} - \dot{\mathbf{q}}^T\mathbf{J}_A^T(\mathbf{q})\mathbf{K}_D\mathbf{J}_A(\mathbf{q})\dot{\mathbf{q}} \leq \mathbf{0}$$

Quindi posso dimostrare che:

$$\dot{\mathbf{q}} \longrightarrow \mathbf{0} \implies \dot{V} = 0 \implies \mathbf{J}_A^T(\mathbf{q})\mathbf{K}_p\tilde{\mathbf{x}} = \mathbf{0}$$

Quindi non $\tilde{\mathbf{x}} \longrightarrow \mathbf{0}$, sempre per il fatto del nullo della matrice. In questo caso il nullo è l'insieme di tutte le forze che si scaricano sulla struttura.

6.13 Inverse Dynamics Control

Dato il modello:

$$\mathbf{B}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{n}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{u}$$

Applico il controllo:

$$\mathbf{u} = \mathbf{B}(\mathbf{q})\mathbf{Y} + \mathbf{n}(\mathbf{q}, \dot{\mathbf{q}})$$

con:

$$\mathbf{Y} = \mathbf{J}_A^{-1}(\mathbf{q})(\ddot{\mathbf{x}}_d + \mathbf{K}_D\dot{\tilde{\mathbf{x}}} + \mathbf{K}_p\tilde{\mathbf{x}} - \mathbf{J}_A(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}})$$

$$\implies \ddot{\mathbf{x}} + \mathbf{K}_D\dot{\tilde{\mathbf{x}}} + K_p\tilde{\mathbf{x}} = 0$$

Che è un'equazione differenziale lineare. Così facendo ho eliminato tutto il modello.

Capitolo 7

Interaction Control (fatto malissimo, vedi i video)

7.1 Esempio 1 DOF

Consideriamo un punto materiale sul quale voglio applicare un controllo posizionale:

$$m\ddot{x} = u$$

$$\implies u = m(\ddot{x}_d + K_v\dot{\tilde{x}} + K_p\tilde{x})$$
$$\implies \ddot{\tilde{x}} + K_v\dot{\tilde{x}} + K_p\tilde{x} = 0$$

Ora aggiungiamo un'integrazione con l'ambiente:

$$m\ddot{x} = u - f$$
$$\implies m\ddot{\tilde{x}} + mK_v\dot{\tilde{x}} + mK_p\tilde{x} = f$$

Se raggiungo il regime, il mio sistema si comporta come una maolla, ma il cui K non posso controllare, essendo moltiplicato per m:

$$mK_p\tilde{x} = f$$

7.2 Fully Dimensionality Case

Assumiamo la forza scambiata con l'ambiente $\mathbf{h}_e \in \mathcal{R}^6$:

$$\mathbf{B}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{F}\dot{\mathbf{q}} + g(\mathbf{q}) = \underbrace{\mathbf{u} - \mathbf{J}^T(\mathbf{q})\mathbf{h}_e}_{\text{Dualità Cinetostatica}}$$

Applichiamo PD + GC:

$$\mathbf{u} = g(\mathbf{q}) + \mathbf{J}_A^T\mathbf{K}_p\tilde{\mathbf{x}} - \mathbf{J}_A^T\mathbf{K}_D\underbrace{\mathbf{J}_A(\mathbf{q})\dot{\mathbf{q}}}_{\dot{\mathbf{x}}_e}$$

Se vado a regime rimane solo:

$$\mathbf{J}_A^T \mathbf{K}_P \tilde{\mathbf{x}} = \mathbf{J}^T(\mathbf{q}) \mathbf{h}_e$$

Assumendo di avere uno Jacobiano a rango pieno, ricoridamo che possiamo passare da uno all'altro attraverso una matrice di trasformazione:

$$\mathbf{J}(\mathbf{q}) = \mathbf{T}_A(\phi) \mathbf{J}_A(\mathbf{q}) = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{T}(\phi) \end{bmatrix} \mathbf{J}_A(\mathbf{q})$$

Quindi sostituendo ed esplicitando $\dot{\mathbf{x}}$ ottengo:

$$\dot{\mathbf{x}} = \mathbf{K}_P^{-1} \underbrace{\mathbf{T}_A^T(\mathbf{x}) \mathbf{h}_e}_{\mathbf{h}_A} = \mathbf{K}_P^{-1} \mathbf{h}_A$$

Vediamo ora la considerazione che possiamo fare:

- Il robot si comporta sempre come una molla
- La sua cedevolezza attiva è $\mathbf{K}_P^{-1}$
- Il problema risiede nel termine $\mathbf{h}_A$, che è l'effetto dell'interazione con forze esterne sulla molla. Il problema è che sarà configurazione dipendente, in quanto il termine $\mathbf{h}_e$ è moltiplicato con $T_A^T(\mathbf{x})$ che dipende dalla configurazione finale.

7.3 Cedevolezza Attiva

Questo tipo di controllo viene ottenuto attraverso il controllore, quindi è possibile tarare questa cedevolezza attraverso i guadagni che utilizzo nel controllore.
Ciò che voglio evitare però, quando uso un controllo puramente posizionale, è la dipendenza della cedevolezza dalla configurazione.
Per risolvere questo problema, devo ridefinire l'errore, nel seguente modo:

$$\tilde{\mathbf{x}} = - \begin{bmatrix} \mathbf{o}_{d,e}^d \\ \boldsymbol{\phi}_{d,e} \end{bmatrix}$$

Quindi calcolo l'errore rispetto alla terna desiderata all'organo terminale e non rispetto alla terna base.
Ora calcoliamo la dinamica dell'errore:

$$\dot{\tilde{\mathbf{x}}} = -\mathbf{T}_A^{-1}(\phi_{d,e}) \begin{bmatrix} \mathbf{R}_d^T & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_d^T \end{bmatrix} \mathbf{J}(\mathbf{q}) \dot{\mathbf{q}} = -\mathbf{J}_{A_d}(\mathbf{q}, \tilde{\mathbf{x}}) \dot{\mathbf{q}}$$

Dove $\mathbf{J}_{A_d}$ sarà lo Jacobiano nel nuovo errore.
A questo punto possiamo implementare questa nuova legge di controllo con ad esempio la PD + GC. Valutiamo l'analisi di stabilità con Lyapunov e dimostriamo che a regime avremo che la relazione che c'è tra forze all'organo temrinale e lo spostamento non è configurazione dipendente.
Mettiamo ora il caso in cui l'ambiente si comporti come una molla:

$$\mathbf{h}_e = \mathbf{K}_d \mathbf{x}_{r,e}$$

Dove $\mathbf{K}$ è semidefinita positiva.
Mi fermo perchè non so che scrivere.

7.4 Controllo di Impedenza

7.4.1 1-DOF

Stesso discorso di prima ma con l'interazione con l'ambiente:

$$m\ddot{x} = u - f$$

A questo punto scelgo l'ingresso come:

$$u = \frac{m}{m_d}(m_d\ddot{x}_d + k_v\dot{\tilde{x}} + k_p\tilde{x} - f) + f$$

Quindi questo vuol dire che dal punto di vista concettuale che:

- La forza che scambio con l'esterno la voglio misurare
- E la compenso

Quindi sto rendendo il mio robot molto rigido. Sostituendo ottengo:

$$m_d\ddot{\tilde{x}} + k_v\dot{\tilde{x}} + k_p\tilde{x} = f$$

Quindi diventa un sistema massa-molla-smorzatore virtuale sotto l'azione della forza esterna.

- Quindi più che dare una posizione desiderata, stiamo dando un comportamento desiderato
- In questo modo possiamo gestire le forze in transitorio (cosa che non abbiamo mai fatto prima)
- Per quanto riguarda il regime avrò sempre un "compromesso tra molle" come nel caso prima

7.4.2 Esempio di Implementazione

Nel caso in cui adottassimo una Feedback Linearization e usassimo quanto visto prima, otterremo come risultato:

$$\mathbf{M}_d\ddot{\tilde{\mathbf{x}}} + \mathbf{K}_D\dot{\tilde{\mathbf{x}}} + \mathbf{K}_v\dot{\tilde{\mathbf{x}}} = \mathbf{M}_d\mathbf{B}_A^{-1}(\mathbf{q})\mathbf{h}_A$$

Come possiamo vedere, con questo approccio non mi libero del problema delle equazioni dipendenti dalla configurazione del robot.
Per ovviare a questo problema, modifichiamo il controllo:

$$\mathbf{u} = \mathbf{B}(\mathbf{q})\mathbf{y} + \mathbf{n}(\mathbf{q}, dq) + \mathbf{J}^T(\mathbf{q})\mathbf{h}_e$$

Sostituendo e facendo dei conti ottengo:

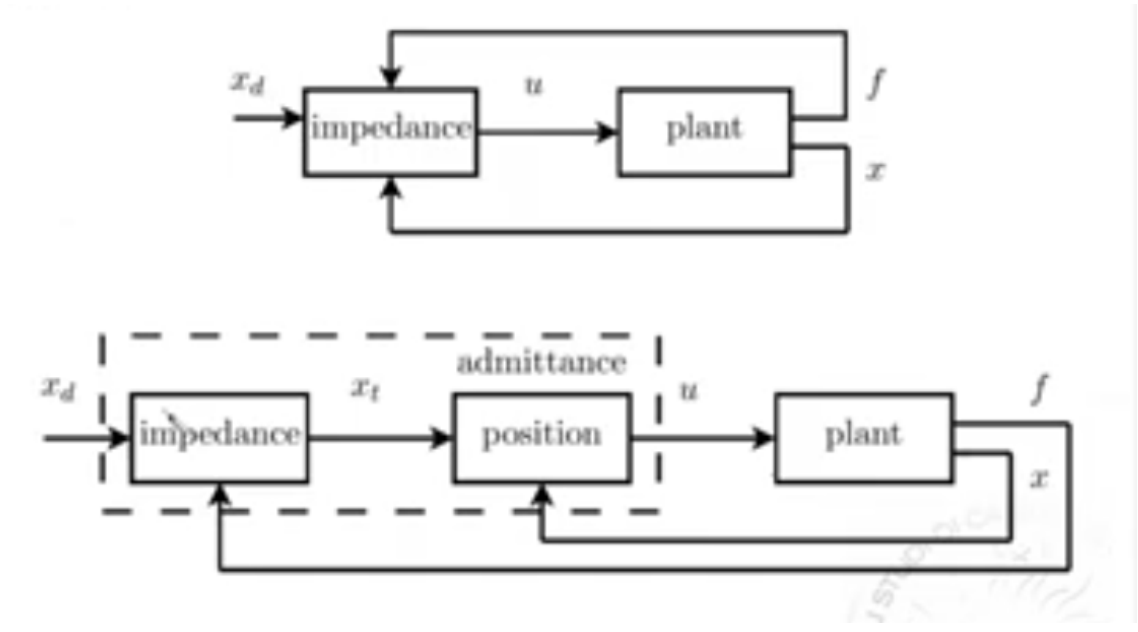
$$\mathbf{M}_d\ddot{\tilde{\mathbf{x}}} + \mathbf{K}_D\dot{\tilde{\mathbf{x}}} + \mathbf{K}_P\tilde{\mathbf{x}} = \mathbf{h}_A$$

A questo punto ho ancora la dipendenza dell'orientamento nella $\mathbf{h}_A$, quindi definiamo anceh qui un nuovo errore, ottenendo questo nuovo controllo:

$$\mathbf{y} = \mathbf{J}_{A_d}^{-1}\mathbf{M}_d^{-1}(\mathbf{K}_D\dot{\tilde{\mathbf{x}}}) + \mathbf{K}_p\tilde{\mathbf{x}} - \mathbf{M}_d\mathbf{J}_{A_d}\dot{\mathbf{q}} + \mathbf{M}_d\dot{\mathbf{b}} - \mathbf{h}_e^d$$

7.5 Controllo di Ammettenza

Nel controllo di impedenza c'è un problema nella scelta dei parametri. In particolare se scelgo un $\mathbf{K}_p$ piccolo, quindi il robot diventa cedevole, ma perdo anceh tutti i benefici del feedback. Da qui nasce la necessità di usare il controllo di Ammettenza.



Nello schema di sopra, l'uscite sono le coppie ai giunti, mentre nello schema di sotto la prima uscita è la posizione dell'organo terminale, se il robot fosse controllato con quella impedenza.

7.6 Controllo di Forza

Quello che vogliamo fare sarà:

- Usare un controllo PD sulla forza
- Creare un loop esterno per la regolazione della forza, che sarà ingresso ad un controllo di posizione model-based

$$\mathbf{f}_e = \mathbf{K}(\mathbf{x}_e - \mathbf{x}_{rest})$$

Otteniamo così il seguente controllore di forza:

$$\mathbf{M}_d\ddot{\mathbf{x}}_e + \mathbf{K}_D\dot{\mathbf{x}}_e + \mathbf{K}_p\mathbf{x}_e = \mathbf{K}_p\mathbf{x}_F$$

Con $\mathbf{x}_F$ un parametro del controllo da decidere, attraverso:

$$\mathbf{x}_F = \mathbf{C}_F(\mathbf{f}_d - \mathbf{f}_e)$$

Quindi x_F è dipendente dall'errore di forza, quindi è un loop esterno. Per quanto riguarda $\mathbf{C}_F$:

- Algebrico, a regime non ha errore nullo
- PI: $\mathbf{C}_F = \mathbf{K}_F + \mathbf{K}_I \int^T$

Quindi a regime avrò che la forza all'organo terminale sarà la forza desiderata.